Sniper

Generated by Doxygen 1.8.17

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 CacheEfficiencyTracker Namespace Reference

Classes

- struct **Callbacks**

Typedefs

- typedef **UInt64**(∗ **CallbackGetOwner**) (**UInt64** user, **core_id_t** core_id, **UInt64** address)
- typedef void(∗ **CallbackNotifyAccess**) (**UInt64** user, **UInt64** owner, **Core::mem_op_t** mem_op_type, **HitWhere::where_t** hit_where)
- typedef void(∗ **CallbackNotifyEvict**) (**UInt64** user, bool on_roi_end, **UInt64** owner, **UInt64** evictor, **CacheBlockInfo::BitsUsedType** bits_used, **UInt32** bits_total)

5.1.1 Typedef Documentation

5.1.1.1 CallbackGetOwner

```
typedef UInt64(* CacheEfficiencyTracker::CallbackGetOwner) ( UInt64 user, core_id_t core_id,
UInt64 address)
```

Definition at line 10 of file cache_efficiency_tracker.h.

5.1.1.2 CallbackNotifyAccess

```
typedef void(* CacheEfficiencyTracker::CallbackNotifyAccess) ( UInt64 user, UInt64 owner,
Core::mem_op_t mem_op_type, HitWhere::where_t hit_where)
```

Definition at line 11 of file cache_efficiency_tracker.h.

5.1.1.3 CallbackNotifyEvict

```
typedef void(* CacheEfficiencyTracker::CallbackNotifyEvict) ( UInt64 user, bool on_roi_end,
UInt64 owner, UInt64 evictor, CacheBlockInfo::BitsUsedType bits_used, UInt32 bits_total)
```

Definition at line 12 of file cache_efficiency_tracker.h.

5.2 config Namespace Reference

Classes

- class **Config**

  Config (p. *328): A class for managing the interface to persistent configuration entries defined at runtime. This class is used to manage a configuration interface. It is the base class for which different back ends will derive from.*
- struct **config_parser**
- class **ConfigFile**

  ConfigFile (p. *362): A flat-file interface for the* ***Config*** (p. *328) Class. This file contains the class that is used to interface a flat file for config input / output. It uses boost::spirit for the config grammar.*
- class **FileNotFound**
- class **Key**

  Key (p. *669): A configuration setting entry This class is used to hold a given setting from a configuration. It contains the actual data, as well as functions to get the type.*
- class **KeyNotFound**
- class **parserError**
- class **SaveError**
- class **Section**

  Section (p. *1163): A configuration section entry This class is used to hold a given section from a configuration. It contains both a list of subsections as well as a list of keys. The root of a configuration tree is of this class type.*

Typedefs

- typedef std::vector< String > **PathElementList**
- typedef std::pair< String, String > **PathPair**
- typedef int_parser< **SInt64**, 10, 1, -1 > **long_parser_t**
- typedef tree_parse_info< **config_parser::iterator_t**, **config_parser::factory_t** > **parse_info_t**
- typedef tree_match< **config_parser::iterator_t**, **config_parser::factory_t** > **tree_match_t**
- typedef tree_match_t::node_t **node_t**
- typedef tree_match_t::const_tree_iterator **tree_iter_t**
- typedef std::map< String, **Section** * > **SectionList**
- typedef std::map< String, **Key** * > **KeyList**
- typedef std::map< String, std::vector< **Key** * > > **KeyArrayList**

Enumerations

- enum **RuleID** {
  defaultID, **sectionID**, **keyNameID**, **keyValueID**,
  keySeparatorID, **keyValueArrayID**, **keyValueSpanID**, **keyID**,
  sectionNameID, **stringID**, **configID** }
- enum **KeyType** { **TYPE_INT_VALID** = 0x01, **TYPE_FLOAT_VALID** = 0x02, **TYPE_STRING_VALID** = 0x04, **TYPE_BOOL_VALID** = 0x08 }

  Enumeration for the types a given key may be.

Functions

- void **Error** (const char *format,...)
- **__attribute__** ((__noreturn__)) void **Error**(const char *msg

Variables

- static **long_parser_t long_p**
- static **long_parser_t long_p**

5.2.1 Typedef Documentation

5.2.1.1 KeyArrayList

```
typedef std::map< String, std::vector< Key* > >  config::KeyArrayList
```

Definition at line 21 of file section.hpp.

5.2.1.2 KeyList

```
typedef std::map< String,  Key* >  config::KeyList
```

Definition at line 20 of file section.hpp.

5.2.1.3 long_parser_t

```
typedef int_parser<  SInt64, 10, 1, -1 >  config::long_parser_t
```

Definition at line 43 of file config_file_grammar.hpp.

5.2.1.4 node_t

```
typedef tree_match_t::node_t  config::node_t
```

Definition at line 156 of file config_file_grammar.hpp.

5.2.1.5 parse_info_t

typedef tree_parse_info< **config_parser::iterator_t**, **config_parser::factory_t**> **config::parse_info_t**

Definition at line 154 of file config_file_grammar.hpp.

5.2.1.6 PathElementList

typedef std::vector< String > **config::PathElementList**

Definition at line 27 of file config.hpp.

5.2.1.7 PathPair

typedef std::pair<String,String> **config::PathPair**

Definition at line 31 of file config.hpp.

5.2.1.8 SectionList

typedef std::map< String, **Section*** > **config::SectionList**

Definition at line 18 of file section.hpp.

5.2.1.9 tree_iter_t

typedef tree_match_t::const_tree_iterator **config::tree_iter_t**

Definition at line 157 of file config_file_grammar.hpp.

5.2.1.10 tree_match_t

typedef tree_match< **config_parser::iterator_t**, **config_parser::factory_t**> **config::tree_match_t**

Definition at line 155 of file config_file_grammar.hpp.

5.2.2 Enumeration Type Documentation

5.2.2.1 KeyType

enum **config::KeyType**

Enumeration for the types a given key may be.

Enumerator

TYPE_INT_VALID	
TYPE_FLOAT_VALID	
TYPE_STRING_VALID	
TYPE_BOOL_VALID	

Definition at line 15 of file key.hpp.

5.2.2.2 RuleID

```
enum config::RuleID
```

Enumerator

defaultID	
sectionID	
keyNameID	
keyValueID	
keySeparatorID	
keyValueArrayID	
keyValueSpanID	
keyID	
sectionNameID	
stringID	
configID	

Definition at line 40 of file config_file_grammar.hpp.

5.2.3 Function Documentation

5.2.3.1 __attribute__()

```
config::__attribute__ (
            (__noreturn__)  ) const
```

5.2.3.2 Error()

```
void config::Error (
            const char * format,
             ...  )
```

Definition at line 16 of file config.cpp.

Referenced by config::Config::getKey(), Config::getNearestAcceptableCoreCount(), Config::getNetworkModels(), ClockSkewMinimizationObject::parseScheme(), and Config::setCacheEfficiencyCallbacks().

5.2.4 Variable Documentation

5.2.4.1 long_p [1/2]

```
long_parser_t config::long_p  [static]
```

Definition at line 25 of file key.cpp.

5.2.4.2 long_p [2/2]

```
long_parser_t config::long_p  [static]
```

Definition at line 44 of file config_file_grammar.hpp.

Referenced by config::config_parser::definition< ScannerT >::definition(), and config::Key::DetermineType().

5.3 FastNehalem Namespace Reference

Classes

- class **Cache**
- class **CacheBase**
- class **CacheLocked**
- class **CacheSet**
- class **Dram**
- class **MemoryManager**

5.4 InstrumentLevel Namespace Reference

Enumerations

- enum **Level** { **INSTR_WITH_BBVS**, **INSTR**, **NONE** }

5.4.1 Enumeration Type Documentation

5.4.1.1 Level

```
enum InstrumentLevel::Level
```

Enumerator

INSTR_WITH_BBVS	
INSTR	
NONE	

Definition at line 10 of file sampling_provider.h.

5.5 lite Namespace Reference

Classes

- struct **pthread_functions_t**

Functions

- void **handleSyscall** (THREADID threadIndex, CONTEXT ∗ctx)
- void **syscallEnterRunModel** (THREADID threadIndex, CONTEXT ∗ctx, SYSCALL_STANDARD syscall_↩ standard, void ∗v)
- void **syscallExitRunModel** (THREADID threadIndex, CONTEXT ∗ctx, SYSCALL_STANDARD syscall_↩ standard, void ∗v)
- bool **interceptSignal** (THREADID threadIndex, INT32 signal, CONTEXT ∗ctx, BOOL hasHandler, const EXCEPTION_INFO ∗pExceptInfo, void ∗v)
- void **addMemoryModeling** (TRACE trace, INS ins, **InstMode::inst_mode_t** inst_mode)
- void **handleMemoryRead** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_update, **IntPtr** read_address, **UInt32** read_data_size)
- void **handleMemoryReadDetailed** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic↩ _update, **IntPtr** read_address, **UInt32** read_data_size)
- void **handleMemoryReadDetailedIssue** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_↩ atomic_update, **IntPtr** read_address, **UInt32** read_data_size)
- ADDRINT **handleMemoryReadFaultinjectionNondetailed** (bool is_atomic_update, ADDRINT read_↩ address, ADDRINT ∗save_ea)
- ADDRINT **handleMemoryReadFaultinjection** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_update, ADDRINT read_address, **UInt32** read_data_size, **UInt32** op_num, ADDRINT ∗save_ea)
- void **completeMemoryWrite** (bool is_atomic_update, ADDRINT write_address, ADDRINT scratch, UINT32 write_size)
- void **handleMemoryWrite** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_update, **IntPtr** write_address, **UInt32** write_data_size)
- void **handleMemoryWriteDetailed** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic↩ _update, **IntPtr** write_address, **UInt32** write_data_size)
- void **handleMemoryWriteDetailedIssue** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_↩ atomic_update, **IntPtr** write_address, **UInt32** write_data_size)
- void **handleMemoryWriteFaultinjection** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is↩ _atomic_update, **IntPtr** write_address, **UInt32** write_data_size)
- void **printStackTrace** (THREADID threadid, char ∗function, BOOL enter)
- void **routineStartCallback** (RTN rtn, INS ins)
- void **routineCallback** (RTN rtn, void ∗v)
- AFUNPTR **getFunptr** (CONTEXT ∗context, string func_name)
- **IntPtr nullFunction** ()
- void **pthreadBefore** (THREADID thread_id)
- void **pthreadAfter** (THREADID thread_id, ADDRINT type_id, ADDRINT retval)

- void **mallocBefore** (THREADID thread_id, ADDRINT eip, ADDRINT size)
- void **mallocAfter** (THREADID thread_id, ADDRINT address)
- void **freeBefore** (THREADID thread_id, ADDRINT eip, ADDRINT address)
- **IntPtr emuGetNprocs** ()
- **IntPtr emuGetCPU** (THREADID thread_id)
- **IntPtr emuClockGettime** (THREADID thread_id, clockid_t clk_id, struct timespec ∗tp)
- **IntPtr emuGettimeofday** (THREADID thread_id, struct timeval ∗tv, struct timezone ∗tz)
- void **emuKmpReapMonitor** (THREADID threadIndex, CONTEXT ∗ctxt)

Variables

- **Lock g_atomic_lock**
- char **g_zeros** [1024] = { 0 }
- std::unordered_map< **core_id_t**, **SubsecondTime** > **pthread_t_start**
- AFUNPTR **ptr_exit** = NULL
- struct **lite::pthread_functions_t pthread_functions** [ ]

5.5.1 Function Documentation

5.5.1.1 addMemoryModeling()

```
void lite::addMemoryModeling (
             TRACE trace,
             INS ins,
              InstMode::inst_mode_t inst_mode )
```

Definition at line 20 of file memory_modeling.cc.

References completeMemoryWrite(), g_toolregs, handleMemoryRead(), handleMemoryReadDetailed(), handleMemoryReadDetailedIssue(), handleMemoryReadFaultinjection(), handleMemoryReadFaultinjectionNondetailed(), handleMemoryWrite(), handleMemoryWriteDetailed(), handleMemoryWriteDetailedIssue(), handleMemoryWriteFaultinjection(), INSTR_IF_CACHEONLY, INSTR_IF_DETAILED, INSTR_IF_FASTFORWARD, INSTRUMENT, LOG_ASSERT_ERROR, ThreadLocalStorage::NUM_SCRATCHPADS, TOOLREG_EA0, TOOLREG_MEM0, and TOOLREG_NUM_MEM.

Referenced by InstructionModeling::addInstructionModeling().

5.5.1.2 completeMemoryWrite()

```
void lite::completeMemoryWrite (
             bool is_atomic_update,
             ADDRINT write_address,
             ADDRINT scratch,
             UINT32 write_size )
```

Definition at line 284 of file memory_modeling.cc.

References g_atomic_lock, and TLock< T_LockCreator >::release().

Referenced by addMemoryModeling().

5.5.1.3 emuClockGettime()

```
IntPtr lite::emuClockGettime (
            THREADID thread_id,
            clockid_t clk_id,
            struct timespec * tp )
```

Definition at line 391 of file routine_replace.cc.

References PerformanceModel::getElapsedTime(), SubsecondTime::getNS(), Core::getPerformanceModel(), and localStore.

Referenced by routineCallback().

5.5.1.4 emuGetCPU()

```
IntPtr lite::emuGetCPU (
            THREADID thread_id )
```

Definition at line 384 of file routine_replace.cc.

References Core::getId(), and localStore.

Referenced by routineCallback().

5.5.1.5 emuGetNprocs()

```
IntPtr lite::emuGetNprocs ( )
```

Definition at line 377 of file routine_replace.cc.

Referenced by routineCallback().

5.5.1.6 emuGettimeofday()

```
IntPtr lite::emuGettimeofday (
            THREADID thread_id,
            struct timeval * tv,
            struct timezone * tz )
```

Definition at line 415 of file routine_replace.cc.

References PerformanceModel::getElapsedTime(), SubsecondTime::getNS(), Core::getPerformanceModel(), localStore, LOG_ASSERT_ERROR, and LOG_ASSERT_WARNING_ONCE.

Referenced by routineCallback().

5.5.1.7 emuKmpReapMonitor()

```
void lite::emuKmpReapMonitor (
            THREADID threadIndex,
            CONTEXT * ctxt )
```

Definition at line 431 of file routine_replace.cc.

References ptr_exit.

Referenced by routineStartCallback().

5.5.1.8 freeBefore()

```
void lite::freeBefore (
            THREADID thread_id,
            ADDRINT eip,
            ADDRINT address )
```

Definition at line 372 of file routine_replace.cc.

References localStore.

Referenced by routineCallback().

5.5.1.9 getFunptr()

```
AFUNPTR lite::getFunptr (
            CONTEXT * context,
            string func_name )
```

Definition at line 318 of file routine_replace.cc.

5.5.1.10 handleMemoryRead()

```
void lite::handleMemoryRead (
            THREADID thread_id,
            BOOL executing,
            ADDRINT eip,
            bool is_atomic_update,
             IntPtr read_address,
             UInt32 read_data_size )
```

Definition at line 179 of file memory_modeling.cc.

References Core::accessMemoryFast(), localStore, and Core::READ.

Referenced by addMemoryModeling().

5.5.1.11 handleMemoryReadDetailed()

```
void lite::handleMemoryReadDetailed (
             THREADID thread_id,
             BOOL executing,
             ADDRINT eip,
             bool is_atomic_update,
              IntPtr read_address,
              UInt32 read_data_size )
```

Definition at line 187 of file memory_modeling.cc.

References localStore, Operand::READ, HitWhere::UNKNOWN, and SubsecondTime::Zero().

Referenced by addMemoryModeling().

5.5.1.12 handleMemoryReadDetailedIssue()

```
void lite::handleMemoryReadDetailedIssue (
             THREADID thread_id,
             BOOL executing,
             ADDRINT eip,
             bool is_atomic_update,
              IntPtr read_address,
              UInt32 read_data_size )
```

Definition at line 194 of file memory_modeling.cc.

References Core::accessMemory(), MemoryResult::hit_where, MemoryResult::latency, localStore, Core::LOCK, makeMemoryResult(), Core::MEM_MODELED_COUNT, Core::MEM_MODELED_RETURN, Core::NONE, Hit↩ Where::PREDICATE_FALSE, Operand::READ, Core::READ, Core::READ_EX, and SubsecondTime::Zero().

Referenced by addMemoryModeling().

5.5.1.13 handleMemoryReadFaultinjection()

```
ADDRINT lite::handleMemoryReadFaultinjection (
             THREADID thread_id,
             BOOL executing,
             ADDRINT eip,
             bool is_atomic_update,
             ADDRINT read_address,
              UInt32 read_data_size,
              UInt32 op_num,
             ADDRINT * save_ea )
```

Definition at line 233 of file memory_modeling.cc.

References Core::accessMemory(), TLock< T_LockCreator >::acquire(), g_atomic_lock, MemoryResult::hit_↩ where, MemoryResult::latency, localStore, SubsecondTime::MaxTime(), Core::MEM_MODELED_COUNT, Core::↩ MEM_MODELED_RETURN, Core::NONE, HitWhere::PREDICATE_FALSE, Operand::READ, Core::READ, Core↩ ::READ_EX, and SubsecondTime::Zero().

Referenced by addMemoryModeling().

5.5.1.14 handleMemoryReadFaultinjectionNondetailed()

```
ADDRINT lite::handleMemoryReadFaultinjectionNondetailed (
             bool is_atomic_update,
             ADDRINT read_address,
             ADDRINT * save_ea )
```

Definition at line 223 of file memory_modeling.cc.

References TLock< T_LockCreator >::acquire(), and g_atomic_lock.

Referenced by addMemoryModeling().

5.5.1.15 handleMemoryWrite()

```
void lite::handleMemoryWrite (
             THREADID thread_id,
             BOOL executing,
             ADDRINT eip,
             bool is_atomic_update,
              IntPtr write_address,
              UInt32 write_data_size )
```

Definition at line 292 of file memory_modeling.cc.

References Core::accessMemoryFast(), localStore, Core::logMemoryHit(), Core::MEM_MODELED_COUNT, and Core::WRITE.

Referenced by addMemoryModeling().

5.5.1.16 handleMemoryWriteDetailed()

```
void lite::handleMemoryWriteDetailed (
             THREADID thread_id,
             BOOL executing,
             ADDRINT eip,
             bool is_atomic_update,
              IntPtr write_address,
              UInt32 write_data_size )
```

Definition at line 304 of file memory_modeling.cc.

References HitWhere::L1_OWN, localStore, Core::logMemoryHit(), Core::MEM_MODELED_RETURN, HitWhere↩ ::UNKNOWN, Operand::WRITE, Core::WRITE, and SubsecondTime::Zero().

Referenced by addMemoryModeling().

5.5.1.17 handleMemoryWriteDetailedIssue()

```
void lite::handleMemoryWriteDetailedIssue (
            THREADID thread_id,
            BOOL executing,
            ADDRINT eip,
            bool is_atomic_update,
             IntPtr write_address,
             UInt32 write_data_size )
```

Definition at line 327 of file memory_modeling.cc.

References Core::accessMemory(), MemoryResult::hit_where, MemoryResult::latency, localStore, makeMemoryResult(), Core::MEM_MODELED_COUNT, Core::MEM_MODELED_RETURN, Core::NONE, HitWhere::PREDICATE_FALSE, Core::UNLOCK, Operand::WRITE, Core::WRITE, and SubsecondTime::Zero().

Referenced by addMemoryModeling().

5.5.1.18 handleMemoryWriteFaultinjection()

```
void lite::handleMemoryWriteFaultinjection (
            THREADID thread_id,
            BOOL executing,
            ADDRINT eip,
            bool is_atomic_update,
             IntPtr write_address,
             UInt32 write_data_size )
```

Definition at line 357 of file memory_modeling.cc.

References Core::accessMemory(), g_zeros, MemoryResult::hit_where, HitWhere::L1_OWN, MemoryResult::latency, localStore, Core::logMemoryHit(), SubsecondTime::MaxTime(), Core::MEM_MODELED_COUNT, Core::MEM_MODELED_NONE, Core::MEM_MODELED_RETURN, Core::NONE, HitWhere::PREDICATE_FALSE, Operand::WRITE, Core::WRITE, and SubsecondTime::Zero().

Referenced by addMemoryModeling().

5.5.1.19 handleSyscall()

```
void lite::handleSyscall (
            THREADID threadIndex,
            CONTEXT * ctx )
```

Definition at line 17 of file handle_syscalls.cc.

References SyscallMdl::syscall_args_t::arg0, SyscallMdl::syscall_args_t::arg1, SyscallMdl::syscall_args_t::arg2, SyscallMdl::syscall_args_t::arg3, SyscallMdl::syscall_args_t::arg4, SyscallMdl::syscall_args_t::arg5, Thread::clear_tid, Thread::getSyscallMdl(), localStore, LOG_ASSERT_ERROR, LOG_PRINT, LOG_PRINT_ERROR, Thread::m_os_info, SyscallMdl::runEnter(), and Thread::tid_ptr.

Referenced by InstructionModeling::addInstructionModeling().

5.5.1.20 interceptSignal()

```
bool lite::interceptSignal (
            THREADID threadIndex,
            INT32 signal,
            CONTEXT * ctx,
            BOOL hasHandler,
            const EXCEPTION_INFO * pExceptInfo,
            void * v )
```

Definition at line 107 of file handle_syscalls.cc.

Referenced by main().

5.5.1.21 mallocAfter()

```
void lite::mallocAfter (
            THREADID thread_id,
            ADDRINT address )
```

Definition at line 363 of file routine_replace.cc.

References localStore.

Referenced by routineCallback().

5.5.1.22 mallocBefore()

```
void lite::mallocBefore (
            THREADID thread_id,
            ADDRINT eip,
            ADDRINT size )
```

Definition at line 357 of file routine_replace.cc.

References localStore.

Referenced by routineCallback().

5.5.1.23 nullFunction()

```
IntPtr lite::nullFunction ( )
```

Definition at line 330 of file routine_replace.cc.

References LOG_PRINT.

Referenced by routineCallback().

5.5.1.24 printStackTrace()

```
void lite::printStackTrace (
            THREADID threadid,
            char * function,
            BOOL enter )
```

Definition at line 45 of file routine_replace.cc.

Referenced by routineCallback().

5.5.1.25 pthreadAfter()

```
void lite::pthreadAfter (
            THREADID thread_id,
            ADDRINT type_id,
            ADDRINT retval )
```

Definition at line 344 of file routine_replace.cc.

References PerformanceModel::getElapsedTime(), Core::getPerformanceModel(), localStore, pthread_functions, pthread_t_start, PthreadEmu::pthreadCount(), lite::pthread_functions_t::state_after, PthreadEmu::STATE_BY_↩RETURN, PthreadEmu::STATE_INREGION, PthreadEmu::STATE_RUNNING, PthreadEmu::updateState(), and SubsecondTime::Zero().

Referenced by routineCallback().

5.5.1.26 pthreadBefore()

```
void lite::pthreadBefore (
            THREADID thread_id )
```

Definition at line 336 of file routine_replace.cc.

References PerformanceModel::getElapsedTime(), Core::getPerformanceModel(), localStore, pthread_t_start, PthreadEmu::STATE_WAITING, and PthreadEmu::updateState().

Referenced by routineCallback().

5.5.1.27 routineCallback()

```
void lite::routineCallback (
            RTN rtn,
            void * v )
```

Definition at line 66 of file routine_replace.cc.

References addRtnTracer(), PthreadEmu::BarrierInit(), PthreadEmu::BarrierWait(), CarbonBarrierInit(), CarbonBarrierWait(), CarbonCondBroadcast(), CarbonCondInit(), CarbonCondSignal(), CarbonCondWait(), CarbonMutexInit(), CarbonMutexLock(), CarbonMutexUnlock(), PthreadEmu::CondBroadcast(), PthreadEmu::CondInit(), PthreadEmu::CondSignal(), PthreadEmu::CondWait(), emuClockGettime(), emuGetCPU(), emuGetNprocs(), emuGettimeofday(), freeBefore(), mallocAfter(), mallocBefore(), PthreadEmu::MutexInit(), PthreadEmu::MutexLock(), PthreadEmu::MutexTrylock(), PthreadEmu::MutexUnlock(), lite::pthread_functions_t::name, nullFunction(), printStackTrace(), pthread_functions, pthreadAfter(), pthreadBefore(), and ptr_exit.

Referenced by main().

5.5.1.28 routineStartCallback()

```
void lite::routineStartCallback (
            RTN rtn,
            INS ins )
```

Definition at line 50 of file routine_replace.cc.

References emuKmpReapMonitor().

Referenced by traceCallback().

5.5.1.29 syscallEnterRunModel()

```
void lite::syscallEnterRunModel (
            THREADID threadIndex,
            CONTEXT * ctx,
            SYSCALL_STANDARD syscall_standard,
            void * v )
```

Definition at line 72 of file handle_syscalls.cc.

References Thread::getSyscallMdl(), SyscallMdl::isEmulated(), localStore, and LOG_ASSERT_ERROR.

Referenced by main().

5.5.1.30 syscallExitRunModel()

```
void lite::syscallExitRunModel (
            THREADID threadIndex,
            CONTEXT * ctx,
            SYSCALL_STANDARD syscall_standard,
            void * v )
```

Definition at line 82 of file handle_syscalls.cc.

References SyscallMdl::getCurrentSyscallNumber(), Thread::getSyscallMdl(), SyscallMdl::isEmulated(), localStore, LOG_ASSERT_ERROR, LOG_PRINT, Thread::m_os_info, SyscallMdl::runExit(), and Thread::tid.

Referenced by main().

5.5.2 Variable Documentation

5.5.2.1 g_atomic_lock

```
Lock lite::g_atomic_lock
```

Definition at line 18 of file memory_modeling.cc.

Referenced by completeMemoryWrite(), handleMemoryReadFaultinjection(), and handleMemoryRead↩FaultinjectionNondetailed().

5.5.2.2 g_zeros

```
char lite::g_zeros[1024] = { 0 }
```

Definition at line 355 of file memory_modeling.cc.

Referenced by handleMemoryWriteFaultinjection().

5.5.2.3 pthread_functions

```
struct lite::pthread_functions_t lite::pthread_functions[]
```

Initial value:

```
= {
   { "pthread_mutex_lock",      PthreadEmu::PTHREAD_MUTEX_LOCK,     PthreadEmu::STATE_INREGION },
   { "pthread_mutex_trylock",   PthreadEmu::PTHREAD_MUTEX_TRYLOCK,  PthreadEmu::STATE_BY_RETURN },
   { "pthread_mutex_unlock",    PthreadEmu::PTHREAD_MUTEX_UNLOCK,   PthreadEmu::STATE_RUNNING },
   { "pthread_cond_wait",       PthreadEmu::PTHREAD_COND_WAIT,      PthreadEmu::STATE_RUNNING },
   { "pthread_cond_signal",     PthreadEmu::PTHREAD_COND_SIGNAL,    PthreadEmu::STATE_RUNNING },
   { "pthread_cond_broadcast",  PthreadEmu::PTHREAD_COND_BROADCAST, PthreadEmu::STATE_RUNNING },
   { "pthread_barrier_wait",    PthreadEmu::PTHREAD_BARRIER_WAIT,   PthreadEmu::STATE_RUNNING },
}
```

Referenced by pthreadAfter(), and routineCallback().

5.5.2.4 pthread_t_start

```
std::unordered_map< core_id_t,  SubsecondTime> lite::pthread_t_start
```

Definition at line 28 of file routine_replace.cc.

Referenced by pthreadAfter(), and pthreadBefore().

5.5.2.5 ptr_exit

```
AFUNPTR lite::ptr_exit = NULL
```

Definition at line 29 of file routine_replace.cc.

Referenced by emuKmpReapMonitor(), and routineCallback().

5.6 Memory Namespace Reference

Classes

- struct **Access**

Functions

- **Access make_access** (**UInt64** address)

5.6.1 Function Documentation

5.6.1.1 make_access()

```
Access Memory::make_access (
            UInt64 address )  [inline]
```

Definition at line 21 of file memory_access.h.

References Memory::Access::set().

Referenced by InstructionDecoder::decode().

5.7 ParametricDramDirectoryMSI Namespace Reference

Classes

- class **CacheCntlr**
- class **CacheCntlrList**
- class **CacheDirectoryWaiter**
- class **CacheMasterCntlr**
- class **CacheParameters**
- class **MemoryManager**
- struct **MshrEntry**
- class **Prefetch**
- class **TLB**
- class **Transition**

Typedefs

- typedef **ReqQueueListTemplate**< **CacheDirectoryWaiter** > **CacheDirectoryWaiterMap**
- typedef std::unordered_map< **IntPtr**, **MshrEntry** > **Mshr**
- typedef std::pair< **core_id_t**, **MemComponent::component_t** > **CoreComponentType**
- typedef std::map< **CoreComponentType**, **CacheCntlr** * > **CacheCntlrMap**

Functions

- char **CStateString** (**CacheState::cstate_t** cstate)
- const char * **ReasonString** (**Transition::reason_t** reason)
- **MshrEntry** **make_mshr** (**SubsecondTime** t_issue, **SubsecondTime** t_complete)

5.7.1 Typedef Documentation

5.7.1.1 CacheCntlrMap

```
typedef std::map< CoreComponentType, CacheCntlr*> ParametricDramDirectoryMSI::CacheCntlrMap
```

Definition at line 27 of file memory_manager.h.

5.7.1.2 CacheDirectoryWaiterMap

```
typedef ReqQueueListTemplate< CacheDirectoryWaiter> ParametricDramDirectoryMSI::Cache↩
DirectoryWaiterMap
```

Definition at line 142 of file cache_cntlr.h.

5.7.1.3 CoreComponentType

```
typedef std::pair< core_id_t, MemComponent::component_t> ParametricDramDirectoryMSI::Core↩
ComponentType
```

Definition at line 24 of file memory_manager.h.

5.7.1.4 Mshr

```
typedef std::unordered_map< IntPtr, MshrEntry> ParametricDramDirectoryMSI::Mshr
```

Definition at line 147 of file cache_cntlr.h.

5.7.2 Function Documentation

5.7.2.1 CStateString()

```
char ParametricDramDirectoryMSI::CStateString (
            CacheState::cstate_t cstate )
```

Definition at line 38 of file cache_cntlr.cc.

References CacheState::EXCLUSIVE, CacheState::INVALID, CacheState::INVALID_COHERENCY, CacheState↩ ::INVALID_COLD, CacheState::INVALID_EVICT, CacheState::MODIFIED, CacheState::OWNED, CacheState::S↩ HARED, and CacheState::SHARED_UPGRADING.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), ParametricDramDirectoryMSI::Cache↩ Cntlr::insertCacheBlock(), ParametricDramDirectoryMSI::CacheCntlr::invalidateCacheBlock(), Parametric↩ DramDirectoryMSI::CacheCntlr::notifyPrevLevelEvict(), ParametricDramDirectoryMSI::CacheCntlr::operation↩ PermissibleinCache(), ParametricDramDirectoryMSI::CacheCntlr::printCache(), ParametricDramDirectoryMSI::↩ CacheCntlr::processInvReqFromDramDirectory(), ParametricDramDirectoryMSI::CacheCntlr::processUpgrade↩ RepFromDramDirectory(), and ParametricDramDirectoryMSI::CacheCntlr::updateCacheBlock().

5.7.2.2 make_mshr()

```
MshrEntry ParametricDramDirectoryMSI::make_mshr (
            SubsecondTime t_issue,
            SubsecondTime t_complete )
```

Definition at line 68 of file cache_cntlr.cc.

References ParametricDramDirectoryMSI::MshrEntry::t_complete, and ParametricDramDirectoryMSI::MshrEntry↩ ::t_issue.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), and Parametric↩ DramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache().

5.7.2.3 ReasonString()

```
const char* ParametricDramDirectoryMSI::ReasonString (
              Transition::reason_t reason )
```

Definition at line 54 of file cache_cntlr.cc.

References ParametricDramDirectoryMSI::Transition::BACK_INVAL, ParametricDramDirectoryMSI::Transition::COHERENCY, ParametricDramDirectoryMSI::Transition::CORE_RD, ParametricDramDirectoryMSI::Transition::CORE_RDEX, ParametricDramDirectoryMSI::Transition::CORE_WR, ParametricDramDirectoryMSI::Transition::EVICT, and ParametricDramDirectoryMSI::Transition::UPGRADE.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr().

5.8 PrL1PrL2DramDirectoryMSI Namespace Reference

Classes

- class **DramCntlr**
- class **DramDirectoryCache**
- class **DramDirectoryCntlr**
- class **ShmemMsg**
- class **ShmemReq**

Typedefs

- typedef **ReqQueueListTemplate**< **ShmemReq** > **ReqQueueList**

Functions

- char **DStateString** (**DirectoryState::dstate_t** state)

5.8.1 Typedef Documentation

5.8.1.1 ReqQueueList

```
typedef ReqQueueListTemplate< ShmemReq> PrL1PrL2DramDirectoryMSI::ReqQueueList
```

Definition at line 11 of file req_queue_list.h.

5.8.2 Function Documentation

5.8.2.1 DStateString()

```
char PrL1PrL2DramDirectoryMSI::DStateString (
              DirectoryState::dstate_t state )
```

Definition at line 22 of file dram_directory_cntlr.cc.

References DirectoryState::EXCLUSIVE, DirectoryState::MODIFIED, DirectoryState::OWNED, DirectoryState::SHARED, and DirectoryState::UNCACHED.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::DramDirectoryCntlr().

5.9 PthreadEmu Namespace Reference

Classes

- struct **pthread_counters_t**

Enumerations

- enum **pthread_enum_t** {
  PTHREAD_MUTEX_LOCK = 0, **PTHREAD_MUTEX_TRYLOCK**, **PTHREAD_MUTEX_UNLOCK**, **PTHREAD_COND_WAIT**,
  PTHREAD_COND_SIGNAL, **PTHREAD_COND_BROADCAST**, **PTHREAD_BARRIER_WAIT**, **PTHREAD_ENUM_LAST** }
- enum **state_t** {
  STATE_STOPPED, **STATE_RUNNING**, **STATE_WAITING**, **STATE_INREGION**,
  STATE_SEPARATOR, **STATE_MAX**, **STATE_BY_RETURN** }

Functions

- void **pthreadCount** (**pthread_enum_t** function, **Core** *core, **SubsecondTime** delay_sync, **SubsecondTime** delay_mem)
- **IntPtr** **futexHbAddress** (pthread_mutex_t *mux)
- void **updateState** (**Core** *core, **state_t** state, **SubsecondTime** delay)
- void **init** ()
- **IntPtr** **MutexInit** (pthread_mutex_t *mux, pthread_mutexattr_t *attributes)
- **IntPtr** **MutexLock** (pthread_mutex_t *mux)
- **IntPtr** **MutexTrylock** (pthread_mutex_t *mux)
- **IntPtr** **MutexUnlock** (pthread_mutex_t *mux)
- **IntPtr** **CondInit** (pthread_cond_t *cond, pthread_condattr_t *attributes)
- **IntPtr** **CondWait** (pthread_cond_t *cond, pthread_mutex_t *mutex)
- **IntPtr** **CondSignal** (pthread_cond_t *cond)
- **IntPtr** **CondBroadcast** (pthread_cond_t *cond)
- **IntPtr** **BarrierInit** (pthread_barrier_t *barrier, pthread_barrierattr_t *attributes, unsigned count)
- **IntPtr** **BarrierWait** (pthread_barrier_t *barrier)

Variables

- bool **pthread_stats_added** = false
- const char * **pthread_names** [ ]
- struct **PthreadEmu::pthread_counters_t** * **pthread_counters** = NULL
- static std::unordered_map< pthread_mutex_t *, **IntPtr** > **futex_map**
- static **Lock** **futex_map_lock**
- static **Lock** **trace_lock**
- static FILE * **trace_fp** = NULL

5.9.1 Enumeration Type Documentation

5.9.1.1 pthread_enum_t

enum **PthreadEmu::pthread_enum_t**

Enumerator

PTHREAD_MUTEX_LOCK	
PTHREAD_MUTEX_TRYLOCK	
PTHREAD_MUTEX_UNLOCK	
PTHREAD_COND_WAIT	
PTHREAD_COND_SIGNAL	
PTHREAD_COND_BROADCAST	
PTHREAD_BARRIER_WAIT	
PTHREAD_ENUM_LAST	

Definition at line 12 of file pthread_emu.h.

5.9.1.2 state_t

enum **PthreadEmu::state_t**

Enumerator

STATE_STOPPED	
STATE_RUNNING	
STATE_WAITING	
STATE_INREGION	
STATE_SEPARATOR	
STATE_MAX	
STATE_BY_RETURN	

Definition at line 19 of file pthread_emu.h.

5.9.2 Function Documentation

5.9.2.1 BarrierInit()

```
IntPtr PthreadEmu::BarrierInit (
            pthread_barrier_t * barrier,
            pthread_barrierattr_t * attributes,
            unsigned count )
```

Definition at line 259 of file pthread_emu.cc.

References Core::accessMemory(), CarbonBarrierInit(), Core::NONE, Core::READ, and Core::WRITE.

Referenced by lite::routineCallback().

5.9.2.2 BarrierWait()

```
IntPtr PthreadEmu::BarrierWait (
            pthread_barrier_t * barrier )
```

Definition at line 282 of file pthread_emu.cc.

References Core::accessMemory(), CarbonBarrierWait(), MemoryResult::latency, Core::MEM_MODELED_FEN↩
CED, Core::NONE, PTHREAD_BARRIER_WAIT, pthreadCount(), Core::READ_EX, STATE_RUNNING, STATE↩
_WAITING, and updateState().

Referenced by lite::routineCallback().

5.9.2.3 CondBroadcast()

```
IntPtr PthreadEmu::CondBroadcast (
            pthread_cond_t * cond )
```

Definition at line 243 of file pthread_emu.cc.

References Core::accessMemory(), CarbonCondBroadcast(), MemoryResult::latency, Core::MEM_MODELED_↩
FENCED, Core::NONE, PTHREAD_COND_BROADCAST, pthreadCount(), and Core::READ_EX.

Referenced by lite::routineCallback().

5.9.2.4 CondInit()

```
IntPtr PthreadEmu::CondInit (
            pthread_cond_t * cond,
            pthread_condattr_t * attributes )
```

Definition at line 190 of file pthread_emu.cc.

References CarbonCondInit().

Referenced by lite::routineCallback().

5.9.2.5 CondSignal()

```
IntPtr PthreadEmu::CondSignal (
            pthread_cond_t * cond )
```

Definition at line 227 of file pthread_emu.cc.

References Core::accessMemory(), CarbonCondSignal(), MemoryResult::latency, Core::MEM_MODELED_FEN↩CED, Core::NONE, PTHREAD_COND_SIGNAL, pthreadCount(), and Core::READ_EX.

Referenced by lite::routineCallback().

5.9.2.6 CondWait()

```
IntPtr PthreadEmu::CondWait (
            pthread_cond_t * cond,
            pthread_mutex_t * mutex )
```

Definition at line 207 of file pthread_emu.cc.

References Core::accessMemory(), CarbonCondWait(), MemoryResult::latency, Core::MEM_MODELED_FENC↩ED, Core::MEM_MODELED_TIME, Core::NONE, PTHREAD_COND_WAIT, pthreadCount(), Core::READ_EX, S↩TATE_RUNNING, STATE_WAITING, and updateState().

Referenced by lite::routineCallback().

5.9.2.7 futexHbAddress()

```
IntPtr PthreadEmu::futexHbAddress (
            pthread_mutex_t * mux )
```

Definition at line 51 of file pthread_emu.cc.

References futex_map, and futex_map_lock.

Referenced by MutexLock(), and MutexUnlock().

5.9.2.8 init()

```
void PthreadEmu::init ( )
```

Definition at line 68 of file pthread_emu.cc.

References __attribute__, LOG_ASSERT_ERROR, pthread_counters, PTHREAD_ENUM_LAST, PTHREAD_M↩UTEX_LOCK, pthread_names, pthread_stats_added, registerStatsMetric(), and trace_fp.

Referenced by Simulator::start(), and ThreadStatsManager::ThreadStats::update().

5.9.2.9 MutexInit()

```
IntPtr PthreadEmu::MutexInit (
            pthread_mutex_t * mux,
            pthread_mutexattr_t * attributes )
```

Definition at line 98 of file pthread_emu.cc.

References CarbonMutexInit().

Referenced by lite::routineCallback().

5.9.2.10 MutexLock()

```
IntPtr PthreadEmu::MutexLock (
            pthread_mutex_t * mux )
```

Definition at line 116 of file pthread_emu.cc.

References Core::accessMemory(), CarbonMutexLock(), futexHbAddress(), Core::getId(), MemoryResult::latency, makeMemoryResult(), Core::MEM_MODELED_FENCED, Core::NONE, pthread_counters, PTHREAD_MUTEX_↩LOCK, PthreadEmu::pthread_counters_t::pthread_mutex_lock_contended, pthreadCount(), Core::READ_EX, ST↩ATE_INREGION, STATE_WAITING, HitWhere::UNKNOWN, updateState(), and SubsecondTime::Zero().

Referenced by lite::routineCallback().

5.9.2.11 MutexTrylock()

```
IntPtr PthreadEmu::MutexTrylock (
            pthread_mutex_t * mux )
```

Definition at line 143 of file pthread_emu.cc.

References Core::accessMemory(), CarbonMutexTrylock(), MemoryResult::latency, SubsecondTime::MaxTime(), Core::MEM_MODELED_FENCED, Core::NONE, PTHREAD_MUTEX_TRYLOCK, pthreadCount(), Core::READ↩_EX, STATE_INREGION, STATE_RUNNING, STATE_WAITING, updateState(), and SubsecondTime::Zero().

Referenced by lite::routineCallback().

5.9.2.12 MutexUnlock()

```
IntPtr PthreadEmu::MutexUnlock (
            pthread_mutex_t * mux )
```

Definition at line 162 of file pthread_emu.cc.

References Core::accessMemory(), CarbonMutexUnlock(), futexHbAddress(), Core::getId(), MemoryResult↩ ::latency, makeMemoryResult(), Core::MEM_MODELED_FENCED, Core::NONE, pthread_counters, PTHRE↩ AD_MUTEX_UNLOCK, PthreadEmu::pthread_counters_t::pthread_mutex_unlock_contended, pthreadCount(), Core::READ_EX, STATE_RUNNING, HitWhere::UNKNOWN, updateState(), and SubsecondTime::Zero().

Referenced by lite::routineCallback().

5.9.2.13 pthreadCount()

```
void PthreadEmu::pthreadCount (
              pthread_enum_t function,
              Core * core,
              SubsecondTime delay_sync,
              SubsecondTime delay_mem )
```

Definition at line 37 of file pthread_emu.cc.

References Core::getId(), PthreadEmu::pthread_counters_t::pthread_count, pthread_counters, PthreadEmu↩ ::pthread_counters_t::pthread_total_delay_mem, and PthreadEmu::pthread_counters_t::pthread_total_delay_↩ sync.

Referenced by BarrierWait(), CondBroadcast(), CondSignal(), CondWait(), MutexLock(), MutexTrylock(), Mutex↩ Unlock(), and lite::pthreadAfter().

5.9.2.14 updateState()

```
void PthreadEmu::updateState (
              Core * core,
              state_t state,
              SubsecondTime delay )
```

Definition at line 60 of file pthread_emu.cc.

References PerformanceModel::getElapsedTime(), Core::getId(), Core::getPerformanceModel(), trace_fp, and trace_lock.

Referenced by BarrierWait(), CondWait(), SyscallMdl::handleFutexCall(), MutexLock(), MutexTrylock(), Mutex↩ Unlock(), lite::pthreadAfter(), and lite::pthreadBefore().

5.9.3 Variable Documentation

5.9.3.1 futex_map

```
std::unordered_map<pthread_mutex_t*, IntPtr> PthreadEmu::futex_map [static]
```

Definition at line 49 of file pthread_emu.cc.

Referenced by futexHbAddress().

5.9.3.2 futex_map_lock

```
Lock PthreadEmu::futex_map_lock [static]
```

Definition at line 50 of file pthread_emu.cc.

Referenced by futexHbAddress().

5.9.3.3 pthread_counters

```
struct PthreadEmu::pthread_counters_t * PthreadEmu::pthread_counters = NULL
```

Referenced by init(), MutexLock(), MutexUnlock(), and pthreadCount().

5.9.3.4 pthread_names

```
const char* PthreadEmu::pthread_names[]
```

Initial value:

```
=
{
   "pthread_mutex_lock", "pthread_mutex_trylock", "pthread_mutex_unlock",
   "pthread_cond_wait", "pthread_cond_signal", "pthread_cond_broadcast",
   "pthread_barrier_wait"
}
```

Definition at line 19 of file pthread_emu.cc.

Referenced by init().

5.9.3.5 pthread_stats_added

```
bool PthreadEmu::pthread_stats_added = false
```

Definition at line 18 of file pthread_emu.cc.

Referenced by init().

5.9.3.6 trace_fp

```
FILE* PthreadEmu::trace_fp = NULL  [static]
```

Definition at line 59 of file pthread_emu.cc.

Referenced by init(), and updateState().

5.9.3.7 trace_lock

```
Lock PthreadEmu::trace_lock  [static]
```

Definition at line 58 of file pthread_emu.cc.

Referenced by updateState().

5.10 std Namespace Reference

Classes

- struct **hash**< **HitWhere::where_t** >
- struct **hash**< **HookType::hook_type_t** >
- struct **hash**< **REG** >
- struct **hash**< **std::deque**< **T** > >

Chapter 6

Class Documentation

6.1 _SELock Class Reference

`#include <setlock.h>`

Inheritance diagram for _SELock:

Public Member Functions

- **_SELock** (**UInt32** core_offset, **UInt32** num_sharers)
- void **acquire_shared** (**UInt32** core_id)
- void **release_shared** (**UInt32** core_id)
- void **downgrade** (**UInt32** core_id)
- void **upgrade** (**UInt32** core_id)

Additional Inherited Members

6.1.1 Detailed Description

Definition at line 41 of file setlock.h.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 _SELock()

```
_SELock::_SELock (
              UInt32 core_offset,
              UInt32 num_sharers )  [inline]
```

Definition at line 44 of file setlock.h.

6.1.3 Member Function Documentation

6.1.3.1 acquire_shared()

```
void _SELock::acquire_shared (
              UInt32 core_id )  [inline]
```

Definition at line 45 of file setlock.h.

References SELock::acquire_shared().

6.1.3.2 downgrade()

```
void _SELock::downgrade (
              UInt32 core_id )  [inline]
```

Definition at line 47 of file setlock.h.

References SELock::downgrade().

6.1.3.3 release_shared()

```
void _SELock::release_shared (
              UInt32 core_id )  [inline]
```

Definition at line 46 of file setlock.h.

References SELock::release_shared().

6.1.3.4 upgrade()

```
void _SELock::upgrade (
              UInt32 core_id )  [inline]
```

Definition at line 48 of file setlock.h.

References SELock::upgrade().

The documentation for this class was generated from the following file:

- common/misc/ **setlock.h**

6.2 _SetLock Class Reference

```
#include <setlock.h>
```

Classes

- class **PersetLock**

Public Member Functions

- **_SetLock** (**UInt32** core_offset, **UInt32** num_sharers)
- void **acquire_exclusive** (void)
- void **release_exclusive** (void)
- void **acquire_shared** (**UInt32** core_id)
- void **release_shared** (**UInt32** core_id)
- void **upgrade** (**UInt32** core_id)
- void **downgrade** (**UInt32** core_id)

Private Member Functions

- class **_SetLock::PersetLock** **__attribute__** ((aligned(64)))

Private Attributes

- std::vector< **PersetLock** > **m_locks**
- **UInt32** **m_core_offset**

6.2.1 Detailed Description

Definition at line 12 of file setlock.h.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 _SetLock()

```
_SetLock::_SetLock (
              UInt32 core_offset,
              UInt32 num_sharers )
```

Definition at line 4 of file setlock.cc.

References TotalTimer::getTimerByStacktrace(), and itostr().

6.2.3 Member Function Documentation

6.2.3.1 __attribute__()

```
class  _SetLock::PersetLock _SetLock::__attribute__ (
             (aligned(64))  )  [private]
```

6.2.3.2 acquire_exclusive()

```
void _SetLock::acquire_exclusive (
             void  )
```

Definition at line 15 of file setlock.cc.

References m_locks.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::acquireStackLock(), and upgrade().

6.2.3.3 acquire_shared()

```
void _SetLock::acquire_shared (
              UInt32 core_id )
```

Definition at line 35 of file setlock.cc.

References m_core_offset, and m_locks.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::acquireLock().

6.2.3.4 downgrade()

```
void _SetLock::downgrade (
              UInt32 core_id )
```

Definition at line 61 of file setlock.cc.

References m_core_offset, and m_locks.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::releaseStackLock().

6.2.3.5 release_exclusive()

```
void _SetLock::release_exclusive (
             void  )
```

Definition at line 27 of file setlock.cc.

References m_locks.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::releaseStackLock().

6.2.3.6 release_shared()

```
void _SetLock::release_shared (
              UInt32 core_id )
```

Definition at line 48 of file setlock.cc.

References m_core_offset, and m_locks.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::releaseLock(), and upgrade().

6.2.3.7 upgrade()

```
void _SetLock::upgrade (
              UInt32 core_id )
```

Definition at line 54 of file setlock.cc.

References acquire_exclusive(), and release_shared().

Referenced by ParametricDramDirectoryMSI::CacheCntlr::acquireStackLock().

6.2.4 Member Data Documentation

6.2.4.1 m_core_offset

UInt32 _SetLock::m_core_offset [private]

Definition at line 35 of file setlock.h.

Referenced by acquire_shared(), downgrade(), and release_shared().

6.2.4.2 m_locks

std::vector< **PersetLock**> _SetLock::m_locks [private]

Definition at line 34 of file setlock.h.

Referenced by acquire_exclusive(), acquire_shared(), downgrade(), release_exclusive(), and release_shared().

The documentation for this class was generated from the following files:

- common/misc/ **setlock.h**
- common/misc/ **setlock.cc**

6.3 _Thread Class Reference

#include <_thread.h>

Inheritance diagram for _Thread:

Public Types

- typedef void(* **ThreadFunc**) (void *)

Public Member Functions

- virtual ~**_Thread** ()
- virtual void **run** ()=0

Static Public Member Functions

- static **_Thread** * **create** (**ThreadFunc** func, void *param)
- static **_Thread** * **create** (**Runnable** *runnable)

6.3.1 Detailed Description

Definition at line 16 of file _thread.h.

6.3.2 Member Typedef Documentation

6.3.2.1 ThreadFunc

```
typedef void(* _Thread::ThreadFunc) (void *)
```

Definition at line 19 of file _thread.h.

6.3.3 Constructor & Destructor Documentation

6.3.3.1 ~_Thread()

```
virtual _Thread::~_Thread ( )  [inline], [virtual]
```

Definition at line 27 of file _thread.h.

6.3.4 Member Function Documentation

6.3.4.1 create() [1/2]

```
static  _Thread* _Thread::create (
              Runnable * runnable )  [inline], [static]
```

Definition at line 22 of file _thread.h.

References create(), and Runnable::threadFunc().

6.3.4.2 create() [2/2]

```
_Thread * _Thread::create (
             ThreadFunc func,
            void * param )  [static]
```

Definition at line 21 of file pin_thread.cc.

Referenced by create(), CoreThread::spawn(), SimThread::spawn(), TraceManager::Monitor::spawn(), and TraceThread::spawn().

6.3.4.3 run()

```
virtual void _Thread::run ( )  [pure virtual]
```

Implemented in **PthreadThread** (p. 954), and **PinThread** (p. 932).

Referenced by CoreThread::spawn(), SimThread::spawn(), and TraceThread::spawn().

The documentation for this class was generated from the following files:

- common/misc/ **_thread.h**
- pin/ **pin_thread.cc**

6.4 Memory::Access Struct Reference

```
#include <memory_access.h>
```

Public Member Functions

- void **set** (**UInt64 address**)

Public Attributes

- union {
  UInt64 address
  UInt64 virt
  UInt64 phys
  };

6.4.1 Detailed Description

Definition at line 6 of file memory_access.h.

6.4.2 Member Function Documentation

6.4.2.1 set()

```
void Memory::Access::set (
              UInt64 address )  [inline]
```

Definition at line 12 of file memory_access.h.

References address.

Referenced by MicroOpPerformanceModel::handleInstruction(), and Memory::make_access().

6.4.3 Member Data Documentation

6.4.3.1 "@10

```
union { ...  }
```

6.4.3.2 address

```
UInt64 Memory::Access::address
```

Definition at line 9 of file memory_access.h.

Referenced by RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), IntervalTimer::issueMemOp(), RobTimer::printRob(), RobSmtTimer::printRob(), set(), and IntervalTimer::simulate().

6.4.3.3 phys

```
UInt64 Memory::Access::phys
```

Definition at line 9 of file memory_access.h.

Referenced by IntervalTimer::blockWindow(), and MemoryDependencies::setDependencies().

6.4.3.4 virt

```
UInt64 Memory::Access::virt
```

Definition at line 9 of file memory_access.h.

The documentation for this struct was generated from the following file:

- common/performance_model/performance_models/micro_op/ **memory_access.h**

6.5 AddressHomeLookup Class Reference

`#include <address_home_lookup.h>`

Public Member Functions

- **AddressHomeLookup** (**UInt32** ahl_param, std::vector< **core_id_t** > &core_list, **UInt32** cache_block_↩ size)
- ~**AddressHomeLookup** ()
- **core_id_t** **getHome** (**IntPtr** address) const
- **IntPtr** **getLinearBlock** (**IntPtr** address) const
- **IntPtr** **getLinearAddress** (**IntPtr** address) const

Private Attributes

- **UInt32** **m_ahl_param**
- **UInt64** **m_ahl_mask**
- std::vector< **core_id_t** > **m_core_list**
- **UInt32** **m_total_modules**
- **UInt32** **m_cache_block_size**

6.5.1 Detailed Description

Definition at line 21 of file address_home_lookup.h.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 AddressHomeLookup()

```
AddressHomeLookup::AddressHomeLookup (
            UInt32 ahl_param,
           std::vector< core_id_t > & core_list,
            UInt32 cache_block_size )
```

Definition at line 4 of file address_home_lookup.cc.

References LOG_ASSERT_ERROR, m_ahl_param, m_cache_block_size, and m_total_modules.

6.5.2.2 ~AddressHomeLookup()

```
AddressHomeLookup::~AddressHomeLookup ( )
```

Definition at line 24 of file address_home_lookup.cc.

6.5.3 Member Function Documentation

6.5.3.1 getHome()

```
core_id_t AddressHomeLookup::getHome (
            IntPtr address ) const
```

Definition at line 29 of file address_home_lookup.cc.

References LOG_ASSERT_ERROR, LOG_PRINT, m_ahl_param, m_core_list, and m_total_modules.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::getHome(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::sendDataToDram(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::sendDataToNUCA().

6.5.3.2 getLinearAddress()

```
IntPtr AddressHomeLookup::getLinearAddress (
            IntPtr address ) const
```

Definition at line 43 of file address_home_lookup.cc.

References getLinearBlock(), m_ahl_mask, and m_ahl_param.

Referenced by CacheBase::splitAddress().

6.5.3.3 getLinearBlock()

```
IntPtr AddressHomeLookup::getLinearBlock (
            IntPtr address ) const
```

Definition at line 38 of file address_home_lookup.cc.

References m_ahl_param, and m_total_modules.

Referenced by getLinearAddress().

6.5.4 Member Data Documentation

6.5.4.1 m_ahl_mask

UInt64 AddressHomeLookup::m_ahl_mask [private]

Definition at line 37 of file address_home_lookup.h.

Referenced by getLinearAddress().

6.5.4.2 m_ahl_param

UInt32 AddressHomeLookup::m_ahl_param [private]

Definition at line 36 of file address_home_lookup.h.

Referenced by AddressHomeLookup(), getHome(), getLinearAddress(), and getLinearBlock().

6.5.4.3 m_cache_block_size

UInt32 AddressHomeLookup::m_cache_block_size [private]

Definition at line 40 of file address_home_lookup.h.

Referenced by AddressHomeLookup().

6.5.4.4 m_core_list

std::vector< **core_id_t**> AddressHomeLookup::m_core_list [private]

Definition at line 38 of file address_home_lookup.h.

Referenced by getHome().

6.5.4.5 m_total_modules

```
UInt32 AddressHomeLookup::m_total_modules  [private]
```

Definition at line 39 of file address_home_lookup.h.

Referenced by AddressHomeLookup(), getHome(), and getLinearBlock().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/ **address_home_lookup.h**
- common/core/memory_subsystem/ **address_home_lookup.cc**

6.6 MemoryTracker::Allocation Struct Reference

Public Member Functions

- **Allocation** ()
- **Allocation** (**UInt64** _size, **AllocationSite** *_site)

Public Attributes

- **UInt64 size**
- **AllocationSite** * **site**

6.6.1 Detailed Description

Definition at line 73 of file memory_tracker.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Allocation() [1/2]

```
MemoryTracker::Allocation::Allocation ( )  [inline]
```

Definition at line 75 of file memory_tracker.h.

6.6.2.2 Allocation() [2/2]

```
MemoryTracker::Allocation::Allocation (
            UInt64 _size,
            AllocationSite * _site )  [inline]
```

Definition at line 76 of file memory_tracker.h.

6.6.3 Member Data Documentation

6.6.3.1 site

```
AllocationSite* MemoryTracker::Allocation::site
```

Definition at line 78 of file memory_tracker.h.

6.6.3.2 size

```
UInt64 MemoryTracker::Allocation::size
```

Definition at line 77 of file memory_tracker.h.

The documentation for this struct was generated from the following file:

- common/system/ **memory_tracker.h**

6.7 MemoryTracker::AllocationSite Struct Reference

Public Member Functions

- **AllocationSite** ()

Public Attributes

- **UInt64 num_allocations**
- **UInt64 total_size**
- **UInt64 total_loads**
- **UInt64 total_stores**
- std::vector< **UInt64** > **hit_where_load**
- std::vector< **UInt64** > **hit_where_store**
- std::unordered_map< **AllocationSite** *, **UInt64** > **evicted_by**

6.7.1 Detailed Description

Definition at line 58 of file memory_tracker.h.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 AllocationSite()

```
MemoryTracker::AllocationSite::AllocationSite ( )  [inline]
```

Definition at line 60 of file memory_tracker.h.

6.7.3 Member Data Documentation

6.7.3.1 evicted_by

```
std::unordered_map< AllocationSite*,  UInt64> MemoryTracker::AllocationSite::evicted_by
```

Definition at line 69 of file memory_tracker.h.

Referenced by MemoryTracker::ce_notify_evict().

6.7.3.2 hit_where_load

```
std::vector< UInt64> MemoryTracker::AllocationSite::hit_where_load
```

Definition at line 68 of file memory_tracker.h.

Referenced by MemoryTracker::ce_notify_access().

6.7.3.3 hit_where_store

```
std::vector< UInt64> MemoryTracker::AllocationSite::hit_where_store
```

Definition at line 68 of file memory_tracker.h.

Referenced by MemoryTracker::ce_notify_access().

6.7.3.4 num_allocations

```
UInt64 MemoryTracker::AllocationSite::num_allocations
```

Definition at line 65 of file memory_tracker.h.

6.7.3.5 total_loads

UInt64 MemoryTracker::AllocationSite::total_loads

Definition at line 67 of file memory_tracker.h.

Referenced by MemoryTracker::ce_notify_access(), and MemoryTracker::∼MemoryTracker().

6.7.3.6 total_size

UInt64 MemoryTracker::AllocationSite::total_size

Definition at line 66 of file memory_tracker.h.

6.7.3.7 total_stores

UInt64 MemoryTracker::AllocationSite::total_stores

Definition at line 67 of file memory_tracker.h.

Referenced by MemoryTracker::ce_notify_access(), and MemoryTracker::∼MemoryTracker().

The documentation for this struct was generated from the following file:

- common/system/ **memory_tracker.h**

6.8 Allocator Class Reference

#include <allocator.h>

Inheritance diagram for Allocator:

Classes

- struct **DataElement**

Public Member Functions

- virtual ~**Allocator** ()
- virtual void * **alloc** (size_t bytes)=0
- virtual void **_dealloc** (void *ptr)=0

Static Public Member Functions

- static void **dealloc** (void *ptr)

6.8.1 Detailed Description

Definition at line 12 of file allocator.h.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 ~Allocator()

```
virtual Allocator::~Allocator ( )  [inline], [virtual]
```

Definition at line 21 of file allocator.h.

6.8.3 Member Function Documentation

6.8.3.1 _dealloc()

```
virtual void Allocator::_dealloc (
            void * ptr )  [pure virtual]
```

Implemented in **TypedAllocator< T, MaxItems >** (p. 1469).

Referenced by dealloc().

6.8.3.2 alloc()

```
virtual void* Allocator::alloc (
            size_t bytes )  [pure virtual]
```

Implemented in **TypedAllocator< T, MaxItems >** (p. 1469).

6.8.3.3 dealloc()

```
static void Allocator::dealloc (
            void * ptr )  [inline], [static]
```

Definition at line 26 of file allocator.h.

References _dealloc(), and Allocator::DataElement::allocator.

Referenced by DynamicInstruction::operator delete(), and DynamicMicroOp::operator delete().

The documentation for this class was generated from the following file:

- common/misc/ **allocator.h**

6.9 TraceManager::app_info_t Struct Reference

Public Member Functions

- **app_info_t** ()

Public Attributes

- **UInt32 thread_count**
- **UInt32 num_threads**
- **UInt32 num_runs**

6.9.1 Detailed Description

Definition at line 28 of file trace_manager.h.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 app_info_t()

```
TraceManager::app_info_t::app_info_t ( )  [inline]
```

Definition at line 30 of file trace_manager.h.

6.9.3 Member Data Documentation

6.9.3.1 num_runs

UInt32 TraceManager::app_info_t::num_runs

Definition at line 37 of file trace_manager.h.

6.9.3.2 num_threads

UInt32 TraceManager::app_info_t::num_threads

Definition at line 36 of file trace_manager.h.

6.9.3.3 thread_count

UInt32 TraceManager::app_info_t::thread_count

Definition at line 35 of file trace_manager.h.

The documentation for this struct was generated from the following file:

- common/trace_frontend/ **trace_manager.h**

6.10 ArithInstruction Class Reference

#include <instruction.h>

Inheritance diagram for ArithInstruction:

Public Member Functions

- **ArithInstruction** (**InstructionType** type, **OperandList** &operands)

Additional Inherited Members

6.10.1 Detailed Description

Definition at line 105 of file instruction.h.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 ArithInstruction()

```
ArithInstruction::ArithInstruction (
              InstructionType type,
              OperandList & operands )  [inline]
```

Definition at line 108 of file instruction.h.

The documentation for this class was generated from the following file:

- common/performance_model/ **instruction.h**

6.11 ATD Class Reference

`#include <cache_atd.h>`

Public Member Functions

- **ATD** (String name, String configName, **core_id_t** core_id, **UInt32** num_sets, **UInt32** associativity, **UInt32** cache_block_size, String replacement_policy, **CacheBase::hash_t** hash_function)
- ∼**ATD** ()
- void **access** (**Core::mem_op_t** mem_op_type, bool hit, **IntPtr** address)

Private Member Functions

- bool **isSampledSet** (**UInt32** set_index)

Private Attributes

- **CacheBase m_cache_base**
- std::unordered_map< **UInt32**, **CacheSet** ∗ > **m_sets**
- **CacheSetInfo** ∗ **m_set_info**
- **UInt64 loads**
- **UInt64 stores**
- **UInt64 load_misses**
- **UInt64 store_misses**
- **UInt64 loads_constructive**
- **UInt64 stores_constructive**
- **UInt64 loads_destructive**
- **UInt64 stores_destructive**

6.11.1 Detailed Description

Definition at line 13 of file cache_atd.h.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 ATD()

```
ATD::ATD (
             String name,
             String configName,
              core_id_t core_id,
              UInt32 num_sets,
              UInt32 associativity,
              UInt32 cache_block_size,
             String replacement_policy,
              CacheBase::hash_t hash_function )
```

Definition at line 8 of file cache_atd.cc.

References CacheSet::createCacheSet(), CacheSet::createCacheSetInfo(), load_misses, loads, loads_constructive, loads_destructive, LOG_ASSERT_ERROR, LOG_PRINT_ERROR, m_set_info, m_sets, CacheBase::PR_L1_CACHE, registerStatsMetric(), rng_next(), rng_seed(), store_misses, stores, stores_constructive, and stores_destructive.

6.11.2.2 ~ATD()

```
ATD::~ATD ( )
```

Definition at line 74 of file cache_atd.cc.

References m_set_info.

6.11.3 Member Function Documentation

6.11.3.1 access()

```
void ATD::access (
              Core::mem_op_t mem_op_type,
             bool hit,
              IntPtr address )
```

Definition at line 85 of file cache_atd.cc.

References isSampledSet(), load_misses, loads, loads_constructive, loads_destructive, m_cache_base, m_sets, CacheState::MODIFIED, CacheBase::splitAddress(), store_misses, stores, stores_constructive, stores_destructive, and Core::WRITE.

6.11.3.2 isSampledSet()

```
bool ATD::isSampledSet (
            UInt32 set_index )  [private]
```

Definition at line 80 of file cache_atd.cc.

References m_sets.

Referenced by access().

6.11.4 Member Data Documentation

6.11.4.1 load_misses

```
UInt64 ATD::load_misses  [private]
```

Definition at line 21 of file cache_atd.h.

Referenced by access(), and ATD().

6.11.4.2 loads

```
UInt64 ATD::loads  [private]
```

Definition at line 20 of file cache_atd.h.

Referenced by access(), and ATD().

6.11.4.3 loads_constructive

```
UInt64 ATD::loads_constructive  [private]
```

Definition at line 22 of file cache_atd.h.

Referenced by access(), and ATD().

6.11.4.4 loads_destructive

```
UInt64 ATD::loads_destructive [private]
```

Definition at line 23 of file cache_atd.h.

Referenced by access(), and ATD().

6.11.4.5 m_cache_base

```
CacheBase ATD::m_cache_base [private]
```

Definition at line 16 of file cache_atd.h.

Referenced by access().

6.11.4.6 m_set_info

```
CacheSetInfo* ATD::m_set_info [private]
```

Definition at line 18 of file cache_atd.h.

Referenced by ATD(), and ~ATD().

6.11.4.7 m_sets

```
std::unordered_map< UInt32, CacheSet*> ATD::m_sets [private]
```

Definition at line 17 of file cache_atd.h.

Referenced by access(), ATD(), and isSampledSet().

6.11.4.8 store_misses

```
UInt64 ATD::store_misses [private]
```

Definition at line 21 of file cache_atd.h.

Referenced by access(), and ATD().

6.11.4.9 stores

UInt64 ATD::stores [private]

Definition at line 20 of file cache_atd.h.

Referenced by access(), and ATD().

6.11.4.10 stores_constructive

UInt64 ATD::stores_constructive [private]

Definition at line 22 of file cache_atd.h.

Referenced by access(), and ATD().

6.11.4.11 stores_destructive

UInt64 ATD::stores_destructive [private]

Definition at line 23 of file cache_atd.h.

Referenced by access(), and ATD().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_atd.h**
- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_atd.cc**

6.12 Barrier Class Reference

`#include <barrier.h>`

Public Member Functions

- **Barrier** (int count)
- ~**Barrier** ()
- void **wait** ()

Private Attributes

- int **m_count**
- int **m_arrived**
- int **m_leaving**
- **Lock m_lock**
- **ConditionVariable m_cond**

6.12.1 Detailed Description

Definition at line 7 of file barrier.h.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 Barrier()

```
Barrier::Barrier (
            int count )
```

Definition at line 4 of file barrier.cc.

6.12.2.2 ~Barrier()

```
Barrier::~Barrier ( )
```

Definition at line 13 of file barrier.cc.

6.12.3 Member Function Documentation

6.12.3.1 wait()

```
void Barrier::wait ( )
```

Definition at line 17 of file barrier.cc.

References TLock< T_LockCreator >::acquire(), ConditionVariable::broadcast(), m_arrived, m_cond, m_count, m_leaving, m_lock, TLock< T_LockCreator >::release(), and ConditionVariable::wait().

6.12.4 Member Data Documentation

6.12.4.1 m_arrived

```
int Barrier::m_arrived  [private]
```

Definition at line 17 of file barrier.h.

Referenced by wait().

6.12.4.2 m_cond

ConditionVariable Barrier::m_cond [private]

Definition at line 20 of file barrier.h.

Referenced by wait().

6.12.4.3 m_count

int Barrier::m_count [private]

Definition at line 16 of file barrier.h.

Referenced by wait().

6.12.4.4 m_leaving

int Barrier::m_leaving [private]

Definition at line 18 of file barrier.h.

Referenced by wait().

6.12.4.5 m_lock

Lock Barrier::m_lock [private]

Definition at line 19 of file barrier.h.

Referenced by wait().

The documentation for this class was generated from the following files:

- common/misc/ **barrier.h**
- common/misc/ **barrier.cc**

6.13 BarrierSyncClient Class Reference

`#include <barrier_sync_client.h>`

Inheritance diagram for BarrierSyncClient:

Public Member Functions

- **BarrierSyncClient** (**Core** ∗core)
- ∼**BarrierSyncClient** ()
- void **enable** ()
- void **disable** ()
- void **synchronize** (**SubsecondTime** time, bool ignore_time, bool abort_func(void ∗)=NULL, void ∗abort_↩ arg=NULL)

Private Attributes

- **Core** ∗ **m_core**
- **SubsecondTime m_barrier_interval**
- **SubsecondTime m_next_sync_time**
- **UInt32 m_num_outstanding**

Additional Inherited Members

6.13.1 Detailed Description

Definition at line 12 of file barrier_sync_client.h.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 BarrierSyncClient()

```
BarrierSyncClient::BarrierSyncClient (
            Core * core )
```

Definition at line 10 of file barrier_sync_client.cc.

References LOG_PRINT_ERROR, m_barrier_interval, m_next_sync_time, and SubsecondTime::NS().

6.13.2.2 ~BarrierSyncClient()

```
BarrierSyncClient::~BarrierSyncClient ( )
```

Definition at line 25 of file barrier_sync_client.cc.

6.13.3 Member Function Documentation

6.13.3.1 disable()

```
void BarrierSyncClient::disable ( )  [inline], [virtual]
```

Implements **ClockSkewMinimizationClient** (p. 299).

Definition at line 27 of file barrier_sync_client.h.

6.13.3.2 enable()

```
void BarrierSyncClient::enable ( )  [inline], [virtual]
```

Implements **ClockSkewMinimizationClient** (p. 299).

Definition at line 26 of file barrier_sync_client.h.

6.13.3.3 synchronize()

```
void BarrierSyncClient::synchronize (
              SubsecondTime time,
             bool ignore_time,
             bool  abort_funcvoid * = NULL,
             void * abort_arg = NULL )  [virtual]
```

Implements **ClockSkewMinimizationClient** (p. 299).

Definition at line 29 of file barrier_sync_client.cc.

References PerformanceModel::getElapsedTime(), Core::getId(), Core::getPerformanceModel(), m_barrier_↩interval, m_core, m_next_sync_time, and SubsecondTime::Zero().

6.13.4 Member Data Documentation

6.13.4.1 m_barrier_interval

```
SubsecondTime BarrierSyncClient::m_barrier_interval [private]
```

Definition at line 17 of file barrier_sync_client.h.

Referenced by BarrierSyncClient(), and synchronize().

6.13.4.2 m_core

```
Core* BarrierSyncClient::m_core [private]
```

Definition at line 15 of file barrier_sync_client.h.

Referenced by synchronize().

6.13.4.3 m_next_sync_time

```
SubsecondTime BarrierSyncClient::m_next_sync_time [private]
```

Definition at line 18 of file barrier_sync_client.h.

Referenced by BarrierSyncClient(), and synchronize().

6.13.4.4 m_num_outstanding

```
UInt32 BarrierSyncClient::m_num_outstanding [private]
```

Definition at line 20 of file barrier_sync_client.h.

The documentation for this class was generated from the following files:

- common/system/ **barrier_sync_client.h**
- common/system/ **barrier_sync_client.cc**

6.14 BarrierSyncServer Class Reference

```
#include <barrier_sync_server.h>
```

Inheritance diagram for BarrierSyncServer:

Public Member Functions

- **BarrierSyncServer** ()
- ~**BarrierSyncServer** ()
- virtual void **setDisable** (bool disable)
- virtual void **setGroup** (**core_id_t** core_id, **core_id_t** master_core_id)
- void **synchronize** (**core_id_t** core_id, **SubsecondTime** time)
- void **release** ()
- void **advance** ()
- void **setFastForward** (bool fastforward, **SubsecondTime** next_barrier_time= **SubsecondTime::Max**↩ **Time**())
- **SubsecondTime** **getGlobalTime** (bool upper_bound=false)
- void **setBarrierInterval** (**SubsecondTime** barrier_interval)
- **SubsecondTime** **getBarrierInterval** () const
- void **printState** (void)

Private Member Functions

- bool **isBarrierReached** (void)
- bool **barrierRelease** (**thread_id_t** thread_id= **INVALID_THREAD_ID**, bool continue_until_release=false)
- void **abortBarrier** (void)
- bool **isCoreRunning** (**core_id_t** core_id, bool siblings=true)
- void **releaseThread** (**thread_id_t** thread_id)
- void **signal** ()
- void **doRelease** (int **n**)
- void **threadExit** (**HooksManager::ThreadTime** ∗argument)
- void **threadStall** (**HooksManager::ThreadStall** ∗argument)
- void **threadMigrate** (**HooksManager::ThreadMigrate** ∗argument)

Static Private Member Functions

- static **SInt64** **hookThreadExit** (**UInt64** object, **UInt64** argument)
- static **SInt64** **hookThreadStall** (**UInt64** object, **UInt64** argument)
- static **SInt64** **hookThreadMigrate** (**UInt64** object, **UInt64** argument)

Private Attributes

- **SubsecondTime** **m_barrier_interval**
- **SubsecondTime** **m_next_barrier_time**
- std::vector< **SubsecondTime** > **m_local_clock_list**
- std::vector< bool > **m_barrier_acquire_list**
- std::vector< **ConditionVariable** ∗ > **m_core_cond**
- std::vector< **core_id_t** > **m_to_release**
- std::vector< **core_id_t** > **m_core_group**
- std::vector< **thread_id_t** > **m_core_thread**
- **SubsecondTime** **m_global_time**
- bool **m_fastforward**
- volatile bool **m_disable**

Additional Inherited Members

6.14.1 Detailed Description

Definition at line 12 of file barrier_sync_server.h.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 BarrierSyncServer()

```
BarrierSyncServer::BarrierSyncServer ( )
```

Definition at line 18 of file barrier_sync_server.cc.

References HookType::HOOK_THREAD_EXIT, HookType::HOOK_THREAD_MIGRATE, HookType::HOOK_TH↩ READ_STALL, hookThreadExit(), hookThreadMigrate(), hookThreadStall(), LOG_PRINT_ERROR, m_barrier_↩ interval, m_core_cond, m_global_time, m_next_barrier_time, SubsecondTime::NS(), HooksManager::ORDER_↩ NOTIFY_POST, and registerStatsMetric().

6.14.2.2 ~BarrierSyncServer()

```
BarrierSyncServer::~BarrierSyncServer ( )
```

Definition at line 50 of file barrier_sync_server.cc.

References m_core_cond.

6.14.3 Member Function Documentation

6.14.3.1 abortBarrier()

```
void BarrierSyncServer::abortBarrier (
            void  )  [private]
```

Definition at line 338 of file barrier_sync_server.cc.

References PerformanceModel::barrierExit(), CLOG, Core::getPerformanceModel(), m_barrier_acquire_list, and m_core_cond.

Referenced by release(), and setDisable().

6.14.3.2 advance()

```
void BarrierSyncServer::advance ( )  [virtual]
```

Implements **ClockSkewMinimizationServer** (p. 304).

Definition at line 187 of file barrier_sync_server.cc.

References barrierRelease(), and INVALID_THREAD_ID.

6.14.3.3 barrierRelease()

```
bool BarrierSyncServer::barrierRelease (
              thread_id_t thread_id =  INVALID_THREAD_ID,
             bool continue_until_release = false )  [private]
```

Definition at line 240 of file barrier_sync_server.cc.

References PerformanceModel::barrierExit(), CLOG, doRelease(), SubsecondTime::getNS(), Core::getPerformanceModel(), HookType::HOOK_PERIODIC, itostr(), LOG_ASSERT_ERROR, LOG_PRINT, m_barrier_acquire_list, m_barrier_interval, m_core_thread, m_disable, m_fastforward, m_global_time, m_local_clock_list, m_next_barrier_time, m_to_release, and SubsecondTime::MaxTime().

Referenced by advance(), signal(), and synchronize().

6.14.3.4 doRelease()

```
void BarrierSyncServer::doRelease (
             int n )  [private]
```

Definition at line 325 of file barrier_sync_server.cc.

References m_core_cond, m_to_release, and n.

Referenced by barrierRelease(), releaseThread(), and synchronize().

6.14.3.5 getBarrierInterval()

```
SubsecondTime BarrierSyncServer::getBarrierInterval ( ) const  [inline], [virtual]
```

Implements **ClockSkewMinimizationServer** (p. 304).

Definition at line 60 of file barrier_sync_server.h.

References m_barrier_interval.

6.14.3.6 getGlobalTime()

```
SubsecondTime BarrierSyncServer::getGlobalTime (
            bool upper_bound = false )  [inline], [virtual]
```

Reimplemented from **ClockSkewMinimizationServer** (p. 304).

Definition at line 58 of file barrier_sync_server.h.

References m_global_time, and m_next_barrier_time.

6.14.3.7 hookThreadExit()

```
static  SInt64 BarrierSyncServer::hookThreadExit (
              UInt64 object,
              UInt64 argument )  [inline], [static], [private]
```

Definition at line 35 of file barrier_sync_server.h.

Referenced by BarrierSyncServer().

6.14.3.8 hookThreadMigrate()

```
static  SInt64 BarrierSyncServer::hookThreadMigrate (
              UInt64 object,
              UInt64 argument )  [inline], [static], [private]
```

Definition at line 41 of file barrier_sync_server.h.

Referenced by BarrierSyncServer().

6.14.3.9 hookThreadStall()

```
static  SInt64 BarrierSyncServer::hookThreadStall (
              UInt64 object,
              UInt64 argument )  [inline], [static], [private]
```

Definition at line 38 of file barrier_sync_server.h.

Referenced by BarrierSyncServer().

6.14.3.10 isBarrierReached()

```
bool BarrierSyncServer::isBarrierReached (
            void  )  [private]
```

Definition at line 193 of file barrier_sync_server.cc.

References INVALID_CORE_ID, isCoreRunning(), m_barrier_acquire_list, m_core_group, m_fastforward, m_↩ local_clock_list, and m_next_barrier_time.

Referenced by signal(), and synchronize().

6.14.3.11 isCoreRunning()

```
bool BarrierSyncServer::isCoreRunning (
             core_id_t core_id,
            bool siblings = true )  [private]
```

Definition at line 160 of file barrier_sync_server.cc.

References Thread::getId(), Core::getState(), Core::getThread(), LOG_ASSERT_ERROR, m_core_group, m_↩ fastforward, and Core::RUNNING.

Referenced by isBarrierReached(), and printState().

6.14.3.12 printState()

```
void BarrierSyncServer::printState (
            void  )  [virtual]
```

Reimplemented from **ClockSkewMinimizationServer** (p. 305).

Definition at line 385 of file barrier_sync_server.cc.

References INVALID_CORE_ID, isCoreRunning(), m_barrier_acquire_list, m_core_group, m_local_clock_list, and m_next_barrier_time.

6.14.3.13 release()

```
void BarrierSyncServer::release ( )  [inline], [virtual]
```

Implements **ClockSkewMinimizationServer** (p. 305).

Definition at line 55 of file barrier_sync_server.h.

References abortBarrier().

6.14.3.14 releaseThread()

```
void BarrierSyncServer::releaseThread (
              thread_id_t thread_id )  [private]
```

Definition at line 135 of file barrier_sync_server.cc.

References doRelease(), m_barrier_acquire_list, m_core_thread, m_local_clock_list, and SubsecondTime::Zero().

Referenced by threadExit(), threadMigrate(), and threadStall().

6.14.3.15 setBarrierInterval()

```
void BarrierSyncServer::setBarrierInterval (
              SubsecondTime barrier_interval )  [inline], [virtual]
```

Implements **ClockSkewMinimizationServer** (p. 305).

Definition at line 59 of file barrier_sync_server.h.

References m_barrier_interval.

6.14.3.16 setDisable()

```
void BarrierSyncServer::setDisable (
             bool disable )  [virtual]
```

Reimplemented from **ClockSkewMinimizationServer** (p. 305).

Definition at line 356 of file barrier_sync_server.cc.

References abortBarrier(), and m_disable.

6.14.3.17 setFastForward()

```
void BarrierSyncServer::setFastForward (
             bool fastforward,
              SubsecondTime next_barrier_time =  SubsecondTime::MaxTime() )  [virtual]
```

Implements **ClockSkewMinimizationServer** (p. 305).

Definition at line 373 of file barrier_sync_server.cc.

References CLOG, m_fastforward, m_next_barrier_time, and SubsecondTime::MaxTime().

6.14.3.18 setGroup()

```
void BarrierSyncServer::setGroup (
            core_id_t core_id,
            core_id_t master_core_id )  [virtual]
```

Implements **ClockSkewMinimizationServer** (p. 306).

Definition at line 364 of file barrier_sync_server.cc.

References INVALID_CORE_ID, LOG_ASSERT_ERROR, m_barrier_acquire_list, and m_core_group.

6.14.3.19 signal()

```
void BarrierSyncServer::signal ( )  [private]
```

Definition at line 150 of file barrier_sync_server.cc.

References barrierRelease(), INVALID_THREAD_ID, isBarrierReached(), and m_disable.

Referenced by threadExit(), and threadStall().

6.14.3.20 synchronize()

```
void BarrierSyncServer::synchronize (
            core_id_t core_id,
            SubsecondTime time )  [virtual]
```

Implements **ClockSkewMinimizationServer** (p. 306).

Definition at line 57 of file barrier_sync_server.cc.

References PerformanceModel::barrierEnter(), PerformanceModel::barrierExit(), barrierRelease(), CLOG, doRelease(), Thread::getId(), Core::getPerformanceModel(), Core::getState(), Core::getThread(), Core::INITIALIZING, INVALID_CORE_ID, isBarrierReached(), itostr(), LOG_ASSERT_ERROR, LOG_PRINT, m_barrier_acquire_list, m_core_cond, m_core_group, m_core_thread, m_disable, m_fastforward, m_local_clock_list, m_next_barrier_time, and Core::RUNNING.

6.14.3.21 threadExit()

```
void BarrierSyncServer::threadExit (
            HooksManager::ThreadTime * argument )  [private]
```

Definition at line 108 of file barrier_sync_server.cc.

References releaseThread(), signal(), and HooksManager::ThreadTime::thread_id.

6.14.3.22 threadMigrate()

```
void BarrierSyncServer::threadMigrate (
            HooksManager::ThreadMigrate * argument )  [private]
```

Definition at line 126 of file barrier_sync_server.cc.

References releaseThread(), and HooksManager::ThreadMigrate::thread_id.

6.14.3.23 threadStall()

```
void BarrierSyncServer::threadStall (
            HooksManager::ThreadStall * argument )  [private]
```

Definition at line 117 of file barrier_sync_server.cc.

References releaseThread(), signal(), and HooksManager::ThreadStall::thread_id.

6.14.4 Member Data Documentation

6.14.4.1 m_barrier_acquire_list

```
std::vector<bool> BarrierSyncServer::m_barrier_acquire_list  [private]
```

Definition at line 18 of file barrier_sync_server.h.

Referenced by abortBarrier(), barrierRelease(), isBarrierReached(), printState(), releaseThread(), setGroup(), and synchronize().

6.14.4.2 m_barrier_interval

```
SubsecondTime BarrierSyncServer::m_barrier_interval  [private]
```

Definition at line 15 of file barrier_sync_server.h.

Referenced by barrierRelease(), BarrierSyncServer(), getBarrierInterval(), and setBarrierInterval().

6.14.4.3 m_core_cond

```
std::vector< ConditionVariable*> BarrierSyncServer::m_core_cond  [private]
```

Definition at line 19 of file barrier_sync_server.h.

Referenced by abortBarrier(), BarrierSyncServer(), doRelease(), synchronize(), and ~BarrierSyncServer().

6.14.4.4 m_core_group

```
std::vector< core_id_t> BarrierSyncServer::m_core_group  [private]
```

Definition at line 21 of file barrier_sync_server.h.

Referenced by isBarrierReached(), isCoreRunning(), printState(), setGroup(), and synchronize().

6.14.4.5 m_core_thread

```
std::vector< thread_id_t> BarrierSyncServer::m_core_thread  [private]
```

Definition at line 22 of file barrier_sync_server.h.

Referenced by barrierRelease(), releaseThread(), and synchronize().

6.14.4.6 m_disable

```
volatile bool BarrierSyncServer::m_disable  [private]
```

Definition at line 25 of file barrier_sync_server.h.

Referenced by barrierRelease(), setDisable(), signal(), and synchronize().

6.14.4.7 m_fastforward

```
bool BarrierSyncServer::m_fastforward  [private]
```

Definition at line 24 of file barrier_sync_server.h.

Referenced by barrierRelease(), isBarrierReached(), isCoreRunning(), setFastForward(), and synchronize().

6.14.4.8 m_global_time

SubsecondTime BarrierSyncServer::m_global_time [private]

Definition at line 23 of file barrier_sync_server.h.

Referenced by barrierRelease(), BarrierSyncServer(), and getGlobalTime().

6.14.4.9 m_local_clock_list

std::vector< **SubsecondTime**> BarrierSyncServer::m_local_clock_list [private]

Definition at line 17 of file barrier_sync_server.h.

Referenced by barrierRelease(), isBarrierReached(), printState(), releaseThread(), and synchronize().

6.14.4.10 m_next_barrier_time

SubsecondTime BarrierSyncServer::m_next_barrier_time [private]

Definition at line 16 of file barrier_sync_server.h.

Referenced by barrierRelease(), BarrierSyncServer(), getGlobalTime(), isBarrierReached(), printState(), setFast↩Forward(), and synchronize().

6.14.4.11 m_to_release

std::vector< **core_id_t**> BarrierSyncServer::m_to_release [private]

Definition at line 20 of file barrier_sync_server.h.

Referenced by barrierRelease(), and doRelease().

The documentation for this class was generated from the following files:

- common/system/ **barrier_sync_server.h**
- common/system/ **barrier_sync_server.cc**

6.15 BaseCoreModel< T > Class Template Reference

`#include <core_model.h>`

Inheritance diagram for BaseCoreModel< T >:

Public Member Functions

- virtual **Allocator** * **createDMOAllocator** () const
- **DynamicMicroOp** * **createDynamicMicroOp** (**Allocator** *alloc, const **MicroOp** *uop, **Component↩ Period** period) const

Additional Inherited Members

6.15.1 Detailed Description

template<typename T>
class BaseCoreModel< T >

Definition at line 42 of file core_model.h.

6.15.2 Member Function Documentation

6.15.2.1 createDMOAllocator()

```
template<typename T >
virtual Allocator* BaseCoreModel< T >::createDMOAllocator ( ) const [inline], [virtual]
```

Implements **CoreModel** (p. 404).

Definition at line 45 of file core_model.h.

6.15.2.2 createDynamicMicroOp()

```
template<typename T >
DynamicMicroOp*  BaseCoreModel< T >::createDynamicMicroOp (
              Allocator * alloc,
             const  MicroOp * uop,
              ComponentPeriod period ) const  [inline], [virtual]
```

Implements **CoreModel** (p. 404).

Definition at line 51 of file core_model.h.

The documentation for this class was generated from the following file:

- common/performance_model/performance_models/core_model/ **core_model.h**

6.16 BaseLock Class Reference

```
#include <lock.h>
```

Inheritance diagram for BaseLock:

Public Member Functions

- virtual void **acquire** ()=0
- virtual void **release** ()=0
- virtual void **acquire_read** ()=0
- virtual void **release_read** ()=0

6.16.1 Detailed Description

Definition at line 36 of file lock.h.

6.16.2 Member Function Documentation

6.16.2.1 acquire()

```
virtual void BaseLock::acquire ( )  [pure virtual]
```

Implemented in **TLock**< **T_LockCreator** > (p. 1418), **TLock**< **LockCreator_RwLock** > (p. 1418), and **TLock**< **LockCreator_Default** > (p. 1418).

Referenced by ScopedLock::ScopedLock().

6.16.2.2 acquire_read()

```
virtual void BaseLock::acquire_read ( )  [pure virtual]
```

Implemented in **TLock**< **T_LockCreator** > (p. 1418), **TLock**< **LockCreator_RwLock** > (p. 1418), and **TLock**< **LockCreator_Default** > (p. 1418).

Referenced by ScopedReadLock::ScopedReadLock().

6.16.2.3 release()

```
virtual void BaseLock::release ( )  [pure virtual]
```

Implemented in **TLock**< **T_LockCreator** > (p. 1418), **TLock**< **LockCreator_RwLock** > (p. 1418), and **TLock**< **LockCreator_Default** > (p. 1418).

Referenced by ScopedLock::~ScopedLock().

6.16.2.4 release_read()

```
virtual void BaseLock::release_read ( )  [pure virtual]
```

Implemented in **TLock**< **T_LockCreator** > (p. 1419), **TLock**< **LockCreator_RwLock** > (p. 1419), and **TLock**< **LockCreator_Default** > (p. 1419).

Referenced by ScopedReadLock::~ScopedReadLock().

The documentation for this class was generated from the following file:

- common/misc/ **lock.h**

6.17 BasicHash Class Reference

`#include <basic_hash.h>`

Inheritance diagram for BasicHash:

Public Member Functions

- **BasicHash** (**UInt64** **size**)
- ~**BasicHash** ()
- std::pair< bool, **UInt64** > **find** (**UInt64** key)
- bool **insert** (**UInt64** key, **UInt64** value)

Protected Types

- typedef std::unordered_map< **UInt64**, **UInt64** > **Bucket**

Protected Attributes

- **Bucket** * **array**
- **UInt64** **size**

6.17.1 Detailed Description

Definition at line 14 of file basic_hash.h.

6.17.2 Member Typedef Documentation

6.17.2.1 Bucket

`typedef std::unordered_map< UInt64, UInt64> BasicHash::Bucket [protected]`

Definition at line 17 of file basic_hash.h.

6.17.3 Constructor & Destructor Documentation

6.17.3.1 BasicHash()

```
BasicHash::BasicHash (
            UInt64 size )
```

Definition at line 3 of file basic_hash.cc.

6.17.3.2 ~BasicHash()

```
BasicHash::~BasicHash ( )
```

Definition at line 7 of file basic_hash.cc.

References array.

6.17.4 Member Function Documentation

6.17.4.1 find()

```
std::pair< bool, UInt64 > BasicHash::find (
            UInt64 key )
```

Definition at line 13 of file basic_hash.cc.

References array, and size.

Referenced by LockFreeHash::find().

6.17.4.2 insert()

```
bool BasicHash::insert (
            UInt64 key,
            UInt64 value )
```

Definition at line 27 of file basic_hash.cc.

References array, and size.

Referenced by LockFreeHash::insert().

6.17.5 Member Data Documentation

6.17.5.1 array

```
Bucket* BasicHash::array  [protected]
```

Definition at line 18 of file basic_hash.h.

Referenced by LockFreeHash::bucket_size(), find(), insert(), and ~BasicHash().

6.17.5.2 size

```
UInt64 BasicHash::size  [protected]
```

Definition at line 19 of file basic_hash.h.

Referenced by LockFreeHash::bucket_size(), find(), and insert().

The documentation for this class was generated from the following files:

- common/misc/ **basic_hash.h**
- common/misc/ **basic_hash.cc**

6.18 BbvCount Class Reference

```
#include <bbv_count.h>
```

Public Member Functions

- **BbvCount** (**core_id_t** core_id)
- ~**BbvCount** ()
- bool **sample** ()
- void **count** (**UInt64** address, **UInt64** count)
- void **reset** (bool save=true)
- **UInt64** **getDiff** ()
- **UInt64** **getDimension** (int dim)
- **UInt64** **getInstructionCount** (void) const

Static Public Attributes

- static const int **NUM_BBV** = 16

Private Member Functions

- void **sampleReset** ()

Private Attributes

- const **core_id_t m_core_id**
- **UInt64 m_instrs_abs**
- std::vector< **UInt64** > **m_bbv_counts_abs**
- **UInt64 m_instrs_reset**
- std::vector< **UInt64** > **m_bbv_reset**
- std::vector< **UInt64** > **m_bbv_previous**
- **UInt64 m_sample_period**
- **UInt64 m_sample_seed**
- **UInt64 m_sample**

6.18.1 Detailed Description

Definition at line 8 of file bbv_count.h.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 BbvCount()

```
BbvCount::BbvCount (
              core_id_t core_id )
```

Definition at line 7 of file bbv_count.cc.

References itostr(), m_bbv_counts_abs, NUM_BBV, registerStatsMetric(), and sampleReset().

6.18.2.2 ~BbvCount()

```
BbvCount::~BbvCount ( )
```

Definition at line 23 of file bbv_count.cc.

References NUM_BBV.

6.18.3 Member Function Documentation

6.18.3.1 count()

```
void BbvCount::count (
              UInt64 address,
              UInt64 count )
```

Definition at line 50 of file bbv_count.cc.

References m_bbv_counts_abs, m_instrs_abs, NUM_BBV, rng_next(), rng_seed(), and sampleReset().

Referenced by Core::countInstructions().

6.18.3.2 getDiff()

```
UInt64 BbvCount::getDiff ( )
```

Definition at line 91 of file bbv_count.cc.

References getDimension(), getInstructionCount(), m_bbv_previous, and NUM_BBV.

6.18.3.3 getDimension()

```
UInt64 BbvCount::getDimension (
            int dim )  [inline]
```

Definition at line 42 of file bbv_count.h.

References m_bbv_counts_abs, and m_bbv_reset.

Referenced by getBbv(), getDiff(), and reset().

6.18.3.4 getInstructionCount()

```
UInt64 BbvCount::getInstructionCount (
            void  ) const  [inline]
```

Definition at line 43 of file bbv_count.h.

References m_instrs_abs, and m_instrs_reset.

Referenced by getBbv(), getDiff(), and reset().

6.18.3.5 reset()

```
void BbvCount::reset (
            bool save = true )
```

Definition at line 76 of file bbv_count.cc.

References getDimension(), getInstructionCount(), m_bbv_counts_abs, m_bbv_previous, m_bbv_reset, m_instrs↩_abs, m_instrs_reset, and NUM_BBV.

6.18.3.6 sample()

```
bool BbvCount::sample ( )
```

Definition at line 44 of file bbv_count.cc.

References m_sample.

Referenced by Core::countInstructions().

6.18.3.7 sampleReset()

```
void BbvCount::sampleReset ( )  [private]
```

Definition at line 34 of file bbv_count.cc.

References m_sample, m_sample_period, m_sample_seed, and rng_next().

Referenced by BbvCount(), and count().

6.18.4 Member Data Documentation

6.18.4.1 m_bbv_counts_abs

```
std::vector< UInt64> BbvCount::m_bbv_counts_abs  [private]
```

Definition at line 15 of file bbv_count.h.

Referenced by BbvCount(), count(), getDimension(), and reset().

6.18.4.2 m_bbv_previous

```
std::vector< UInt64> BbvCount::m_bbv_previous  [private]
```

Definition at line 22 of file bbv_count.h.

Referenced by getDiff(), and reset().

6.18.4.3 m_bbv_reset

```
std::vector< UInt64> BbvCount::m_bbv_reset  [private]
```

Definition at line 19 of file bbv_count.h.

Referenced by getDimension(), and reset().

6.18.4.4 m_core_id

```
const  core_id_t BbvCount::m_core_id  [private]
```

Definition at line 11 of file bbv_count.h.

6.18.4.5 m_instrs_abs

```
 UInt64 BbvCount::m_instrs_abs  [private]
```

Definition at line 14 of file bbv_count.h.

Referenced by count(), getInstructionCount(), and reset().

6.18.4.6 m_instrs_reset

```
 UInt64 BbvCount::m_instrs_reset  [private]
```

Definition at line 18 of file bbv_count.h.

Referenced by getInstructionCount(), and reset().

6.18.4.7 m_sample

UInt64 BbvCount::m_sample [private]

Definition at line 26 of file bbv_count.h.

Referenced by sample(), and sampleReset().

6.18.4.8 m_sample_period

UInt64 BbvCount::m_sample_period [private]

Definition at line 24 of file bbv_count.h.

Referenced by sampleReset().

6.18.4.9 m_sample_seed

UInt64 BbvCount::m_sample_seed [private]

Definition at line 25 of file bbv_count.h.

Referenced by sampleReset().

6.18.4.10 NUM_BBV

const int BbvCount::NUM_BBV = 16 [static]

Definition at line 33 of file bbv_count.h.

Referenced by BbvCount(), count(), getBbv(), getDiff(), reset(), HooksPy::PyBbv::setup(), and ~BbvCount().

The documentation for this class was generated from the following files:

- common/core/ **bbv_count.h**
- common/core/ **bbv_count.cc**

6.19 BitVector Class Reference

#include <bit_vector.h>

Public Member Functions

- **BitVector** (**UInt32** bits)
- ∼**BitVector** ()
- **SInt32 find** ()
- bool **resetFind** ()
- bool **bTestBit** (**UInt8** word, **UInt32** bit)
- **UInt32 capacity** ()
- **UInt32 size** ()
- void **reset** ()
- bool **at** (**UInt32** bit)
- void **set** (**UInt32** bit)
- void **clear** (**UInt32** bit)
- void **set** (const **BitVector** &vec2)
- void **clear** (const **BitVector** &vec2)
- bool **test** (const **BitVector** &vec2)

Private Attributes

- **UInt32 m_capacity**
- **UInt32 m_size**
- **SInt32 m_last_pos**
- const **UInt32 VECTOR_SIZE**
- std::vector< **UInt64** > **m_words**

6.19.1 Detailed Description

Definition at line 17 of file bit_vector.h.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 BitVector()

```
BitVector::BitVector (
              UInt32 bits )  [inline]
```

Definition at line 35 of file bit_vector.h.

References m_words, and VECTOR_SIZE.

6.19.2.2 ∼BitVector()

```
BitVector::~BitVector ( )  [inline]
```

Definition at line 43 of file bit_vector.h.

6.19.3 Member Function Documentation

6.19.3.1 at()

```
bool BitVector::at (
              UInt32 bit )
```

Definition at line 122 of file bit_vector.cc.

References m_capacity, and m_words.

Referenced by clear(), and set().

6.19.3.2 bTestBit()

```
bool BitVector::bTestBit (
              UInt8 word,
              UInt32 bit )
```

Definition at line 112 of file bit_vector.cc.

Referenced by find().

6.19.3.3 capacity()

```
UInt32 BitVector::capacity ( )  [inline]
```

Definition at line 58 of file bit_vector.h.

References m_capacity.

6.19.3.4 clear() [1/2]

```
void BitVector::clear (
             const  BitVector & vec2 )
```

6.19.3.5 clear() [2/2]

```
void BitVector::clear (
              UInt32 bit )
```

Definition at line 148 of file bit_vector.cc.

References at(), m_capacity, m_size, and m_words.

6.19.3.6 find()

```
 SInt32 BitVector::find ( )
```

Definition at line 39 of file bit_vector.cc.

References bTestBit(), m_capacity, m_last_pos, and m_words.

6.19.3.7 reset()

```
void BitVector::reset ( )
```

Definition at line 8 of file bit_vector.cc.

References m_last_pos, m_size, m_words, and VECTOR_SIZE.

6.19.3.8 resetFind()

```
bool BitVector::resetFind ( )
```

Definition at line 21 of file bit_vector.cc.

References m_last_pos.

6.19.3.9 set() [1/2]

```
void BitVector::set (
             const  BitVector & vec2 )
```

6.19.3.10 set() [2/2]

```
void BitVector::set (
            UInt32 bit )
```

Definition at line 133 of file bit_vector.cc.

References at(), m_capacity, m_size, and m_words.

6.19.3.11 size()

```
UInt32 BitVector::size ( )  [inline]
```

Definition at line 59 of file bit_vector.h.

References m_size.

6.19.3.12 test()

```
bool BitVector::test (
            const  BitVector & vec2 )
```

6.19.4 Member Data Documentation

6.19.4.1 m_capacity

```
UInt32 BitVector::m_capacity  [private]
```

Definition at line 20 of file bit_vector.h.

Referenced by at(), capacity(), clear(), find(), and set().

6.19.4.2 m_last_pos

```
SInt32 BitVector::m_last_pos  [private]
```

Definition at line 22 of file bit_vector.h.

Referenced by find(), reset(), and resetFind().

6.19.4.3 m_size

```
UInt32 BitVector::m_size  [private]
```

Definition at line 21 of file bit_vector.h.

Referenced by clear(), reset(), set(), and size().

6.19.4.4 m_words

```
std::vector< UInt64> BitVector::m_words  [private]
```

Definition at line 24 of file bit_vector.h.

Referenced by at(), BitVector(), clear(), find(), reset(), and set().

6.19.4.5 VECTOR_SIZE

```
const  UInt32 BitVector::VECTOR_SIZE  [private]
```

Definition at line 23 of file bit_vector.h.

Referenced by BitVector(), and reset().

The documentation for this class was generated from the following files:

- common/misc/ **bit_vector.h**
- common/misc/ **bit_vector.cc**

6.20 FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlocksVector Struct Reference

Public Member Functions

- **BlocksVector** ()
- ~**BlocksVector** ()

Public Attributes

- std::vector< **MemBlock** > **data**

6.20.1 Detailed Description

template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
struct FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlocksVector

Definition at line 107 of file FSBAllocator.hh.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 BlocksVector()

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlocksVector::BlocksVector ( )
[inline]
```

Definition at line 111 of file FSBAllocator.hh.

References FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlocksVector::data.

6.20.2.2 ~BlocksVector()

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlocksVector::~BlocksVector ( )
[inline]
```

Definition at line 113 of file FSBAllocator.hh.

References FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlocksVector::data.

6.20.3 Member Data Documentation

6.20.3.1 data

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
std::vector< MemBlock>  FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::Blocks↩
Vector::data
```

Definition at line 109 of file FSBAllocator.hh.

Referenced by FSBAllocator_ElemAllocator< sizeof(DataElement)+sizeof(T), 0, T >::allocate(), FSBAllocator_↩ ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlocksVector::BlocksVector(), FSBAllocator_ElemAllocator< sizeof(DataElement)+sizeof(T), 0, T >::deallocate(), and FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlocksVector::~BlocksVector().

The documentation for this struct was generated from the following file:

- common/misc/ **FSBAllocator.hh**

6.21 BottleGraphManager Class Reference

`#include <bottlegraph.h>`

Public Member Functions

- **BottleGraphManager** (int max_threads)
- void **threadStart** (**thread_id_t** thread_id)
- void **update** (**SubsecondTime** time, **thread_id_t** thread_id, bool running)

Private Attributes

- **SubsecondTime m_time_last**
- std::vector< bool > **m_running**
- std::vector< **SubsecondTime** > **m_contrib**
- std::vector< **SubsecondTime** > **m_runtime**

6.21.1 Detailed Description

Definition at line 9 of file bottlegraph.h.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 BottleGraphManager()

```
BottleGraphManager::BottleGraphManager (
            int max_threads )
```

Definition at line 7 of file bottlegraph.cc.

6.21.3 Member Function Documentation

6.21.3.1 threadStart()

```
void BottleGraphManager::threadStart (
             thread_id_t thread_id )
```

Definition at line 14 of file bottlegraph.cc.

References m_contrib, m_runtime, registerStatsMetric(), and SubsecondTime::Zero().

Referenced by ThreadStatsManager::threadStart().

6.21.3.2 update()

```
void BottleGraphManager::update (
              SubsecondTime time,
              thread_id_t thread_id,
             bool running )
```

Definition at line 22 of file bottlegraph.cc.

References INVALID_THREAD_ID, m_contrib, m_running, m_runtime, and m_time_last.

Referenced by ThreadStatsManager::pre_stat_write(), ThreadStatsManager::threadExit(), ThreadStatsManager::threadResume(), ThreadStatsManager::threadStall(), and ThreadStatsManager::threadStart().

6.21.4 Member Data Documentation

6.21.4.1 m_contrib

```
std::vector< SubsecondTime> BottleGraphManager::m_contrib  [private]
```

Definition at line 20 of file bottlegraph.h.

Referenced by threadStart(), and update().

6.21.4.2 m_running

```
std::vector<bool> BottleGraphManager::m_running  [private]
```

Definition at line 19 of file bottlegraph.h.

Referenced by update().

6.21.4.3 m_runtime

```
std::vector< SubsecondTime> BottleGraphManager::m_runtime  [private]
```

Definition at line 21 of file bottlegraph.h.

Referenced by threadStart(), and update().

6.21.4.4 m_time_last

```
SubsecondTime BottleGraphManager::m_time_last  [private]
```

Definition at line 18 of file bottlegraph.h.

Referenced by update().

The documentation for this class was generated from the following files:

- common/misc/ **bottlegraph.h**
- common/misc/ **bottlegraph.cc**

6.22 DynamicInstruction::BranchInfo Struct Reference

```
#include <dynamic_instruction.h>
```

Public Attributes

- bool **is_branch**
- bool **taken**
- **IntPtr target**

6.22.1 Detailed Description

Definition at line 24 of file dynamic_instruction.h.

6.22.2 Member Data Documentation

6.22.2.1 is_branch

```
bool DynamicInstruction::BranchInfo::is_branch
```

Definition at line 26 of file dynamic_instruction.h.

Referenced by DynamicInstruction::addBranch(), DynamicInstruction::DynamicInstruction(), and DynamicInstruction::isBranch().

6.22.2.2 taken

```
bool DynamicInstruction::BranchInfo::taken
```

Definition at line 27 of file dynamic_instruction.h.

Referenced by DynamicInstruction::addBranch(), DynamicInstruction::getBranchCost(), and MicroOpPerformance↩ Model::handleInstruction().

6.22.2.3 target

```
IntPtr DynamicInstruction::BranchInfo::target
```

Definition at line 28 of file dynamic_instruction.h.

Referenced by DynamicInstruction::addBranch(), DynamicInstruction::getBranchCost(), and MicroOpPerformance↩ Model::handleInstruction().

The documentation for this struct was generated from the following file:

- common/performance_model/ **dynamic_instruction.h**

6.23 BranchInstruction Class Reference

```
#include <instruction.h>
```

Inheritance diagram for BranchInstruction:

Public Member Functions

- **BranchInstruction** (**OperandList** &l)

Additional Inherited Members

6.23.1 Detailed Description

Definition at line 184 of file instruction.h.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 BranchInstruction()

```
BranchInstruction::BranchInstruction (
              OperandList & l )
```

Definition at line 112 of file instruction.cc.

The documentation for this class was generated from the following files:

- common/performance_model/ **instruction.h**
- common/performance_model/ **instruction.cc**

6.24 BranchPredictor Class Reference

```
#include <branch_predictor.h>
```

Inheritance diagram for BranchPredictor:

Public Member Functions

- **BranchPredictor** ()
- **BranchPredictor** (String name, **core_id_t** core_id)
- virtual ~**BranchPredictor** ()
- virtual bool **predict** (**IntPtr** ip, **IntPtr** target)=0
- virtual void **update** (bool predicted, bool actual, **IntPtr** ip, **IntPtr** target)=0
- **UInt64 getMispredictPenalty** ()
- **UInt64 getNumCorrectPredictions** ()
- **UInt64 getNumIncorrectPredictions** ()
- void **resetCounters** ()

Static Public Member Functions

- static **BranchPredictor** * **create** (**core_id_t** core_id)

Protected Member Functions

- void **updateCounters** (bool predicted, bool actual)

Private Attributes

- **UInt64 m_correct_predictions**
- **UInt64 m_incorrect_predictions**

Static Private Attributes

- static **UInt64 m_mispredict_penalty**

6.24.1 Detailed Description

Definition at line 8 of file branch_predictor.h.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 BranchPredictor() [1/2]

```
BranchPredictor::BranchPredictor ( )
```

Definition at line 8 of file branch_predictor.cc.

6.24.2.2 BranchPredictor() [2/2]

```
BranchPredictor::BranchPredictor (
             String name,
              core_id_t core_id )
```

Definition at line 14 of file branch_predictor.cc.

References m_correct_predictions, m_incorrect_predictions, and registerStatsMetric().

6.24.2.3 ~BranchPredictor()

```
BranchPredictor::~BranchPredictor ( )  [virtual]
```

Definition at line 22 of file branch_predictor.cc.

6.24.3 Member Function Documentation

6.24.3.1 create()

```
BranchPredictor * BranchPredictor::create (
            core_id_t core_id ) [static]
```

Definition at line 27 of file branch_predictor.cc.

References cfg, config::Config::getIntArray(), config::Config::getStringArray(), LOG_PRINT_ERROR, and m_↩ mispredict_penalty.

Referenced by PerformanceModel::PerformanceModel().

6.24.3.2 getMispredictPenalty()

```
UInt64 BranchPredictor::getMispredictPenalty ( )
```

Definition at line 63 of file branch_predictor.cc.

References m_mispredict_penalty.

Referenced by DynamicInstruction::getBranchCost().

6.24.3.3 getNumCorrectPredictions()

```
UInt64 BranchPredictor::getNumCorrectPredictions ( ) [inline]
```

Definition at line 21 of file branch_predictor.h.

References m_correct_predictions.

6.24.3.4 getNumIncorrectPredictions()

```
UInt64 BranchPredictor::getNumIncorrectPredictions ( ) [inline]
```

Definition at line 22 of file branch_predictor.h.

References m_incorrect_predictions.

Referenced by TraceThread::handleEmuFunc().

6.24.3.5 predict()

```
virtual bool BranchPredictor::predict (
              IntPtr ip,
              IntPtr target )  [pure virtual]
```

Implemented in **PentiumMBranchTargetBuffer** (p. 900), **IndirectBranchTargetBuffer** (p. 614), **SimpleBimodalTable** (p. 1217), **GlobalPredictor** (p. 590), **PentiumMBranchPredictor** (p. 896), **OneBitBranchPredictor** (p. 884), and **BranchTargetBuffer** (p. 131).

Referenced by Core::accessBranchPredictor().

6.24.3.6 resetCounters()

```
void BranchPredictor::resetCounters ( )
```

Definition at line 68 of file branch_predictor.cc.

References m_correct_predictions, and m_incorrect_predictions.

6.24.3.7 update()

```
virtual void BranchPredictor::update (
             bool predicted,
             bool actual,
              IntPtr ip,
              IntPtr target )  [pure virtual]
```

Implemented in **PentiumMBranchTargetBuffer** (p. 900), **IndirectBranchTargetBuffer** (p. 614), **SimpleBimodalTable** (p. 1217), **GlobalPredictor** (p. 590), **BranchTargetBuffer** (p. 131), **PentiumMBranchPredictor** (p. 896), and **OneBitBranchPredictor** (p. 885).

Referenced by Core::accessBranchPredictor().

6.24.3.8 updateCounters()

```
void BranchPredictor::updateCounters (
             bool predicted,
             bool actual )  [protected]
```

Definition at line 74 of file branch_predictor.cc.

References m_correct_predictions, and m_incorrect_predictions.

Referenced by OneBitBranchPredictor::update(), and PentiumMBranchPredictor::update().

6.24.4 Member Data Documentation

6.24.4.1 m_correct_predictions

UInt64 BranchPredictor::m_correct_predictions [private]

Definition at line 30 of file branch_predictor.h.

Referenced by BranchPredictor(), getNumCorrectPredictions(), resetCounters(), and updateCounters().

6.24.4.2 m_incorrect_predictions

UInt64 BranchPredictor::m_incorrect_predictions [private]

Definition at line 31 of file branch_predictor.h.

Referenced by BranchPredictor(), getNumIncorrectPredictions(), resetCounters(), and updateCounters().

6.24.4.3 m_mispredict_penalty

UInt64 BranchPredictor::m_mispredict_penalty [static], [private]

Definition at line 33 of file branch_predictor.h.

Referenced by create(), and getMispredictPenalty().

The documentation for this class was generated from the following files:

- common/performance_model/ **branch_predictor.h**
- common/performance_model/ **branch_predictor.cc**

6.25 BranchPredictorReturnValue Class Reference

`#include <branch_predictor_return_value.h>`

Public Types

- enum **BranchType** {
  InvalidBranch = 0, **DirectBranch**, **IndirectBranch**, **UnconditionalBranch**,
  ConditionalBranch }

Public Attributes

- bool **prediction**
- bool **hit**
- **IntPtr target**
- **BranchType type**

Static Public Attributes

- static const char * **BranchTypeNames** [ ]

Friends

- std::ostream & **operator**<< (std::ostream &stream, const **BranchPredictorReturnValue** &value)

6.25.1 Detailed Description

Definition at line 10 of file branch_predictor_return_value.h.

6.25.2 Member Enumeration Documentation

6.25.2.1 BranchType

```
enum BranchPredictorReturnValue::BranchType
```

Enumerator

InvalidBranch	
DirectBranch	
IndirectBranch	
UnconditionalBranch	
ConditionalBranch	

Definition at line 14 of file branch_predictor_return_value.h.

6.25.3 Friends And Related Function Documentation

6.25.3.1 operator<<

```
std::ostream& operator<< (
            std::ostream & stream,
            const  BranchPredictorReturnValue & value )  [friend]
```

Definition at line 34 of file branch_predictor_return_value.h.

6.25.4 Member Data Documentation

6.25.4.1 BranchTypeNames

```
const char * BranchPredictorReturnValue::BranchTypeNames  [static]
```

Initial value:

```
=
   {
      "InvalidBranch",
      "DirectBranch",
      "IndirectBranch",
      "UnconditionalBranch",
      "ConditionalBranch"
   }
```

Definition at line 23 of file branch_predictor_return_value.h.

Referenced by operator<<().

6.25.4.2 hit

```
bool BranchPredictorReturnValue::hit
```

Definition at line 26 of file branch_predictor_return_value.h.

Referenced by LoopBranchPredictor::lookup(), GlobalPredictor::lookup(), operator<<(), and PentiumMBranch↩ Predictor::predict().

6.25.4.3 prediction

```
bool BranchPredictorReturnValue::prediction
```

Definition at line 25 of file branch_predictor_return_value.h.

Referenced by LoopBranchPredictor::lookup(), GlobalPredictor::lookup(), operator<<(), and PentiumMBranch↩ Predictor::predict().

6.25.4.4 target

```
IntPtr BranchPredictorReturnValue::target
```

Definition at line 27 of file branch_predictor_return_value.h.

Referenced by operator<<().

6.25.4.5 type

```
BranchType BranchPredictorReturnValue::type
```

Definition at line 28 of file branch_predictor_return_value.h.

Referenced by operator<<().

The documentation for this class was generated from the following files:

- common/performance_model/branch_predictors/ **branch_predictor_return_value.h**
- common/performance_model/branch_predictors/ **branch_predictor_return_value.cc**

6.26 BranchTargetBuffer Class Reference

```
#include <btb.h>
```

Inheritance diagram for BranchTargetBuffer:

Private Member Functions

- bool **predict** (**IntPtr** ip, **IntPtr** target)
- **BranchPredictorReturnValue** **lookup** (**IntPtr** ip, **IntPtr** target)
- void **update** (bool predicted, bool actual, **IntPtr** ip, **IntPtr** target)

Additional Inherited Members

6.26.1 Detailed Description

Definition at line 6 of file btb.h.

6.26.2 Member Function Documentation

6.26.2.1 lookup()

```
BranchPredictorReturnValue BranchTargetBuffer::lookup (
            IntPtr ip,
            IntPtr target )  [inline], [private]
```

Definition at line 14 of file btb.h.

References BranchPredictorReturnValue::InvalidBranch.

6.26.2.2 predict()

```
bool BranchTargetBuffer::predict (
            IntPtr ip,
            IntPtr target )  [inline], [private], [virtual]
```

Implements **BranchPredictor** (p. 125).

Definition at line 9 of file btb.h.

6.26.2.3 update()

```
void BranchTargetBuffer::update (
            bool predicted,
            bool actual,
            IntPtr ip,
            IntPtr target )  [inline], [private], [virtual]
```

Implements **BranchPredictor** (p. 126).

Definition at line 23 of file btb.h.

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **btb.h**

6.27 Cache Class Reference

```
#include <cache.h>
```

Inheritance diagram for Cache:

Public Member Functions

- **Cache** (String name, String cfgname, **core_id_t** core_id, **UInt32** num_sets, **UInt32** associativity, **UInt32** cache_block_size, String replacement_policy, **cache_t** cache_type, **hash_t** hash= **CacheBase::HASH_↩ MASK**, **FaultInjector** ∗fault_injector=NULL, **AddressHomeLookup** ∗ahl=NULL)
- ∼**Cache** ()
- **Lock** & **getSetLock** (**IntPtr** addr)
- bool **invalidateSingleLine** (**IntPtr** addr)
- **CacheBlockInfo** ∗ **accessSingleLine** (**IntPtr** addr, **access_t** access_type, **Byte** ∗buff, **UInt32** bytes, **SubsecondTime** now, bool update_replacement)
- void **insertSingleLine** (**IntPtr** addr, **Byte** ∗fill_buff, bool ∗eviction, **IntPtr** ∗evict_addr, **CacheBlockInfo** ∗evict_block_info, **Byte** ∗evict_buff, **SubsecondTime** now, **CacheCntlr** ∗cntlr=NULL)
- **CacheBlockInfo** ∗ **peekSingleLine** (**IntPtr** addr)
- **CacheBlockInfo** ∗ **peekBlock** (**UInt32** set_index, **UInt32** way) const
- void **updateCounters** (bool cache_hit)
- void **updateHits** (**Core::mem_op_t** mem_op_type, **UInt64** hits)
- void **enable** ()
- void **disable** ()

Private Attributes

- bool **m_enabled**
- **UInt64 m_num_accesses**
- **UInt64 m_num_hits**
- **cache_t m_cache_type**
- **CacheSet** ∗∗ **m_sets**
- **CacheSetInfo** ∗ **m_set_info**
- **FaultInjector** ∗ **m_fault_injector**

Additional Inherited Members

6.27.1 Detailed Description

Definition at line 18 of file cache.h.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 Cache()

```
Cache::Cache (
            String name,
            String cfgname,
             core_id_t core_id,
             UInt32 num_sets,
             UInt32 associativity,
             UInt32 cache_block_size,
            String replacement_policy,
             cache_t cache_type,
             hash_t hash =  CacheBase::HASH_MASK,
             FaultInjector * fault_injector = NULL,
             AddressHomeLookup * ahl = NULL )
```

Definition at line 7 of file cache.cc.

References CacheSet::createCacheSet(), CacheSet::createCacheSetInfo(), CacheBase::m_associativity, CacheBase::m_blocksize, m_cache_type, CacheBase::m_num_sets, m_set_info, and m_sets.

6.27.2.2 ~Cache()

```
Cache::~Cache ( )
```

Definition at line 40 of file cache.cc.

References CacheBase::m_name, CacheBase::m_num_sets, m_set_info, and m_sets.

6.27.3 Member Function Documentation

6.27.3.1 accessSingleLine()

```
CacheBlockInfo * Cache::accessSingleLine (
             IntPtr addr,
             access_t access_type,
             Byte * buff,
             UInt32 bytes,
             SubsecondTime now,
            bool update_replacement )
```

Definition at line 82 of file cache.cc.

References CacheSet::find(), CacheBase::LOAD, CacheBase::m_associativity, m_fault_injector, m_sets, FaultInjector::postWrite(), FaultInjector::preRead(), CacheSet::read_line(), CacheBase::splitAddress(), and CacheSet::write_line().

Referenced by ParametricDramDirectoryMSI::CacheCntlr::accessCache(), DramCache::doAccess(), ParametricDramDirectoryMSI::TLB::lookup(), NucaCache::read(), ParametricDramDirectoryMSI::CacheCntlr::retrieveCacheBlock(), NucaCache::write(), and ParametricDramDirectoryMSI::CacheCntlr::writeCacheBlock().

6.27.3.2 disable()

```
void Cache::disable ( )  [inline]
```

Definition at line 70 of file cache.h.

References m_enabled.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::disable().

6.27.3.3 enable()

```
void Cache::enable ( )  [inline]
```

Definition at line 69 of file cache.h.

References m_enabled.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::enable().

6.27.3.4 getSetLock()

```
Lock & Cache::getSetLock (
            IntPtr addr )
```

Definition at line 59 of file cache.cc.

References CacheSet::getLock(), CacheBase::m_num_sets, m_sets, and CacheBase::splitAddress().

6.27.3.5 insertSingleLine()

```
void Cache::insertSingleLine (
             IntPtr addr,
             Byte * fill_buff,
            bool * eviction,
             IntPtr * evict_addr,
             CacheBlockInfo * evict_block_info,
             Byte * evict_buff,
             SubsecondTime now,
             CacheCntlr * cntlr = NULL )
```

Definition at line 120 of file cache.cc.

References __attribute__, CacheBlockInfo::create(), CacheSet::find(), CacheBlockInfo::getTag(), CacheSet::insert(), LOG_ASSERT_ERROR, CacheBase::m_associativity, m_cache_type, m_fault_injector, m_sets, FaultInjector::postWrite(), CacheBlockInfo::setTag(), CacheBase::splitAddress(), and CacheBase::tagToAddress().

Referenced by ParametricDramDirectoryMSI::TLB::allocate(), ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), DramCache::insertLine(), and NucaCache::write().

6.27.3.6 invalidateSingleLine()

```
bool Cache::invalidateSingleLine (
            IntPtr addr )
```

Definition at line 70 of file cache.cc.

References CacheSet::invalidate(), CacheBase::m_num_sets, m_sets, and CacheBase::splitAddress().

Referenced by ParametricDramDirectoryMSI::CacheCntlr::invalidateCacheBlock().

6.27.3.7 peekBlock()

```
CacheBlockInfo* Cache::peekBlock (
            UInt32 set_index,
            UInt32 way ) const  [inline]
```

Definition at line 63 of file cache.h.

References m_sets, and CacheSet::peekBlock().

Referenced by ParametricDramDirectoryMSI::CacheCntlr::flush(), ParametricDramDirectoryMSI::CacheCntlr↩ ::printCache(), and ParametricDramDirectoryMSI::CacheCntlr::walkUsageBits().

6.27.3.8 peekSingleLine()

```
CacheBlockInfo * Cache::peekSingleLine (
            IntPtr addr )
```

Definition at line 155 of file cache.cc.

References CacheSet::find(), m_sets, and CacheBase::splitAddress().

Referenced by DramCache::callPrefetcher(), DramCache::doAccess(), ParametricDramDirectoryMSI::Cache↩ Cntlr::getCacheBlockInfo(), DramCache::insertLine(), NucaCache::read(), and NucaCache::write().

6.27.3.9 updateCounters()

```
void Cache::updateCounters (
            bool cache_hit )
```

Definition at line 164 of file cache.cc.

References m_enabled, m_num_accesses, and m_num_hits.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore(), ParametricDram↩ DirectoryMSI::CacheCntlr::processShmemReqFromPrevCache(), and ParametricDramDirectoryMSI::Cache↩ Cntlr::updateHits().

6.27.3.10 updateHits()

```
void Cache::updateHits (
              Core::mem_op_t mem_op_type,
              UInt64 hits )
```

Definition at line 174 of file cache.cc.

References m_enabled, m_num_accesses, and m_num_hits.

6.27.4 Member Data Documentation

6.27.4.1 m_cache_type

```
cache_t Cache::m_cache_type  [private]
```

Definition at line 28 of file cache.h.

Referenced by Cache(), and insertSingleLine().

6.27.4.2 m_enabled

```
bool Cache::m_enabled  [private]
```

Definition at line 21 of file cache.h.

Referenced by disable(), enable(), updateCounters(), and updateHits().

6.27.4.3 m_fault_injector

```
FaultInjector* Cache::m_fault_injector  [private]
```

Definition at line 32 of file cache.h.

Referenced by accessSingleLine(), and insertSingleLine().

6.27.4.4 m_num_accesses

```
UInt64 Cache::m_num_accesses  [private]
```

Definition at line 24 of file cache.h.

Referenced by updateCounters(), and updateHits().

6.27.4.5 m_num_hits

```
UInt64 Cache::m_num_hits  [private]
```

Definition at line 25 of file cache.h.

Referenced by updateCounters(), and updateHits().

6.27.4.6 m_set_info

```
CacheSetInfo* Cache::m_set_info  [private]
```

Definition at line 30 of file cache.h.

Referenced by Cache(), and ~Cache().

6.27.4.7 m_sets

```
CacheSet** Cache::m_sets  [private]
```

Definition at line 29 of file cache.h.

Referenced by accessSingleLine(), Cache(), getSetLock(), insertSingleLine(), invalidateSingleLine(), peekBlock(), peekSingleLine(), and ~Cache().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache.h**
- common/core/memory_subsystem/cache/ **cache.cc**

6.28 FastNehalem::Cache< assoc, size_kb > Class Template Reference

```
#include <fast_cache.h>
```

Inheritance diagram for FastNehalem::Cache< assoc, size_kb >:

Public Member Functions

- **Cache** (**Core** ∗core, String name, **MemComponent::component_t** mem_component, **UInt64** latency, **CacheBase** ∗next_level)
- virtual ∼**Cache** ()
- **SubsecondTime access** (**Core::mem_op_t** mem_op_type, **IntPtr** tag)

Private Attributes

- const **MemComponent::component_t m_mem_component**
- const **ComponentLatency m_latency**
- **CacheBase** ∗const **m_next_level**
- const **UInt64 m_num_sets**
- const **IntPtr m_sets_mask**
- std::vector< **CacheSet**< assoc > > **m_sets**
- **UInt64 m_loads**
- **UInt64 m_stores**
- **UInt64 m_load_misses**
- **UInt64 m_store_misses**

6.28.1 Detailed Description

template<UInt32 assoc, UInt32 size_kb>
class FastNehalem::Cache< assoc, size_kb >

Definition at line 70 of file fast_cache.h.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 Cache()

```
template<UInt32 assoc, UInt32 size_kb>
FastNehalem::Cache< assoc, size_kb >::  Cache (
              Core * core,
             String name,
              MemComponent::component_t mem_component,
              UInt64 latency,
              CacheBase * next_level )  [inline]
```

Definition at line 82 of file fast_cache.h.

References floorLog2(), Core::getId(), LOG_ASSERT_ERROR, FastNehalem::Cache< assoc, size_kb >::m_load_misses, FastNehalem::Cache< assoc, size_kb >::m_loads, FastNehalem::Cache< assoc, size_kb >::m_num_sets, FastNehalem::Cache< assoc, size_kb >::m_store_misses, FastNehalem::Cache< assoc, size_kb >::m_stores, and registerStatsMetric().

6.28.2.2 ~Cache()

```
template<UInt32 assoc, UInt32 size_kb>
virtual  FastNehalem::Cache< assoc, size_kb >::~ Cache ( )  [inline], [virtual]
```

Definition at line 97 of file fast_cache.h.

6.28.3 Member Function Documentation

6.28.3.1 access()

```
template<UInt32 assoc, UInt32 size_kb>
SubsecondTime  FastNehalem::Cache< assoc, size_kb >::access (
              Core::mem_op_t mem_op_type,
              IntPtr tag )  [inline], [virtual]
```

Implements **FastNehalem::CacheBase** (p. 150).

Reimplemented in **FastNehalem::CacheLocked< assoc, size_kb >** (p. 201).

Definition at line 99 of file fast_cache.h.

References FastNehalem::CacheBase::access(), ComponentLatency::getLatency(), FastNehalem::Cache< assoc, size_kb >::m_latency, FastNehalem::Cache< assoc, size_kb >::m_load_misses, FastNehalem::Cache< assoc, size_kb >::m_loads, FastNehalem::Cache< assoc, size_kb >::m_next_level, FastNehalem::Cache< assoc, size_kb >::m_sets, FastNehalem::Cache< assoc, size_kb >::m_sets_mask, FastNehalem::Cache< assoc, size_kb >::m_store_misses, FastNehalem::Cache< assoc, size_kb >::m_stores, and Core::WRITE.

Referenced by FastNehalem::CacheLocked< assoc, size_kb >::access().

6.28.4 Member Data Documentation

6.28.4.1 m_latency

```
template<UInt32 assoc, UInt32 size_kb>
const  ComponentLatency  FastNehalem::Cache< assoc, size_kb >::m_latency  [private]
```

Definition at line 74 of file fast_cache.h.

Referenced by FastNehalem::Cache< assoc, size_kb >::access().

6.28.4.2 m_load_misses

```
template<UInt32 assoc, UInt32 size_kb>
UInt64 FastNehalem::Cache< assoc, size_kb >::m_load_misses [private]
```

Definition at line 79 of file fast_cache.h.

Referenced by FastNehalem::Cache< assoc, size_kb >::access(), and FastNehalem::Cache< assoc, size_kb >::Cache().

6.28.4.3 m_loads

```
template<UInt32 assoc, UInt32 size_kb>
UInt64 FastNehalem::Cache< assoc, size_kb >::m_loads [private]
```

Definition at line 79 of file fast_cache.h.

Referenced by FastNehalem::Cache< assoc, size_kb >::access(), and FastNehalem::Cache< assoc, size_kb >::Cache().

6.28.4.4 m_mem_component

```
template<UInt32 assoc, UInt32 size_kb>
const MemComponent::component_t FastNehalem::Cache< assoc, size_kb >::m_mem_component [private]
```

Definition at line 73 of file fast_cache.h.

6.28.4.5 m_next_level

```
template<UInt32 assoc, UInt32 size_kb>
CacheBase* const FastNehalem::Cache< assoc, size_kb >::m_next_level [private]
```

Definition at line 75 of file fast_cache.h.

Referenced by FastNehalem::Cache< assoc, size_kb >::access().

6.28.4.6 m_num_sets

```
template<UInt32 assoc, UInt32 size_kb>
const UInt64 FastNehalem::Cache< assoc, size_kb >::m_num_sets [private]
```

Definition at line 76 of file fast_cache.h.

Referenced by FastNehalem::Cache< assoc, size_kb >::Cache().

6.28.4.7 m_sets

```
template<UInt32 assoc, UInt32 size_kb>
std::vector< CacheSet<assoc> >  FastNehalem::Cache< assoc, size_kb >::m_sets  [private]
```

Definition at line 78 of file fast_cache.h.

Referenced by FastNehalem::Cache< assoc, size_kb >::access().

6.28.4.8 m_sets_mask

```
template<UInt32 assoc, UInt32 size_kb>
const  IntPtr  FastNehalem::Cache< assoc, size_kb >::m_sets_mask  [private]
```

Definition at line 77 of file fast_cache.h.

Referenced by FastNehalem::Cache< assoc, size_kb >::access().

6.28.4.9 m_store_misses

```
template<UInt32 assoc, UInt32 size_kb>
UInt64  FastNehalem::Cache< assoc, size_kb >::m_store_misses  [private]
```

Definition at line 79 of file fast_cache.h.

Referenced by FastNehalem::Cache< assoc, size_kb >::access(), and FastNehalem::Cache< assoc, size_kb >::Cache().

6.28.4.10 m_stores

```
template<UInt32 assoc, UInt32 size_kb>
UInt64  FastNehalem::Cache< assoc, size_kb >::m_stores  [private]
```

Definition at line 79 of file fast_cache.h.

Referenced by FastNehalem::Cache< assoc, size_kb >::access(), and FastNehalem::Cache< assoc, size_kb >::Cache().

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/fast_nehalem/ **fast_cache.h**

6.29 CacheBase Class Reference

`#include <cache_base.h>`

Inheritance diagram for CacheBase:

Public Types

- enum **access_t** {
  INVALID_ACCESS_TYPE, **MIN_ACCESS_TYPE**, **LOAD** = MIN_ACCESS_TYPE, **STORE**,
  MAX_ACCESS_TYPE = STORE, **NUM_ACCESS_TYPES** = MAX_ACCESS_TYPE - MIN_ACCESS_TYPE + 1 }
- enum **cache_t** {
  INVALID_CACHE_TYPE, **MIN_CACHE_TYPE**, **PR_L1_CACHE** = MIN_CACHE_TYPE, **PR_L2_CACHE**,
  SHARED_CACHE, **MAX_CACHE_TYPE** = SHARED_CACHE, **NUM_CACHE_TYPES** = MAX_CACHE↩
  _TYPE - MIN_CACHE_TYPE + 1 }
- enum **hash_t** {
  INVALID_HASH_TYPE, **HASH_MASK**, **HASH_MOD**, **HASH_RNG1_MOD**,
  HASH_RNG2_MOD, **HASH_PRIME_DIS**, **HASH_XOR_MOD**, **HASH_MER_MOD** }
- enum **ReplacementPolicy** {
  ROUND_ROBIN = 0, **LRU**, **LRU_QBS**, **NRU**,
  MRU, **NMRU**, **PLRU**, **SRRIP**,
  SRRIP_QBS, **RANDOM**, **KRUGER**, **NUM_REPLACEMENT_POLICIES** }

Public Member Functions

- **CacheBase** (String name, **UInt32** num_sets, **UInt32** associativity, **UInt32** cache_block_size, **Cache↩ Base::hash_t** hash, **AddressHomeLookup** *ahl=NULL)
- virtual ~**CacheBase** ()
- void **splitAddress** (const **IntPtr** addr, **IntPtr** &tag, **UInt32** &set_index) const
- void **splitAddress** (const **IntPtr** addr, **IntPtr** &tag, **UInt32** &set_index, **UInt32** &block_offset) const
- **IntPtr** **tagToAddress** (const **IntPtr** tag)
- String **getName** (void)
- **UInt32** **getNumSets** () const
- **UInt32** **getAssociativity** () const

Static Public Member Functions

- static **hash_t** **parseAddressHash** (String hash_name)

Protected Attributes

- String **m_name**
- **UInt64 m_cache_size**
- **UInt32 m_associativity**
- **UInt32 m_blocksize**
- **CacheBase::hash_t m_hash**
- **UInt32 m_num_sets**
- **AddressHomeLookup** * **m_ahl**
- **UInt32 m_log_blocksize**
- **UInt32 m_log_num_sets**

6.29.1 Detailed Description

Definition at line 14 of file cache_base.h.

6.29.2 Member Enumeration Documentation

6.29.2.1 access_t

enum **CacheBase::access_t**

Enumerator

INVALID_ACCESS_TYPE	
MIN_ACCESS_TYPE	
LOAD	
STORE	
MAX_ACCESS_TYPE	
NUM_ACCESS_TYPES	

Definition at line 18 of file cache_base.h.

6.29.2.2 cache_t

enum **CacheBase::cache_t**

Enumerator

INVALID_CACHE_TYPE	
MIN_CACHE_TYPE	
PR_L1_CACHE	
PR_L2_CACHE	
SHARED_CACHE	
MAX_CACHE_TYPE	
NUM_CACHE_TYPES	

Definition at line 28 of file cache_base.h.

6.29.2.3 hash_t

enum **CacheBase::hash_t**

Enumerator

INVALID_HASH_TYPE	
HASH_MASK	
HASH_MOD	
HASH_RNG1_MOD	
HASH_RNG2_MOD	
HASH_PRIME_DIS	
HASH_XOR_MOD	
HASH_MER_MOD	

Definition at line 39 of file cache_base.h.

6.29.2.4 ReplacementPolicy

enum **CacheBase::ReplacementPolicy**

Enumerator

ROUND_ROBIN	
LRU	
LRU_QBS	
NRU	
MRU	
NMRU	
PLRU	
SRRIP	
SRRIP_QBS	
RANDOM	
KRUGER	
NUM_REPLACEMENT_POLICIES	

Definition at line 51 of file cache_base.h.

6.29.3 Constructor & Destructor Documentation

6.29.3.1 CacheBase()

```
CacheBase::CacheBase (
            String name,
             UInt32 num_sets,
             UInt32 associativity,
             UInt32 cache_block_size,
             CacheBase::hash_t hash,
             AddressHomeLookup * ahl = NULL )
```

Definition at line 7 of file cache_base.cc.

References floorLog2(), HASH_MASK, LOG_ASSERT_ERROR, m_blocksize, m_log_blocksize, m_log_num_sets, and m_num_sets.

6.29.3.2 ~CacheBase()

```
CacheBase::~CacheBase ( )  [virtual]
```

Definition at line 26 of file cache_base.cc.

6.29.4 Member Function Documentation

6.29.4.1 getAssociativity()

```
UInt32 CacheBase::getAssociativity ( ) const  [inline]
```

Definition at line 94 of file cache_base.h.

References m_associativity.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::flush(), ParametricDramDirectoryMSI::CacheCntlr::printCache(), and ParametricDramDirectoryMSI::CacheCntlr::walkUsageBits().

6.29.4.2 getName()

```
String CacheBase::getName (
            void  )  [inline]
```

Definition at line 91 of file cache_base.h.

References m_name.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::printCache().

6.29.4.3 getNumSets()

```
UInt32 CacheBase::getNumSets ( ) const  [inline]
```

Definition at line 93 of file cache_base.h.

References m_num_sets.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::flush(), ParametricDramDirectoryMSI::CacheCntlr::printCache(), and ParametricDramDirectoryMSI::CacheCntlr::walkUsageBits().

6.29.4.4 parseAddressHash()

```
CacheBase::hash_t CacheBase::parseAddressHash (
            String hash_name )  [static]
```

Definition at line 31 of file cache_base.cc.

References HASH_MASK, HASH_MER_MOD, HASH_MOD, HASH_PRIME_DIS, HASH_RNG1_MOD, HASH_RNG2_MOD, HASH_XOR_MOD, and LOG_PRINT_ERROR.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), DramCache::DramCache(), and NucaCache::NucaCache().

6.29.4.5 splitAddress() [1/2]

```
void CacheBase::splitAddress (
            const  IntPtr addr,
             IntPtr & tag,
             UInt32 & set_index ) const
```

Definition at line 52 of file cache_base.cc.

References AddressHomeLookup::getLinearAddress(), HASH_MASK, HASH_MER_MOD, HASH_MOD, HASH_PRIME_DIS, HASH_RNG1_MOD, HASH_RNG2_MOD, HASH_XOR_MOD, LOG_PRINT_ERROR, m_ahl, m_hash, m_log_blocksize, m_log_num_sets, m_num_sets, rng_next(), and rng_seed().

Referenced by ATD::access(), Cache::accessSingleLine(), Cache::getSetLock(), Cache::insertSingleLine(), Cache::invalidateSingleLine(), Cache::peekSingleLine(), and splitAddress().

6.29.4.6 splitAddress() [2/2]

```
void CacheBase::splitAddress (
            const  IntPtr addr,
             IntPtr & tag,
             UInt32 & set_index,
             UInt32 & block_offset ) const
```

Definition at line 112 of file cache_base.cc.

References m_blocksize, and splitAddress().

6.29.4.7 tagToAddress()

```
IntPtr CacheBase::tagToAddress (
            const  IntPtr tag )
```

Definition at line 120 of file cache_base.cc.

References m_log_blocksize.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::flushBlock(), Cache::insertSingleLine(), and ParametricDramDirectoryMSI::CacheCntlr::isInLowerLevelCache().

6.29.5 Member Data Documentation

6.29.5.1 m_ahl

```
AddressHomeLookup* CacheBase::m_ahl  [protected]
```

Definition at line 76 of file cache_base.h.

Referenced by splitAddress().

6.29.5.2 m_associativity

```
UInt32 CacheBase::m_associativity  [protected]
```

Definition at line 72 of file cache_base.h.

Referenced by Cache::accessSingleLine(), Cache::Cache(), getAssociativity(), and Cache::insertSingleLine().

6.29.5.3 m_blocksize

```
UInt32 CacheBase::m_blocksize  [protected]
```

Definition at line 73 of file cache_base.h.

Referenced by Cache::Cache(), CacheBase(), and splitAddress().

6.29.5.4 m_cache_size

UInt64 CacheBase::m_cache_size [protected]

Definition at line 71 of file cache_base.h.

6.29.5.5 m_hash

CacheBase::hash_t CacheBase::m_hash [protected]

Definition at line 74 of file cache_base.h.

Referenced by splitAddress().

6.29.5.6 m_log_blocksize

UInt32 CacheBase::m_log_blocksize [protected]

Definition at line 79 of file cache_base.h.

Referenced by CacheBase(), splitAddress(), and tagToAddress().

6.29.5.7 m_log_num_sets

UInt32 CacheBase::m_log_num_sets [protected]

Definition at line 80 of file cache_base.h.

Referenced by CacheBase(), and splitAddress().

6.29.5.8 m_name

String CacheBase::m_name [protected]

Definition at line 70 of file cache_base.h.

Referenced by getName(), and Cache::~Cache().

6.29.5.9 m_num_sets

```
UInt32 CacheBase::m_num_sets  [protected]
```

Definition at line 75 of file cache_base.h.

Referenced by Cache::Cache(), CacheBase(), getNumSets(), Cache::getSetLock(), Cache::invalidateSingleLine(), splitAddress(), and Cache::∼Cache().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_base.h**
- common/core/memory_subsystem/cache/ **cache_base.cc**

6.30 FastNehalem::CacheBase Class Reference

```
#include <memory_manager.h>
```

Inheritance diagram for FastNehalem::CacheBase:

Public Member Functions

- virtual ∼**CacheBase** ()
- virtual **SubsecondTime access** (**Core::mem_op_t** mem_op_type, **IntPtr** tag)=0

6.30.1 Detailed Description

Definition at line 8 of file memory_manager.h.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 ∼CacheBase()

```
virtual FastNehalem::CacheBase::∼CacheBase ( )  [inline], [virtual]
```

Definition at line 11 of file memory_manager.h.

6.30.3 Member Function Documentation

6.30.3.1 access()

```
virtual SubsecondTime FastNehalem::CacheBase::access (
            Core::mem_op_t mem_op_type,
            IntPtr tag ) [pure virtual]
```

Implemented in **FastNehalem::CacheLocked< assoc, size_kb >** (p. 201), **FastNehalem::Cache< assoc, size_kb >** (p. 139), and **FastNehalem::Dram** (p. 451).

Referenced by FastNehalem::Cache< assoc, size_kb >::access().

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/fast_nehalem/ **memory_manager.h**

6.31 CacheBlockInfo Class Reference

`#include <cache_block_info.h>`

Inheritance diagram for CacheBlockInfo:

Public Types

- enum **option_t** { **PREFETCH**, **WARMUP**, **NUM_OPTIONS** }
- typedef **UInt8 BitsUsedType**

Public Member Functions

- **CacheBlockInfo** (**IntPtr** tag=∼0, **CacheState::cstate_t** cstate= **CacheState::INVALID**, **UInt64** options=0)
- virtual ∼**CacheBlockInfo** ()
- virtual void **invalidate** (void)
- virtual void **clone** (**CacheBlockInfo** ∗cache_block_info)
- bool **isValid** () const
- **IntPtr getTag** () const
- **CacheState::cstate_t getCState** () const
- void **setTag** (**IntPtr** tag)
- void **setCState** (**CacheState::cstate_t** cstate)
- **UInt64 getOwner** () const
- void **setOwner** (**UInt64** owner)
- bool **hasOption** (**option_t** option)
- void **setOption** (**option_t** option)
- void **clearOption** (**option_t** option)
- **BitsUsedType getUsage** () const
- bool **updateUsage** (**UInt32** offset, **UInt32** size)
- bool **updateUsage** (**BitsUsedType** used)

Static Public Member Functions

- static **CacheBlockInfo** * **create** (**CacheBase::cache_t** cache_type)
- static const char * **getOptionName** (**option_t** option)

Static Public Attributes

- static const **UInt8** **BitsUsedOffset** = 3

Private Attributes

- **IntPtr** **m_tag**
- **CacheState::cstate_t** **m_cstate**
- **UInt64** **m_owner**
- **BitsUsedType** **m_used**
- **UInt8** **m_options**

Static Private Attributes

- static const char * **option_names** [ ]

6.31.1 Detailed Description

Definition at line 8 of file cache_block_info.h.

6.31.2 Member Typedef Documentation

6.31.2.1 BitsUsedType

typedef **UInt8** **CacheBlockInfo::BitsUsedType**

Definition at line 19 of file cache_block_info.h.

6.31.3 Member Enumeration Documentation

6.31.3.1 option_t

enum **CacheBlockInfo::option_t**

Enumerator

PREFETCH	
WARMUP	
NUM_OPTIONS	

Definition at line 11 of file cache_block_info.h.

6.31.4 Constructor & Destructor Documentation

6.31.4.1 CacheBlockInfo()

```
CacheBlockInfo::CacheBlockInfo (
              IntPtr tag = ~0,
              CacheState::cstate_t cstate =  CacheState::INVALID,
              UInt64 options = 0 )
```

Definition at line 24 of file cache_block_info.cc.

6.31.4.2 ~CacheBlockInfo()

```
CacheBlockInfo::~CacheBlockInfo ( )  [virtual]
```

Definition at line 32 of file cache_block_info.cc.

6.31.5 Member Function Documentation

6.31.5.1 clearOption()

```
void CacheBlockInfo::clearOption (
              option_t option )  [inline]
```

Definition at line 56 of file cache_block_info.h.

References m_options.

Referenced by DramCache::doAccess(), ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore(), and ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache().

6.31.5.2 clone()

```
void CacheBlockInfo::clone (
            CacheBlockInfo * cache_block_info )  [virtual]
```

Reimplemented in **SharedCacheBlockInfo** (p. 1180), and **PrL2CacheBlockInfo** (p. 941).

Definition at line 63 of file cache_block_info.cc.

References getCState(), getTag(), m_cstate, m_options, m_owner, m_tag, and m_used.

Referenced by PrL2CacheBlockInfo::clone(), SharedCacheBlockInfo::clone(), and CacheSet::insert().

6.31.5.3 create()

```
CacheBlockInfo * CacheBlockInfo::create (
            CacheBase::cache_t cache_type )  [static]
```

Definition at line 36 of file cache_block_info.cc.

References LOG_PRINT_ERROR, CacheBase::PR_L1_CACHE, CacheBase::PR_L2_CACHE, and CacheBase::SHARED_CACHE.

Referenced by CacheSet::CacheSet(), and Cache::insertSingleLine().

6.31.5.4 getCState()

```
CacheState::cstate_t CacheBlockInfo::getCState ( ) const  [inline]
```

Definition at line 46 of file cache_block_info.h.

References m_cstate.

Referenced by clone(), ParametricDramDirectoryMSI::CacheCntlr::flush(), ParametricDramDirectoryMSI::CacheCntlr::getCacheState(), ParametricDramDirectoryMSI::CacheCntlr::initiateDirectoryAccess(), ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), DramCache::insertLine(), CacheSetKruger::isValidReplacement(), CacheSetKruger::isValidReplacement2(), ParametricDramDirectoryMSI::CacheCntlr::printCache(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache(), ParametricDramDirectoryMSI::CacheCntlr::updateCacheBlock(), NucaCache::write(), and ParametricDramDirectoryMSI::CacheCntlr::writeCacheBlock().

6.31.5.5 getOptionName()

```
const char * CacheBlockInfo::getOptionName (
            option_t option )  [static]
```

Definition at line 13 of file cache_block_info.cc.

References NUM_OPTIONS, and option_names.

6.31.5.6 getOwner()

```
UInt64 CacheBlockInfo::getOwner ( ) const  [inline]
```

Definition at line 51 of file cache_block_info.h.

References m_owner.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), ParametricDramDirectoryMSI::↩ CacheCntlr::processMemOpFromCore(), and ParametricDramDirectoryMSI::CacheCntlr::walkUsageBits().

6.31.5.7 getTag()

```
IntPtr CacheBlockInfo::getTag ( ) const  [inline]
```

Definition at line 45 of file cache_block_info.h.

References m_tag.

Referenced by clone(), ParametricDramDirectoryMSI::CacheCntlr::flushBlock(), Cache::insertSingleLine(), and ParametricDramDirectoryMSI::CacheCntlr::isInLowerLevelCache().

6.31.5.8 getUsage()

```
BitsUsedType CacheBlockInfo::getUsage ( ) const  [inline]
```

Definition at line 58 of file cache_block_info.h.

References m_used.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), ParametricDramDirectoryMSI::↩ CacheCntlr::processMemOpFromCore(), and ParametricDramDirectoryMSI::CacheCntlr::walkUsageBits().

6.31.5.9 hasOption()

```
bool CacheBlockInfo::hasOption (
            option_t option )  [inline]
```

Definition at line 54 of file cache_block_info.h.

References m_options.

Referenced by DramCache::doAccess(), ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), Parametric↩ DramDirectoryMSI::CacheCntlr::processMemOpFromCore(), ParametricDramDirectoryMSI::CacheCntlr::process↩ ShmemReqFromPrevCache(), ParametricDramDirectoryMSI::CacheCntlr::updateCacheBlock(), and Parametric↩ DramDirectoryMSI::CacheCntlr::walkUsageBits().

6.31.5.10 invalidate()

```
void CacheBlockInfo::invalidate (
            void  )  [virtual]
```

Reimplemented in **SharedCacheBlockInfo** (p. 1180), and **PrL2CacheBlockInfo** (p. 942).

Definition at line 56 of file cache_block_info.cc.

References CacheState::INVALID, m_cstate, and m_tag.

Referenced by PrL2CacheBlockInfo::invalidate(), SharedCacheBlockInfo::invalidate(), CacheSet::invalidate(), and ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore().

6.31.5.11 isValid()

```
bool CacheBlockInfo::isValid ( ) const  [inline]
```

Definition at line 43 of file cache_block_info.h.

References m_tag.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::walkUsageBits().

6.31.5.12 setCState()

```
void CacheBlockInfo::setCState (
             CacheState::cstate_t cstate )  [inline]
```

Definition at line 49 of file cache_block_info.h.

References m_cstate.

Referenced by DramCache::doAccess(), CacheSetKruger::injectTest(), DramCache::insertLine(), ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache(), ParametricDramDirectoryMSI::CacheCntlr::setCacheState(), ParametricDramDirectoryMSI::CacheCntlr::updateCacheBlock(), and NucaCache::write().

6.31.5.13 setOption()

```
void CacheBlockInfo::setOption (
             option_t option )  [inline]
```

Definition at line 55 of file cache_block_info.h.

References m_options.

Referenced by DramCache::callPrefetcher(), ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), and ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache().

6.31.5.14 setOwner()

```
void CacheBlockInfo::setOwner (
              UInt64 owner )  [inline]
```

Definition at line 52 of file cache_block_info.h.

References m_owner.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock().

6.31.5.15 setTag()

```
void CacheBlockInfo::setTag (
              IntPtr tag )  [inline]
```

Definition at line 48 of file cache_block_info.h.

References m_tag.

Referenced by Cache::insertSingleLine().

6.31.5.16 updateUsage() [1/2]

```
bool CacheBlockInfo::updateUsage (
              BitsUsedType used )
```

Definition at line 85 of file cache_block_info.cc.

References m_used.

6.31.5.17 updateUsage() [2/2]

```
bool CacheBlockInfo::updateUsage (
              UInt32 offset,
              UInt32 size )
```

Definition at line 73 of file cache_block_info.cc.

References BitsUsedOffset.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore(), and ParametricDram↩ DirectoryMSI::CacheCntlr::updateUsageBits().

6.31.6 Member Data Documentation

6.31.6.1 BitsUsedOffset

```
const UInt8 CacheBlockInfo::BitsUsedOffset = 3 [static]
```

Definition at line 18 of file cache_block_info.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), updateUsage(), and ParametricDramDirectoryMSI::CacheCntlr::walkUsageBits().

6.31.6.2 m_cstate

```
CacheState::cstate_t CacheBlockInfo::m_cstate [private]
```

Definition at line 25 of file cache_block_info.h.

Referenced by clone(), getCState(), invalidate(), and setCState().

6.31.6.3 m_options

```
UInt8 CacheBlockInfo::m_options [private]
```

Definition at line 28 of file cache_block_info.h.

Referenced by clearOption(), clone(), hasOption(), and setOption().

6.31.6.4 m_owner

```
UInt64 CacheBlockInfo::m_owner [private]
```

Definition at line 26 of file cache_block_info.h.

Referenced by clone(), getOwner(), and setOwner().

6.31.6.5 m_tag

```
IntPtr CacheBlockInfo::m_tag [private]
```

Definition at line 24 of file cache_block_info.h.

Referenced by clone(), getTag(), invalidate(), isValid(), and setTag().

6.31.6.6 m_used

```
BitsUsedType CacheBlockInfo::m_used [private]
```

Definition at line 27 of file cache_block_info.h.

Referenced by clone(), getUsage(), and updateUsage().

6.31.6.7 option_names

```
const char * CacheBlockInfo::option_names [static], [private]
```

Initial value:

```
=
{
   "prefetch",
   "warmup",
}
```

Definition at line 30 of file cache_block_info.h.

Referenced by getOptionName().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_block_info.h**
- common/core/memory_subsystem/cache/ **cache_block_info.cc**

6.32 ParametricDramDirectoryMSI::CacheCntlr Class Reference

```
#include <cache_cntlr.h>
```

Inheritance diagram for ParametricDramDirectoryMSI::CacheCntlr:

Public Member Functions

- **CacheCntlr** (**MemComponent::component_t** mem_component, String name, **core_id_t** core_id, **MemoryManager** *memory_manager, **AddressHomeLookup** *tag_directory_home_lookup, **Semaphore** *user_thread_sem, **Semaphore** *network_thread_sem, **UInt32** cache_block_size, **CacheParameters** &cache_params, **ShmemPerfModel** *shmem_perf_model, bool is_last_level_cache)
- virtual ~**CacheCntlr** ()
- **Cache** * **getCache** ()
- **Lock** & **getLock** ()
- void **setPrevCacheCntlrs** (**CacheCntlrList** &prev_cache_cntlrs)
- void **setNextCacheCntlr** (**CacheCntlr** *next_cache_cntlr)
- void **createSetLocks** (**UInt32** cache_block_size, **UInt32** num_sets, **UInt32** core_offset, **UInt32** num_cores)
- void **setDRAMDirectAccess** (**DramCntlrInterface** *dram_cntlr, **UInt64** num_outstanding)
- **HitWhere::where_t** **processMemOpFromCore** (**Core::lock_signal_t** lock_signal, **Core::mem_op_t** mem_op_type, **IntPtr** ca_address, **UInt32** offset, **Byte** *data_buf, **UInt32** data_length, bool modeled, bool count)
- void **updateHits** (**Core::mem_op_t** mem_op_type, **UInt64** hits)
- void **notifyPrevLevelInsert** (**core_id_t** core_id, **MemComponent::component_t** mem_component, **IntPtr** address)
- void **notifyPrevLevelEvict** (**core_id_t** core_id, **MemComponent::component_t** mem_component, **IntPtr** address)
- void **handleMsgFromDramDirectory** (**core_id_t** sender, **PrL1PrL2DramDirectoryMSI::ShmemMsg** *shmem_msg)
- void **acquireLock** (**UInt64** address)
- void **releaseLock** (**UInt64** address)
- void **acquireStackLock** (**UInt64** address, bool this_is_locked=false)
- void **releaseStackLock** (**UInt64** address, bool this_is_locked=false)
- bool **isMasterCache** (void)
- bool **isFirstLevel** (void)
- bool **isLastLevel** (void)
- bool **isShared** (**core_id_t** core_id)
- bool **isInLowerLevelCache** (**CacheBlockInfo** *block_info)
- void **incrementQBSLookupCost** ()
- void **flush** ()
- void **enable** ()
- void **disable** ()

Private Member Functions

- void **updateCounters** (**Core::mem_op_t** mem_op_type, **IntPtr** address, bool cache_hit, **CacheState::cstate_t** state, **Prefetch::prefetch_type_t** isPrefetch)
- void **cleanupMshr** ()
- void **transition** (**IntPtr** address, **Transition::reason_t** reason, **CacheState::cstate_t** old_state, **CacheState::cstate_t** new_state)
- void **updateUncoreStatistics** (**HitWhere::where_t** hit_where, **SubsecondTime** now)
- void **accessCache** (**Core::mem_op_t** mem_op_type, **IntPtr** ca_address, **UInt32** offset, **Byte** *data_buf, **UInt32** data_length, bool update_replacement)
- bool **operationPermissibleinCache** (**IntPtr** address, **Core::mem_op_t** mem_op_type, **CacheBlockInfo** **cache_block_info=NULL)
- void **copyDataFromNextLevel** (**Core::mem_op_t** mem_op_type, **IntPtr** address, bool modeled, **SubsecondTime** **t_start**)
- void **trainPrefetcher** (**IntPtr** address, bool cache_hit, bool prefetch_hit, **SubsecondTime** t_issue)
- void **Prefetch** (**SubsecondTime** **t_start**)
- void **doPrefetch** (**IntPtr** prefetch_address, **SubsecondTime** **t_start**)

- **SharedCacheBlockInfo** * **getCacheBlockInfo** (**IntPtr** address)
- **CacheState::cstate_t** **getCacheState** (**IntPtr** address)
- **CacheState::cstate_t** **getCacheState** (**CacheBlockInfo** *cache_block_info)
- **SharedCacheBlockInfo** * **setCacheState** (**IntPtr** address, **CacheState::cstate_t** cstate)
- void **invalidateCacheBlock** (**IntPtr** address)
- void **retrieveCacheBlock** (**IntPtr** address, **Byte** *data_buf, **ShmemPerfModel::Thread_t** thread_num, bool update_replacement)
- **SharedCacheBlockInfo** * **insertCacheBlock** (**IntPtr** address, **CacheState::cstate_t** cstate, **Byte** *data_buf, **core_id_t** requester, **ShmemPerfModel::Thread_t** thread_num)
- std::pair< **SubsecondTime**, bool > **updateCacheBlock** (**IntPtr** address, **CacheState::cstate_t** cstate, **Transition::reason_t** reason, **Byte** *out_buf, **ShmemPerfModel::Thread_t** thread_num)
- void **writeCacheBlock** (**IntPtr** address, **UInt32** offset, **Byte** *data_buf, **UInt32** data_length, **ShmemPerfModel::Thread_t** thread_num)
- **HitWhere::where_t** **processShmemReqFromPrevCache** (**CacheCntlr** *requester, **Core::mem_op_t** mem_op_type, **IntPtr** address, bool modeled, bool count, **Prefetch::prefetch_type_t** isPrefetch, **SubsecondTime** t_issue, bool have_write_lock)
- boost::tuple< **HitWhere::where_t**, **SubsecondTime** > **accessDRAM** (**Core::mem_op_t** mem_op_type, **IntPtr** address, bool isPrefetch, **Byte** *data_buf)
- void **initiateDirectoryAccess** (**Core::mem_op_t** mem_op_type, **IntPtr** address, bool isPrefetch, **SubsecondTime** t_issue)
- void **processExReqToDirectory** (**IntPtr** address)
- void **processShReqToDirectory** (**IntPtr** address)
- void **processUpgradeReqToDirectory** (**IntPtr** address, **ShmemPerf** *perf, **ShmemPerfModel::Thread_t** thread_num)
- void **processExRepFromDramDirectory** (**core_id_t** sender, **core_id_t** requester, **PrL1PrL2DramDirectoryMSI::ShmemMsg** *shmem_msg)
- void **processShRepFromDramDirectory** (**core_id_t** sender, **core_id_t** requester, **PrL1PrL2DramDirectoryMSI::ShmemMsg** *shmem_msg)
- void **processUpgradeRepFromDramDirectory** (**core_id_t** sender, **core_id_t** requester, **PrL1PrL2DramDirectoryMSI::ShmemMsg** *shmem_msg)
- void **processInvReqFromDramDirectory** (**core_id_t** sender, **PrL1PrL2DramDirectoryMSI::ShmemMsg** *shmem_msg)
- void **processFlushReqFromDramDirectory** (**core_id_t** sender, **PrL1PrL2DramDirectoryMSI::ShmemMsg** *shmem_msg)
- void **processWbReqFromDramDirectory** (**core_id_t** sender, **PrL1PrL2DramDirectoryMSI::ShmemMsg** *shmem_msg)
- **UInt32** **getCacheBlockSize** ()
- **MemoryManager** * **getMemoryManager** ()
- **ShmemPerfModel** * **getShmemPerfModel** ()
- void **updateUsageBits** (**IntPtr** address, **CacheBlockInfo::BitsUsedType** used)
- void **walkUsageBits** ()
- void **wakeUpUserThread** (**Semaphore** *user_thread_sem=NULL)
- void **waitForUserThread** (**Semaphore** *network_thread_sem=NULL)
- void **waitForNetworkThread** (void)
- void **wakeUpNetworkThread** (void)
- **Semaphore** * **getUserThreadSemaphore** (void)
- **Semaphore** * **getNetworkThreadSemaphore** (void)
- **core_id_t** **getHome** (**IntPtr** address)
- **CacheCntlr** * **lastLevelCache** (void)
- void **flushBlock** (**CacheBlockInfo** *block_info)
- void **printCache** ()

Static Private Member Functions

- static **SInt64** **__walkUsageBits** (**UInt64** arg0, **UInt64** arg1)

Private Attributes

- **MemComponent::component_t m_mem_component**
- **MemoryManager** * **m_memory_manager**
- **CacheMasterCntlr** * **m_master**
- **CacheCntlr** * **m_next_cache_cntlr**
- **CacheCntlr** * **m_last_level**
- **AddressHomeLookup** * **m_tag_directory_home_lookup**
- std::unordered_map< **IntPtr**, **MemComponent::component_t** > **m_shmem_req_source_map**
- bool **m_perfect**
- bool **m_passthrough**
- bool **m_coherent**
- bool **m_prefetch_on_prefetch_hit**
- bool **m_l1_mshr**
- struct {
  UInt64 loads
  UInt64 stores
  UInt64 load_misses
  UInt64 store_misses
  UInt64 load_overlapping_misses
  UInt64 store_overlapping_misses
  UInt64 loads_state [**CacheState::NUM_CSTATE_STATES**]
  UInt64 stores_state [**CacheState::NUM_CSTATE_STATES**]
  UInt64 loads_where [**HitWhere::NUM_HITWHERES**]
  UInt64 stores_where [**HitWhere::NUM_HITWHERES**]
  UInt64 load_misses_state [**CacheState::NUM_CSTATE_STATES**]
  UInt64 store_misses_state [**CacheState::NUM_CSTATE_STATES**]
  UInt64 loads_prefetch
  UInt64 stores_prefetch
  UInt64 hits_prefetch
  UInt64 evict_prefetch
  UInt64 invalidate_prefetch
  UInt64 evict [**CacheState::NUM_CSTATE_STATES**]
  UInt64 backinval [**CacheState::NUM_CSTATE_STATES**]
  UInt64 hits_warmup
  UInt64 evict_warmup
  UInt64 invalidate_warmup
  SubsecondTime total_latency
  SubsecondTime snoop_latency
  SubsecondTime qbs_query_latency
  SubsecondTime mshr_latency
  UInt64 prefetches
  UInt64 coherency_downgrades
  UInt64 coherency_upgrades
  UInt64 coherency_invalidates
  UInt64 coherency_writebacks
  } **stats**

- **core_id_t m_core_id**
- **UInt32 m_cache_block_size**
- bool **m_cache_writethrough**
- **ComponentLatency m_writeback_time**
- **ComponentBandwidthPerCycle m_next_level_read_bandwidth**
- **UInt32 m_shared_cores**
- **core_id_t m_core_id_master**
- **Semaphore** * **m_user_thread_sem**
- **Semaphore** * **m_network_thread_sem**

- volatile **HitWhere::where_t m_last_remote_hit_where**
- **ShmemPerf** * **m_shmem_perf**
- **ShmemPerf** * **m_shmem_perf_global**
- **SubsecondTime m_shmem_perf_totaltime**
- **UInt64 m_shmem_perf_numrequests**
- **ShmemPerf m_dummy_shmem_perf**
- **ShmemPerfModel** * **m_shmem_perf_model**

Friends

- class **CacheCntlrList**
- class **MemoryManager**

6.32.1 Detailed Description

Definition at line 201 of file cache_cntlr.h.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 CacheCntlr()

```
CacheCntlr::CacheCntlr (
              MemComponent::component_t mem_component,
             String name,
              core_id_t core_id,
              MemoryManager * memory_manager,
              AddressHomeLookup * tag_directory_home_lookup,
              Semaphore * user_thread_sem,
              Semaphore * network_thread_sem,
              UInt32 cache_block_size,
              CacheParameters & cache_params,
              ShmemPerfModel * shmem_perf_model,
             bool is_last_level_cache )
```

Definition at line 124 of file cache_cntlr.cc.

References __walkUsageBits(), ParametricDramDirectoryMSI::CacheParameters::associativity, ParametricDramDirectoryMSI::CacheParameters::configName, ParametricDramDirectoryMSI::CacheMasterCntlr::createATDs(), Prefetcher::createPrefetcher(), CacheState::CSTATE_FIRST, ParametricDramDirectoryMSI::CStateString(), ParametricDramDirectoryMSI::MemoryManager::getCacheCntlrAt(), ShmemPerf::getComponent(), getMemoryManager(), ParametricDramDirectoryMSI::CacheParameters::hash_function, HitWhereString(), HookType::HOOK_ROI_END, isMasterCache(), MemComponent::L1_DCACHE, MemComponent::L1_ICACHE, LOG_ASSERT_ERROR, ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, m_cache_block_size, m_core_id, m_core_id_master, m_master, m_passthrough, m_prefetch_on_prefetch_hit, ParametricDramDirectoryMSI::CacheMasterCntlr::m_prefetcher, m_shared_cores, m_shmem_perf_global, m_shmem_perf_numrequests, m_shmem_perf_totaltime, CacheState::NUM_CSTATE_SPECIAL_STATES, CacheState::NUM_CSTATE_STATES, HitWhere::NUM_HITWHERES, ParametricDramDirectoryMSI::Transition::NUM_REASONS, ParametricDramDirectoryMSI::CacheParameters::num_sets, ShmemPerf::NUM_SHMEM_TIMES, HooksManager::ORDER_NOTIFY_PRE, ParametricDramDirectoryMSI::CacheParameters::outstanding_misses, CacheBase::parseAddressHash(), ParametricDramDirectoryMSI::CacheParameters::prefetcher, ParametricDramDirectoryMSI::Transition::REASON_FIRST, ParametricDramDirectoryMSI::ReasonString(), registerStatsMetric(), ParametricDramDirectoryMSI::CacheParameters::replacement_policy, CacheBase::SHARED_CACHE, ShmemReasonString(), stats, HitWhere::WHERE_FIRST, and SubsecondTime::Zero().

6.32.2.2 ~CacheCntlr()

```
CacheCntlr::~CacheCntlr ( )  [virtual]
```

Definition at line 275 of file cache_cntlr.cc.

References HitWhereString(), isMasterCache(), m_core_id, m_master, m_shmem_perf, and m_shmem_perf_↩ global.

6.32.3 Member Function Documentation

6.32.3.1 __walkUsageBits()

```
static  SInt64 ParametricDramDirectoryMSI::CacheCntlr::__walkUsageBits (
              UInt64 arg0,
              UInt64 arg1 )  [inline], [static], [private]
```

Definition at line 331 of file cache_cntlr.h.

Referenced by CacheCntlr().

6.32.3.2 accessCache()

```
void CacheCntlr::accessCache (
              Core::mem_op_t mem_op_type,
              IntPtr ca_address,
              UInt32 offset,
              Byte * data_buf,
              UInt32 data_length,
             bool update_replacement )  [private]
```

Definition at line 1272 of file cache_cntlr.cc.

References ShmemPerfModel::_USER_THREAD, Cache::accessSingleLine(), getShmemPerfModel(), Cache↩ Base::LOAD, LOG_ASSERT_ERROR, LOG_PRINT_ERROR, ParametricDramDirectoryMSI::CacheMasterCntlr↩ ::m_cache, m_cache_writethrough, m_master, m_next_cache_cntlr, MYLOG, Core::READ, Core::READ_EX, CacheBase::STORE, Core::WRITE, and writeCacheBlock().

Referenced by processMemOpFromCore().

6.32.3.3 accessDRAM()

```
boost::tuple< HitWhere::where_t, SubsecondTime > CacheCntlr::accessDRAM (
              Core::mem_op_t mem_op_type,
              IntPtr address,
             bool isPrefetch,
              Byte * data_buf )  [private]
```

Definition at line 1091 of file cache_cntlr.cc.

References ShmemPerfModel::_USER_THREAD, DramCntlrInterface::getDataFromDram(), ShmemPerfModel::getElapsedTime(), getLock(), getShmemPerfModel(), LOG_PRINT_ERROR, m_core_id_master, ParametricDramDirectoryMSI::CacheMasterCntlr::m_dram_cntlr, m_master, m_shmem_perf, DramCntlrInterface::putDataToDram(), Core::READ, Core::READ_EX, and Core::WRITE.

Referenced by insertCacheBlock(), and processShmemReqFromPrevCache().

6.32.3.4 acquireLock()

```
void CacheCntlr::acquireLock (
              UInt64 address )
```

Definition at line 2252 of file cache_cntlr.cc.

References _SetLock::acquire_shared(), ParametricDramDirectoryMSI::CacheMasterCntlr::getSetLock(), isFirstLevel(), lastLevelCache(), m_core_id, m_master, m_mem_component, and MYLOG.

Referenced by processMemOpFromCore().

6.32.3.5 acquireStackLock()

```
void CacheCntlr::acquireStackLock (
              UInt64 address,
             bool this_is_locked = false )
```

Definition at line 2267 of file cache_cntlr.cc.

References _SetLock::acquire_exclusive(), ParametricDramDirectoryMSI::CacheMasterCntlr::getSetLock(), lastLevelCache(), m_core_id, m_master, m_mem_component, MYLOG, and _SetLock::upgrade().

Referenced by doPrefetch(), handleMsgFromDramDirectory(), processMemOpFromCore(), processShmemReqFromPrevCache(), and writeCacheBlock().

6.32.3.6 cleanupMshr()

```
void CacheCntlr::cleanupMshr ( )  [private]
```

Definition at line 2171 of file cache_cntlr.cc.

References m_master, SubsecondTime::MaxTime(), and ParametricDramDirectoryMSI::CacheMasterCntlr::mshr.

Referenced by handleMsgFromDramDirectory(), processShmemReqFromPrevCache(), and updateCounters().

6.32.3.7 copyDataFromNextLevel()

```
void CacheCntlr::copyDataFromNextLevel (
              Core::mem_op_t mem_op_type,
              IntPtr address,
             bool modeled,
              SubsecondTime t_start )  [private]
```

Definition at line 622 of file cache_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, ShmemPerfModel::_USER_THREAD, getCacheBlockInfo(), getCacheBlockSize(), getCacheState(), ContentionModel::getCompletionTime(), getMemoryManager(), ComponentBandwidthPerCycle::getRoundedLatency(), ParametricDramDirectoryMSI::MemoryManager::incrElapsedTime(), insertCacheBlock(), ComponentBandwidthPerCycle::isInfinite(), LOG_ASSERT_ERROR, m_core_id, m_master, m_mem_component, m_next_cache_cntlr, ParametricDramDirectoryMSI::CacheMasterCntlr::m_next_level_read_bandwidth, m_next_level_read_bandwidth, MYLOG, operationPermissibleinCache(), retrieveCacheBlock(), updateCacheBlock(), and ParametricDramDirectoryMSI::Transition::UPGRADE.

Referenced by processMemOpFromCore(), and processShmemReqFromPrevCache().

6.32.3.8 createSetLocks()

```
void ParametricDramDirectoryMSI::CacheCntlr::createSetLocks (
              UInt32 cache_block_size,
              UInt32 num_sets,
              UInt32 core_offset,
              UInt32 num_cores )  [inline]
```

Definition at line 375 of file cache_cntlr.h.

References ParametricDramDirectoryMSI::CacheMasterCntlr::createSetLocks(), and m_master.

6.32.3.9 disable()

```
void ParametricDramDirectoryMSI::CacheCntlr::disable ( )  [inline]
```

Definition at line 410 of file cache_cntlr.h.

References Cache::disable(), ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, and m_master.

6.32.3.10 doPrefetch()

```
void CacheCntlr::doPrefetch (
            IntPtr prefetch_address,
            SubsecondTime t_start ) [private]
```

Definition at line 725 of file cache_cntlr.cc.

References ShmemPerfModel::_USER_THREAD, acquireStackLock(), ShmemPerfModel::getElapsedTime(), getShmemPerfModel(), LOG_ASSERT_ERROR, HitWhere::MISS, MYLOG, ParametricDramDirectoryMSI::Prefetch::OWN, processShmemReqFromPrevCache(), Core::READ, releaseStackLock(), ShmemPerfModel::setElapsedTime(), stats, t_start, waitForNetworkThread(), and wakeUpNetworkThread().

Referenced by Prefetch().

6.32.3.11 enable()

```
void ParametricDramDirectoryMSI::CacheCntlr::enable ( ) [inline]
```

Definition at line 409 of file cache_cntlr.h.

References Cache::enable(), ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, and m_master.

6.32.3.12 flush()

```
void CacheCntlr::flush ( ) [virtual]
```

Reimplemented from **CacheCntlr** (p. 197).

Definition at line 1750 of file cache_cntlr.cc.

References flushBlock(), CacheBase::getAssociativity(), CacheBlockInfo::getCState(), CacheBase::getNumSets(), ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, m_master, CacheState::MODIFIED, Cache::peekBlock(), and printCache().

6.32.3.13 flushBlock()

```
void CacheCntlr::flushBlock (
            CacheBlockInfo * block_info ) [private]
```

Definition at line 1766 of file cache_cntlr.cc.

References MemComponent::DRAM, PrL1PrL2DramDirectoryMSI::ShmemMsg::DRAM_WRITE_REQ, getCacheBlockSize(), ParametricDramDirectoryMSI::MemoryManager::getDramCntlr(), getMemoryManager(), CacheBlockInfo::getTag(), DramCntlrInterface::handleMsgFromTagDirectory(), ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, m_core_id, m_dummy_shmem_perf, m_master, MemComponent::TAG_DIR, and CacheBase::tagToAddress().

Referenced by flush().

6.32.3.14 getCache()

```
Cache* ParametricDramDirectoryMSI::CacheCntlr::getCache ( )  [inline]
```

Definition at line 370 of file cache_cntlr.h.

References ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, and m_master.

Referenced by ParametricDramDirectoryMSI::MemoryManager::getCache(), processMemOpFromCore(), processShmemReqFromPrevCache(), and updateHits().

6.32.3.15 getCacheBlockInfo()

```
SharedCacheBlockInfo * CacheCntlr::getCacheBlockInfo (
            IntPtr address )  [private]
```

Definition at line 1310 of file cache_cntlr.cc.

References ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, m_master, and Cache::peekSingleLine().

Referenced by copyDataFromNextLevel(), getCacheState(), handleMsgFromDramDirectory(), initiateDirectory↩Access(), notifyPrevLevelEvict(), notifyPrevLevelInsert(), operationPermissibleinCache(), processMemOpFrom↩Core(), processShmemReqFromPrevCache(), setCacheState(), updateCacheBlock(), and updateUsageBits().

6.32.3.16 getCacheBlockSize()

```
UInt32 ParametricDramDirectoryMSI::CacheCntlr::getCacheBlockSize ( )  [inline], [private]
```

Definition at line 325 of file cache_cntlr.h.

References m_cache_block_size.

Referenced by copyDataFromNextLevel(), flushBlock(), insertCacheBlock(), processFlushReqFromDram↩Directory(), processMemOpFromCore(), processShmemReqFromPrevCache(), processWbReqFromDram↩Directory(), retrieveCacheBlock(), updateCacheBlock(), walkUsageBits(), and writeCacheBlock().

6.32.3.17 getCacheState() [1/2]

```
CacheState::cstate_t CacheCntlr::getCacheState (
            CacheBlockInfo * cache_block_info )  [private]
```

Definition at line 1323 of file cache_cntlr.cc.

References CacheBlockInfo::getCState(), and CacheState::INVALID.

6.32.3.18 getCacheState() [2/2]

```
CacheState::cstate_t CacheCntlr::getCacheState (
            IntPtr address ) [private]
```

Definition at line 1316 of file cache_cntlr.cc.

References getCacheBlockInfo().

Referenced by copyDataFromNextLevel(), handleMsgFromDramDirectory(), insertCacheBlock(), invalidateCache↩ Block(), notifyPrevLevelEvict(), operationPermissibleinCache(), processExReqToDirectory(), processFlushReq↩ FromDramDirectory(), processInvReqFromDramDirectory(), processMemOpFromCore(), processShmemReq↩ FromPrevCache(), processUpgradeRepFromDramDirectory(), processUpgradeReqToDirectory(), processWb↩ ReqFromDramDirectory(), and updateCacheBlock().

6.32.3.19 getHome()

```
core_id_t ParametricDramDirectoryMSI::CacheCntlr::getHome (
            IntPtr address ) [inline], [private]
```

Definition at line 347 of file cache_cntlr.h.

References AddressHomeLookup::getHome(), and m_tag_directory_home_lookup.

Referenced by insertCacheBlock(), processExReqToDirectory(), processShReqToDirectory(), and process↩ UpgradeReqToDirectory().

6.32.3.20 getLock()

```
Lock& ParametricDramDirectoryMSI::CacheCntlr::getLock ( ) [inline]
```

Definition at line 371 of file cache_cntlr.h.

References ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache_lock, and m_master.

Referenced by accessDRAM(), handleMsgFromDramDirectory(), initiateDirectoryAccess(), insertCacheBlock(), Prefetch(), processMemOpFromCore(), processShmemReqFromPrevCache(), trainPrefetcher(), updateCache↩ Block(), updateHits(), and updateUsageBits().

6.32.3.21 getMemoryManager()

```
MemoryManager* ParametricDramDirectoryMSI::CacheCntlr::getMemoryManager ( ) [inline], [private]
```

Definition at line 326 of file cache_cntlr.h.

References m_memory_manager.

Referenced by CacheCntlr(), copyDataFromNextLevel(), flushBlock(), incrementQBSLookupCost(), insert↩ CacheBlock(), processExReqToDirectory(), processFlushReqFromDramDirectory(), processInvReqFromDram↩ Directory(), processMemOpFromCore(), processShmemReqFromPrevCache(), processShReqToDirectory(), processUpgradeReqToDirectory(), and processWbReqFromDramDirectory().

6.32.3.22 getNetworkThreadSemaphore()

```
Semaphore * CacheCntlr::getNetworkThreadSemaphore (
            void ) [private]
```

Definition at line 2337 of file cache_cntlr.cc.

References m_network_thread_sem.

6.32.3.23 getShmemPerfModel()

```
ShmemPerfModel* ParametricDramDirectoryMSI::CacheCntlr::getShmemPerfModel ( ) [inline], [private]
```

Definition at line 327 of file cache_cntlr.h.

References m_shmem_perf_model.

Referenced by accessCache(), accessDRAM(), doPrefetch(), handleMsgFromDramDirectory(), initiate↩ DirectoryAccess(), insertCacheBlock(), processFlushReqFromDramDirectory(), processInvReqFromDram↩ Directory(), processMemOpFromCore(), processShmemReqFromPrevCache(), processWbReqFromDram↩ Directory(), retrieveCacheBlock(), updateCounters(), and writeCacheBlock().

6.32.3.24 getUserThreadSemaphore()

```
Semaphore * CacheCntlr::getUserThreadSemaphore (
            void ) [private]
```

Definition at line 2331 of file cache_cntlr.cc.

References m_user_thread_sem.

6.32.3.25 handleMsgFromDramDirectory()

```
void CacheCntlr::handleMsgFromDramDirectory (
             core_id_t sender,
             PrL1PrL2DramDirectoryMSI::ShmemMsg * shmem_msg )
```

Definition at line 1821 of file cache_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, ShmemPerfModel::_USER_THREAD, TLock< T_LockCreator >::acquire(), acquireStackLock(), ParametricDramDirectoryMSI::CacheDirectoryWaiter::cache_cntlr, cleanup↩ Mshr(), ReqQueueListTemplate< T_Req >::dequeue(), ReqQueueListTemplate< T_Req >::empty(), PrL1↩ PrL2DramDirectoryMSI::ShmemMsg::EX_REP, ParametricDramDirectoryMSI::CacheDirectoryWaiter::exclusive, PrL1PrL2DramDirectoryMSI::ShmemMsg::FLUSH_REQ, ReqQueueListTemplate< T_Req >::front(), PrL1PrL2↩ DramDirectoryMSI::ShmemMsg::getAddress(), getCacheBlockInfo(), getCacheState(), ShmemPerfModel::get↩ ElapsedTime(), getLock(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), SubsecondTime::getNS(),

getShmemPerfModel(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getWhere(), PrL1PrL2DramDirectoryMSI::↩ ShmemMsg::INV_REQ, INVALID_CORE_ID, ParametricDramDirectoryMSI::CacheDirectoryWaiter::isPrefetch, LOG_PRINT_ERROR, m_core_id, ParametricDramDirectoryMSI::CacheMasterCntlr::m_directory_waiters, m_↩ master, m_network_thread_sem, m_shmem_perf, m_user_thread_sem, ParametricDramDirectoryMSI::make_↩ mshr(), ParametricDramDirectoryMSI::CacheMasterCntlr::mshr, MYLOG, ShmemPerf::PENDING_HIT, Cache↩ BlockInfo::PREFETCH, processExRepFromDramDirectory(), processFlushReqFromDramDirectory(), process↩ InvReqFromDramDirectory(), processShRepFromDramDirectory(), processUpgradeRepFromDramDirectory(), processUpgradeReqToDirectory(), processWbReqFromDramDirectory(), TLock< T_LockCreator >::release(), releaseStackLock(), ShmemPerfModel::setElapsedTime(), CacheBlockInfo::setOption(), PrL1PrL2DramDirectory↩ MSI::ShmemMsg::SH_REP, CacheState::SHARED, ParametricDramDirectoryMSI::CacheDirectoryWaiter::t_issue, ShmemPerf::updateTime(), updateUncoreStatistics(), PrL1PrL2DramDirectoryMSI::ShmemMsg::UPGRADE_REP, waitForUserThread(), wakeUpUserThread(), and PrL1PrL2DramDirectoryMSI::ShmemMsg::WB_REQ.

Referenced by ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork().

6.32.3.26 incrementQBSLookupCost()

```
void CacheCntlr::incrementQBSLookupCost ( )  [virtual]
```

Reimplemented from **CacheCntlr** (p. 197).

Definition at line 1743 of file cache_cntlr.cc.

References ShmemPerfModel::_USER_THREAD, atomic_add_subsecondtime(), ComponentLatency::get↩ Latency(), getMemoryManager(), ParametricDramDirectoryMSI::MemoryManager::incrElapsedTime(), m_↩ writeback_time, and stats.

6.32.3.27 initiateDirectoryAccess()

```
void CacheCntlr::initiateDirectoryAccess (
              Core::mem_op_t mem_op_type,
              IntPtr address,
             bool isPrefetch,
              SubsecondTime t_issue )  [private]
```

Definition at line 1118 of file cache_cntlr.cc.

References ShmemPerfModel::_USER_THREAD, ReqQueueListTemplate< T_Req >::enqueue(), getCache↩ BlockInfo(), CacheBlockInfo::getCState(), getLock(), getShmemPerfModel(), LOG_PRINT_ERROR, m_core_↩ id, ParametricDramDirectoryMSI::CacheMasterCntlr::m_directory_waiters, m_master, m_shmem_perf, MYLOG, processExReqToDirectory(), processShReqToDirectory(), processUpgradeReqToDirectory(), Core::READ, Core↩ ::READ_EX, ShmemPerf::reset(), CacheState::SHARED, ReqQueueListTemplate< T_Req >::size(), and Core::↩ WRITE.

Referenced by processShmemReqFromPrevCache().

6.32.3.28 insertCacheBlock()

```
SharedCacheBlockInfo * CacheCntlr::insertCacheBlock (
            IntPtr address,
            CacheState::cstate_t cstate,
            Byte * data_buf,
            core_id_t requester,
            ShmemPerfModel::Thread_t thread_num )  [private]
```

Definition at line 1365 of file cache_cntlr.cc.

References ShmemPerfModel::_USER_THREAD, accessDRAM(), atomic_add_subsecondtime(), ParametricDramDirectoryMSI::Transition::BACK_INVAL, CacheBlockInfo::BitsUsedOffset, InstMode::CACHE_ONLY, ParametricDramDirectoryMSI::CStateString(), ParametricDramDirectoryMSI::Transition::EVICT, CacheState::EXCLUSIVE, PrL1PrL2DramDirectoryMSI::ShmemMsg::FLUSH_REP, getCacheBlockSize(), getCacheState(), ContentionModel::getCompletionTime(), CacheBlockInfo::getCState(), ShmemPerfModel::getElapsedTime(), getHome(), getLock(), getMemoryManager(), CacheBlockInfo::getOwner(), getShmemPerfModel(), ContentionModel::getStartTime(), CacheBlockInfo::getUsage(), CacheBlockInfo::hasOption(), ParametricDramDirectoryMSI::MemoryManager::incrElapsedTime(), Cache::insertSingleLine(), PrL1PrL2DramDirectoryMSI::ShmemMsg::INV_REP, CacheState::INVALID, MemComponent::LAST_LEVEL_CACHE, LOG_ASSERT_ERROR, LOG_PRINT, ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, m_cache_writethrough, m_coherent, m_core_id, m_core_id_master, ParametricDramDirectoryMSI::CacheMasterCntlr::m_dram_cntlr, ParametricDramDirectoryMSI::CacheMasterCntlr::m_dram_outstanding_writebacks, m_dummy_shmem_perf, ParametricDramDirectoryMSI::CacheMasterCntlr::m_evicting_address, ParametricDramDirectoryMSI::CacheMasterCntlr::m_evicting_buf, m_master, m_mem_component, m_next_cache_cntlr, m_perfect, ParametricDramDirectoryMSI::CacheMasterCntlr::m_prev_cache_cntlrs, CacheState::MODIFIED, MYLOG, notifyPrevLevelEvict(), notifyPrevLevelInsert(), CacheBlockInfo::PREFETCH, ParametricDramDirectoryMSI::MemoryManager::sendMsg(), setCacheState(), CacheBlockInfo::setOption(), CacheBlockInfo::setOwner(), CacheState::SHARED, stats, MemComponent::TAG_DIR, transition(), HitWhere::UNKNOWN, CacheBlockInfo::WARMUP, Core::WRITE, writeCacheBlock(), and SubsecondTime::Zero().

Referenced by copyDataFromNextLevel(), processExRepFromDramDirectory(), processMemOpFromCore(), processShmemReqFromPrevCache(), and processShRepFromDramDirectory().

6.32.3.29 invalidateCacheBlock()

```
void CacheCntlr::invalidateCacheBlock (
            IntPtr address )  [private]
```

Definition at line 1337 of file cache_cntlr.cc.

References __attribute__, ParametricDramDirectoryMSI::CStateString(), getCacheState(), CacheState::INVALID, Cache::invalidateSingleLine(), ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, m_core_id_master, m_master, m_mem_component, m_next_cache_cntlr, CacheState::MODIFIED, MYLOG, and notifyPrevLevelEvict().

Referenced by processMemOpFromCore(), processShmemReqFromPrevCache(), and updateCacheBlock().

6.32.3.30 isFirstLevel()

```
bool ParametricDramDirectoryMSI::CacheCntlr::isFirstLevel (
            void ) [inline]
```

Definition at line 400 of file cache_cntlr.h.

References m_master, and ParametricDramDirectoryMSI::CacheMasterCntlr::m_prev_cache_cntlrs.

Referenced by acquireLock(), and releaseLock().

6.32.3.31 isInLowerLevelCache()

```
bool CacheCntlr::isInLowerLevelCache (
             CacheBlockInfo * block_info ) [virtual]
```

Reimplemented from **CacheCntlr** (p. 197).

Definition at line 1729 of file cache_cntlr.cc.

References CacheBlockInfo::getTag(), ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, m_master, ParametricDramDirectoryMSI::CacheMasterCntlr::m_prev_cache_cntlrs, and CacheBase::tagToAddress().

6.32.3.32 isLastLevel()

```
bool ParametricDramDirectoryMSI::CacheCntlr::isLastLevel (
            void ) [inline]
```

Definition at line 401 of file cache_cntlr.h.

References m_next_cache_cntlr.

6.32.3.33 isMasterCache()

```
bool ParametricDramDirectoryMSI::CacheCntlr::isMasterCache (
            void ) [inline]
```

Definition at line 399 of file cache_cntlr.h.

References m_core_id, and m_core_id_master.

Referenced by CacheCntlr(), and ~CacheCntlr().

6.32.3.34 isShared()

```
bool CacheCntlr::isShared (
            core_id_t core_id )
```

Definition at line 2302 of file cache_cntlr.cc.

References m_core_id_master, and m_shared_cores.

6.32.3.35 lastLevelCache()

```
CacheCntlr * CacheCntlr::lastLevelCache (
           void ) [private]
```

Definition at line 2289 of file cache_cntlr.cc.

References m_last_level, and m_next_cache_cntlr.

Referenced by acquireLock(), acquireStackLock(), releaseLock(), and releaseStackLock().

6.32.3.36 notifyPrevLevelEvict()

```
void CacheCntlr::notifyPrevLevelEvict (
            core_id_t core_id,
            MemComponent::component_t mem_component,
            IntPtr address )
```

Definition at line 1040 of file cache_cntlr.cc.

References ParametricDramDirectoryMSI::CStateString(), getCacheBlockInfo(), getCacheState(), ParametricDramDirectoryMSI::CacheMasterCntlr::m_evicting_address, ParametricDramDirectoryMSI::CacheMasterCntlr::m_evicting_buf, m_master, ParametricDramDirectoryMSI::CacheMasterCntlr::m_prev_cache_cntlrs, and MYLOG.

Referenced by insertCacheBlock(), and invalidateCacheBlock().

6.32.3.37 notifyPrevLevelInsert()

```
void CacheCntlr::notifyPrevLevelInsert (
            core_id_t core_id,
            MemComponent::component_t mem_component,
            IntPtr address )
```

Definition at line 1029 of file cache_cntlr.cc.

References getCacheBlockInfo(), m_master, and ParametricDramDirectoryMSI::CacheMasterCntlr::m_prev_cache_cntlrs.

Referenced by insertCacheBlock().

6.32.3.38 operationPermissibleinCache()

```
bool CacheCntlr::operationPermissibleinCache (
              IntPtr address,
              Core::mem_op_t mem_op_type,
              CacheBlockInfo ** cache_block_info = NULL )  [private]
```

Definition at line 1240 of file cache_cntlr.cc.

References ParametricDramDirectoryMSI::CStateString(), getCacheBlockInfo(), getCacheState(), LOG_PRINT_↩ ERROR, MYLOG, Core::READ, Core::READ_EX, CacheState::readable(), CacheState::writable(), and Core::W↩ RITE.

Referenced by copyDataFromNextLevel(), Prefetch(), processMemOpFromCore(), processShmemReqFromPrev↩ Cache(), and trainPrefetcher().

6.32.3.39 Prefetch()

```
void CacheCntlr::Prefetch (
              SubsecondTime t_start )  [private]
```

Definition at line 688 of file cache_cntlr.cc.

References atomic_add_subsecondtime(), doPrefetch(), getLock(), INVALID_ADDRESS, m_master, m_next↩ _cache_cntlr, ParametricDramDirectoryMSI::CacheMasterCntlr::m_prefetch_list, ParametricDramDirectoryMSI↩ ::CacheMasterCntlr::m_prefetch_next, operationPermissibleinCache(), Prefetch(), PREFETCH_INTERVAL, and Core::READ.

Referenced by Prefetch(), and processMemOpFromCore().

6.32.3.40 printCache()

```
void CacheCntlr::printCache ( )  [private]
```

Definition at line 1796 of file cache_cntlr.cc.

References ParametricDramDirectoryMSI::CStateString(), CacheBase::getAssociativity(), CacheBlockInfo::getC↩ State(), CacheBase::getName(), CacheBase::getNumSets(), CacheState::INVALID, ParametricDramDirectoryM↩ SI::CacheMasterCntlr::m_cache, m_master, and Cache::peekBlock().

Referenced by flush().

6.32.3.41 processExRepFromDramDirectory()

```
void CacheCntlr::processExRepFromDramDirectory (
              core_id_t sender,
              core_id_t requester,
              PrL1PrL2DramDirectoryMSI::ShmemMsg * shmem_msg )  [private]
```

Definition at line 1935 of file cache_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, CacheState::EXCLUSIVE, PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataBuf(), insertCacheBlock(), m_mem_component, and MYLOG.

Referenced by handleMsgFromDramDirectory().

6.32.3.42 processExReqToDirectory()

```
void CacheCntlr::processExReqToDirectory (
              IntPtr address )  [private]
```

Definition at line 1176 of file cache_cntlr.cc.

References ShmemPerfModel::_USER_THREAD, PrL1PrL2DramDirectoryMSI::ShmemMsg::EX_REQ, getCacheState(), getHome(), getMemoryManager(), CacheState::INVALID, MemComponent::LAST_LEVEL_CACHE, LOG_ASSERT_ERROR, m_core_id_master, m_shmem_perf, MYLOG, ParametricDramDirectoryMSI::MemoryManager::sendMsg(), CacheState::SHARED, MemComponent::TAG_DIR, and HitWhere::UNKNOWN.

Referenced by initiateDirectoryAccess().

6.32.3.43 processFlushReqFromDramDirectory()

```
void CacheCntlr::processFlushReqFromDramDirectory (
              core_id_t sender,
              PrL1PrL2DramDirectoryMSI::ShmemMsg * shmem_msg )  [private]
```

Definition at line 2031 of file cache_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, CachePerfModel::ACCESS_CACHE_DATA_AND_TAGS, CachePerfModel::ACCESS_CACHE_TAGS, ParametricDramDirectoryMSI::Transition::COHERENCY, PrL1PrL2DramDirectoryMSI::ShmemMsg::FLUSH_REP, PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), getCacheBlockSize(), getCacheState(), getMemoryManager(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), getShmemPerfModel(), ParametricDramDirectoryMSI::MemoryManager::incrElapsedTime(), CacheState::INVALID, MemComponent::LAST_LEVEL_CACHE, m_mem_component, MYLOG, ShmemPerf::REMOTE_CACHE_WB, ParametricDramDirectoryMSI::MemoryManager::sendMsg(), MemComponent::TAG_DIR, HitWhere::UNKNOWN, updateCacheBlock(), and ShmemPerf::updateTime().

Referenced by handleMsgFromDramDirectory().

6.32.3.44 processInvReqFromDramDirectory()

```
void CacheCntlr::processInvReqFromDramDirectory (
              core_id_t sender,
              PrL1PrL2DramDirectoryMSI::ShmemMsg * shmem_msg )  [private]
```

Definition at line 1992 of file cache_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, CachePerfModel::ACCESS_CACHE_TAGS, ParametricDram↩DirectoryMSI::Transition::COHERENCY, ParametricDramDirectoryMSI::CStateString(), PrL1PrL2DramDirectory↩MSI::ShmemMsg::getAddress(), getCacheState(), getMemoryManager(), PrL1PrL2DramDirectoryMSI::Shmem↩Msg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), getShmemPerfModel(), Parametric↩DramDirectoryMSI::MemoryManager::incrElapsedTime(), PrL1PrL2DramDirectoryMSI::ShmemMsg::INV_REP, CacheState::INVALID, MemComponent::LAST_LEVEL_CACHE, m_mem_component, MYLOG, ShmemPerf↩::REMOTE_CACHE_INV, ParametricDramDirectoryMSI::MemoryManager::sendMsg(), CacheState::SHARED, MemComponent::TAG_DIR, HitWhere::UNKNOWN, updateCacheBlock(), and ShmemPerf::updateTime().

Referenced by handleMsgFromDramDirectory().

6.32.3.45 processMemOpFromCore()

```
HitWhere::where_t CacheCntlr::processMemOpFromCore (
              Core::lock_signal_t lock_signal,
              Core::mem_op_t mem_op_type,
              IntPtr ca_address,
              UInt32 offset,
              Byte * data_buf,
              UInt32 data_length,
             bool modeled,
             bool count )
```

Definition at line 317 of file cache_cntlr.cc.

References ShmemPerfModel::_USER_THREAD, CachePerfModel::ACCESS_CACHE_DATA_AND_TAGS, CachePerfModel::ACCESS_CACHE_TAGS, accessCache(), acquireLock(), acquireStackLock(), InstMode↩::CACHE_ONLY, CacheBlockInfo::clearOption(), copyDataFromNextLevel(), getCache(), getCacheBlockInfo(), getCacheBlockSize(), getCacheState(), ContentionModel::getCompletionTime(), ShmemPerfModel::getElapsed↩Time(), getLock(), getMemoryManager(), SubsecondTime::getNS(), CacheBlockInfo::getOwner(), getShmem↩PerfModel(), ContentionModel::getStartTime(), ContentionModel::getTagCompletionTime(), CacheBlockInfo::get↩Usage(), CacheBlockInfo::hasOption(), HitWhereString(), ShmemPerfModel::incrElapsedTime(), Parametric↩DramDirectoryMSI::MemoryManager::incrElapsedTime(), insertCacheBlock(), CacheState::INVALID, Cache↩BlockInfo::invalidate(), invalidateCacheBlock(), Core::LOCK, LOG_ASSERT_ERROR, LOG_PRINT, LOG_PR↩INT_ERROR, m_cache_writethrough, m_core_id, ParametricDramDirectoryMSI::CacheMasterCntlr::m_l1_mshr, m_l1_mshr, m_master, m_mem_component, m_next_cache_cntlr, m_passthrough, m_perfect, Parametric↩DramDirectoryMSI::CacheMasterCntlr::m_prefetcher, ParametricDramDirectoryMSI::CacheMasterCntlr::m_smt↩_lock, SubsecondTime::MaxTime(), HitWhere::MISS, CacheState::MODIFIED, ParametricDramDirectoryMSI::↩CacheMasterCntlr::mshr, mshr_latency, MYLOG, Core::NONE, ParametricDramDirectoryMSI::Prefetch::NONE, operationPermissibleinCache(), CacheBlockInfo::PREFETCH, Prefetch(), processShmemReqFromPrevCache(), Core::READ, Core::READ_EX, releaseLock(), releaseStackLock(), CacheBlockInfo::setCState(), CacheState↩::SHARED, stats, t_start, total_latency, trainPrefetcher(), Core::UNLOCK, Cache::updateCounters(), update↩Counters(), CacheBlockInfo::updateUsage(), updateUsageBits(), waitForNetworkThread(), wakeUpNetwork↩Thread(), CacheBlockInfo::WARMUP, and Core::WRITE.

Referenced by ParametricDramDirectoryMSI::MemoryManager::coreInitiateMemoryAccess().

6.32.3.46 processShmemReqFromPrevCache()

```
HitWhere::where_t CacheCntlr::processShmemReqFromPrevCache (
              CacheCntlr * requester,
              Core::mem_op_t mem_op_type,
              IntPtr address,
             bool modeled,
             bool count,
              Prefetch::prefetch_type_t isPrefetch,
              SubsecondTime t_issue,
             bool have_write_lock )  [private]
```

Definition at line 757 of file cache_cntlr.cc.

References ShmemPerfModel::_USER_THREAD, CachePerfModel::ACCESS_CACHE_DATA_AND_TAGS, CachePerfModel::ACCESS_CACHE_TAGS, accessDRAM(), acquireStackLock(), atomic_add_subsecondtime(), InstMode::CACHE_ONLY, cleanupMshr(), CacheBlockInfo::clearOption(), ParametricDramDirectoryMSI::Transition::COHERENCY, copyDataFromNextLevel(), CacheState::EXCLUSIVE, getCache(), getCacheBlockInfo(), getCacheBlockSize(), getCacheState(), CacheBlockInfo::getCState(), ShmemPerfModel::getElapsedTime(), getLock(), getMemoryManager(), getShmemPerfModel(), CacheBlockInfo::hasOption(), HitWhereString(), ParametricDramDirectoryMSI::MemoryManager::incrElapsedTime(), initiateDirectoryAccess(), insertCacheBlock(), CacheState::INVALID, SharedCacheBlockInfo::invalidate(), invalidateCacheBlock(), itostr(), LOG_ASSERT_ERROR, m_core_id, ParametricDramDirectoryMSI::CacheMasterCntlr::m_dram_cntlr, m_last_remote_hit_where, m_master, m_mem_component, m_next_cache_cntlr, m_passthrough, m_perfect, ParametricDramDirectoryMSI::CacheMasterCntlr::m_prefetcher, ParametricDramDirectoryMSI::CacheMasterCntlr::m_prev_cache_cntlrs, m_shared_cores, m_shmem_perf, ParametricDramDirectoryMSI::make_mshr(), HitWhere::MISS, CacheState::MODIFIED, ParametricDramDirectoryMSI::CacheMasterCntlr::mshr, MYLOG, ParametricDramDirectoryMSI::Prefetch::NONE, operationPermissibleinCache(), ParametricDramDirectoryMSI::Prefetch::OTHER, CacheBlockInfo::PREFETCH, processShmemReqFromPrevCache(), Core::READ, releaseStackLock(), ShmemPerf::reset(), retrieveCacheBlock(), CacheBlockInfo::setCState(), CacheBlockInfo::setOption(), CacheState::SHARED, HitWhere::SIBLING, stats, trainPrefetcher(), HitWhere::UNKNOWN, Cache::updateCounters(), updateCounters(), updateUncoreStatistics(), ParametricDramDirectoryMSI::Transition::UPGRADE, CacheBlockInfo::WARMUP, and SubsecondTime::Zero().

Referenced by doPrefetch(), processMemOpFromCore(), and processShmemReqFromPrevCache().

6.32.3.47 processShRepFromDramDirectory()

```
void CacheCntlr::processShRepFromDramDirectory (
              core_id_t sender,
              core_id_t requester,
              PrL1PrL2DramDirectoryMSI::ShmemMsg * shmem_msg )  [private]
```

Definition at line 1948 of file cache_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataBuf(), insertCacheBlock(), m_mem_component, MYLOG, and CacheState::SHARED.

Referenced by handleMsgFromDramDirectory().

6.32.3.48 processShReqToDirectory()

```
void CacheCntlr::processShReqToDirectory (
            IntPtr address )  [private]
```

Definition at line 1217 of file cache_cntlr.cc.

References ShmemPerfModel::_USER_THREAD, getHome(), getMemoryManager(), MemComponent::LAST_↩ LEVEL_CACHE, m_core_id_master, m_shmem_perf, MYLOG, ParametricDramDirectoryMSI::MemoryManager↩ ::sendMsg(), PrL1PrL2DramDirectoryMSI::ShmemMsg::SH_REQ, MemComponent::TAG_DIR, and HitWhere::U↩ NKNOWN.

Referenced by initiateDirectoryAccess().

6.32.3.49 processUpgradeRepFromDramDirectory()

```
void CacheCntlr::processUpgradeRepFromDramDirectory (
            core_id_t sender,
            core_id_t requester,
            PrL1PrL2DramDirectoryMSI::ShmemMsg * shmem_msg )  [private]
```

Definition at line 1961 of file cache_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, ParametricDramDirectoryMSI::CStateString(), PrL1PrL2Dram↩ DirectoryMSI::ShmemMsg::getAddress(), getCacheState(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getData↩ Buf(), CacheState::INVALID, LOG_ASSERT_ERROR, LOG_PRINT_ERROR, m_mem_component, Cache↩ State::MODIFIED, MYLOG, CacheState::SHARED_UPGRADING, updateCacheBlock(), and ParametricDram↩ DirectoryMSI::Transition::UPGRADE.

Referenced by handleMsgFromDramDirectory().

6.32.3.50 processUpgradeReqToDirectory()

```
void CacheCntlr::processUpgradeReqToDirectory (
            IntPtr address,
            ShmemPerf * perf,
            ShmemPerfModel::Thread_t thread_num )  [private]
```

Definition at line 1197 of file cache_cntlr.cc.

References getCacheState(), getHome(), getMemoryManager(), MemComponent::LAST_LEVEL_CACHE, m_↩ core_id_master, MYLOG, ParametricDramDirectoryMSI::MemoryManager::sendMsg(), setCacheState(), Cache↩ State::SHARED, CacheState::SHARED_UPGRADING, MemComponent::TAG_DIR, HitWhere::UNKNOWN, and PrL1PrL2DramDirectoryMSI::ShmemMsg::UPGRADE_REQ.

Referenced by handleMsgFromDramDirectory(), and initiateDirectoryAccess().

6.32.3.51 processWbReqFromDramDirectory()

```
void CacheCntlr::processWbReqFromDramDirectory (
              core_id_t sender,
              PrL1PrL2DramDirectoryMSI::ShmemMsg * shmem_msg )  [private]
```

Definition at line 2068 of file cache_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, CachePerfModel::ACCESS_CACHE_DATA_AND_TAGS, CachePerfModel::ACCESS_CACHE_TAGS, ParametricDramDirectoryMSI::Transition::COHERENCY, PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), getCacheBlockSize(), getCacheState(), getMemoryManager(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), getShmemPerfModel(), ParametricDramDirectoryMSI::MemoryManager::incrElapsedTime(), CacheState::INVALID, MemComponent::LAST_LEVEL_CACHE, m_mem_component, MYLOG, ShmemPerf::REMOTE_CACHE_FWD, ShmemPerf::REMOTE_CACHE_WB, ParametricDramDirectoryMSI::MemoryManager::sendMsg(), CacheState::SHARED, CacheState::SHARED_UPGRADING, MemComponent::TAG_DIR, HitWhere::UNKNOWN, updateCacheBlock(), ShmemPerf::updateTime(), and PrL1PrL2DramDirectoryMSI::ShmemMsg::WB_REP.

Referenced by handleMsgFromDramDirectory().

6.32.3.52 releaseLock()

```
void CacheCntlr::releaseLock (
              UInt64 address )
```

Definition at line 2260 of file cache_cntlr.cc.

References ParametricDramDirectoryMSI::CacheMasterCntlr::getSetLock(), isFirstLevel(), lastLevelCache(), m_core_id, m_master, m_mem_component, MYLOG, and _SetLock::release_shared().

Referenced by processMemOpFromCore().

6.32.3.53 releaseStackLock()

```
void CacheCntlr::releaseStackLock (
              UInt64 address,
             bool this_is_locked = false )
```

Definition at line 2279 of file cache_cntlr.cc.

References _SetLock::downgrade(), ParametricDramDirectoryMSI::CacheMasterCntlr::getSetLock(), lastLevelCache(), m_core_id, m_master, m_mem_component, MYLOG, and _SetLock::release_exclusive().

Referenced by doPrefetch(), handleMsgFromDramDirectory(), processMemOpFromCore(), processShmemReqFromPrevCache(), and writeCacheBlock().

6.32.3.54 retrieveCacheBlock()

```
void CacheCntlr::retrieveCacheBlock (
              IntPtr address,
              Byte * data_buf,
              ShmemPerfModel::Thread_t thread_num,
             bool update_replacement )  [private]
```

Definition at line 1352 of file cache_cntlr.cc.

References __attribute__, Cache::accessSingleLine(), getCacheBlockSize(), ShmemPerfModel::getElapsed↩ Time(), getShmemPerfModel(), CacheBase::LOAD, LOG_ASSERT_ERROR, ParametricDramDirectoryMSI::↩ CacheMasterCntlr::m_cache, and m_master.

Referenced by copyDataFromNextLevel(), processShmemReqFromPrevCache(), and updateCacheBlock().

6.32.3.55 setCacheState()

```
SharedCacheBlockInfo * CacheCntlr::setCacheState (
              IntPtr address,
              CacheState::cstate_t cstate )  [private]
```

Definition at line 1329 of file cache_cntlr.cc.

References getCacheBlockInfo(), and CacheBlockInfo::setCState().

Referenced by insertCacheBlock(), and processUpgradeReqToDirectory().

6.32.3.56 setDRAMDirectAccess()

```
void CacheCntlr::setDRAMDirectAccess (
              DramCntlrInterface * dram_cntlr,
              UInt64 num_outstanding )
```

Definition at line 305 of file cache_cntlr.cc.

References m_core_id, ParametricDramDirectoryMSI::CacheMasterCntlr::m_dram_cntlr, ParametricDram↩ DirectoryMSI::CacheMasterCntlr::m_dram_outstanding_writebacks, and m_master.

6.32.3.57 setNextCacheCntlr()

```
void ParametricDramDirectoryMSI::CacheCntlr::setNextCacheCntlr (
              CacheCntlr * next_cache_cntlr )  [inline]
```

Definition at line 374 of file cache_cntlr.h.

References m_next_cache_cntlr.

Referenced by ParametricDramDirectoryMSI::MemoryManager::MemoryManager().

6.32.3.58 setPrevCacheCntlrs()

```
void CacheCntlr::setPrevCacheCntlrs (
              CacheCntlrList & prev_cache_cntlrs )
```

Definition at line 293 of file cache_cntlr.cc.

References LOG_ASSERT_ERROR, m_master, and ParametricDramDirectoryMSI::CacheMasterCntlr::m_prev_↩ cache_cntlrs.

Referenced by ParametricDramDirectoryMSI::MemoryManager::MemoryManager().

6.32.3.59 trainPrefetcher()

```
void CacheCntlr::trainPrefetcher (
              IntPtr address,
             bool cache_hit,
             bool prefetch_hit,
              SubsecondTime t_issue )  [private]
```

Definition at line 662 of file cache_cntlr.cc.

References getLock(), Prefetcher::getNextAddress(), m_core_id, m_master, ParametricDramDirectory↩ MSI::CacheMasterCntlr::m_prefetch_list, ParametricDramDirectoryMSI::CacheMasterCntlr::m_prefetch_next, m_prefetch_on_prefetch_hit, ParametricDramDirectoryMSI::CacheMasterCntlr::m_prefetcher, operation↩ PermissibleinCache(), PREFETCH_INTERVAL, PREFETCH_MAX_QUEUE_LENGTH, and Core::READ.

Referenced by processMemOpFromCore(), and processShmemReqFromPrevCache().

6.32.3.60 transition()

```
void CacheCntlr::transition (
              IntPtr address,
              Transition::reason_t reason,
              CacheState::cstate_t old_state,
              CacheState::cstate_t new_state )  [private]
```

Definition at line 2190 of file cache_cntlr.cc.

References ParametricDramDirectoryMSI::Transition::BACK_INVAL, ParametricDramDirectoryMSI::Transition::C↩ OHERENCY, ParametricDramDirectoryMSI::Transition::EVICT, CacheState::INVALID, CacheState::INVALID_CO↩ HERENCY, CacheState::INVALID_COLD, CacheState::INVALID_EVICT, and stats.

Referenced by insertCacheBlock(), updateCacheBlock(), and updateCounters().

6.32.3.61 updateCacheBlock()

```
std::pair< SubsecondTime, bool > CacheCntlr::updateCacheBlock (
            IntPtr address,
            CacheState::cstate_t cstate,
            Transition::reason_t reason,
            Byte * out_buf,
            ShmemPerfModel::Thread_t thread_num ) [private]
```

Definition at line 1537 of file cache_cntlr.cc.

References __attribute__, ParametricDramDirectoryMSI::Transition::BACK_INVAL, ParametricDramDirectory↩MSI::Transition::COHERENCY, ParametricDramDirectoryMSI::CStateString(), ParametricDramDirectoryMSI::↩Transition::EVICT, getCacheBlockInfo(), getCacheBlockSize(), getCacheState(), CacheBlockInfo::getCState(), ComponentLatency::getLatency(), getLock(), CacheBlockInfo::hasOption(), CacheState::INVALID, invalidate↩CacheBlock(), LOG_ASSERT_ERROR, m_cache_writethrough, m_coherent, m_master, m_next_cache_cntlr, ParametricDramDirectoryMSI::CacheMasterCntlr::m_prev_cache_cntlrs, m_writeback_time, CacheState::MO↩DIFIED, MYLOG, CacheState::NUM_CSTATE_STATES, CacheBlockInfo::PREFETCH, retrieveCacheBlock(), CacheBlockInfo::setCState(), CacheState::SHARED, stats, transition(), ParametricDramDirectoryMSI::Transition↩::UPGRADE, CacheBlockInfo::WARMUP, writeCacheBlock(), and SubsecondTime::Zero().

Referenced by copyDataFromNextLevel(), processFlushReqFromDramDirectory(), processInvReqFromDram↩Directory(), processUpgradeRepFromDramDirectory(), and processWbReqFromDramDirectory().

6.32.3.62 updateCounters()

```
void CacheCntlr::updateCounters (
            Core::mem_op_t mem_op_type,
            IntPtr address,
            bool cache_hit,
            CacheState::cstate_t state,
            Prefetch::prefetch_type_t isPrefetch ) [private]
```

Definition at line 2113 of file cache_cntlr.cc.

References ShmemPerfModel::_USER_THREAD, ParametricDramDirectoryMSI::CacheMasterCntlr::access↩ATDs(), cleanupMshr(), ParametricDramDirectoryMSI::Transition::CORE_RD, ParametricDramDirectoryMSI::↩Transition::CORE_RDEX, ParametricDramDirectoryMSI::Transition::CORE_WR, ShmemPerfModel::getElapsed↩Time(), getShmemPerfModel(), CacheState::INVALID, m_core_id, m_core_id_master, m_master, CacheState::M↩ODIFIED, ParametricDramDirectoryMSI::CacheMasterCntlr::mshr, ParametricDramDirectoryMSI::Prefetch::NONE, ParametricDramDirectoryMSI::Prefetch::OWN, Core::READ, Core::READ_EX, CacheState::SHARED, stats, transition(), and Core::WRITE.

Referenced by processMemOpFromCore(), processShmemReqFromPrevCache(), and updateHits().

6.32.3.63 updateHits()

```
void CacheCntlr::updateHits (
            Core::mem_op_t mem_op_type,
            UInt64 hits )
```

Definition at line 608 of file cache_cntlr.cc.

References getCache(), getLock(), CacheState::MODIFIED, ParametricDramDirectoryMSI::Prefetch::NONE, Core::READ, CacheState::SHARED, Cache::updateCounters(), and updateCounters().

6.32.3.64 updateUncoreStatistics()

```
void CacheCntlr::updateUncoreStatistics (
              HitWhere::where_t hit_where,
              SubsecondTime now )  [private]
```

Definition at line 2208 of file cache_cntlr.cc.

References ShmemPerf::add(), ShmemPerf::disable(), ShmemPerf::getInitialTime(), INVALID_CORE_ID, m_last↩
_remote_hit_where, m_shmem_perf, m_shmem_perf_global, m_shmem_perf_numrequests, m_shmem_perf_↩
totaltime, ShmemPerf::reset(), and SubsecondTime::Zero().

Referenced by handleMsgFromDramDirectory(), and processShmemReqFromPrevCache().

6.32.3.65 updateUsageBits()

```
void CacheCntlr::updateUsageBits (
              IntPtr address,
              CacheBlockInfo::BitsUsedType used )  [private]
```

Definition at line 1057 of file cache_cntlr.cc.

References getCacheBlockInfo(), getLock(), m_next_cache_cntlr, m_perfect, CacheBlockInfo::updateUsage(), and updateUsageBits().

Referenced by processMemOpFromCore(), and updateUsageBits().

6.32.3.66 waitForNetworkThread()

```
void CacheCntlr::waitForNetworkThread (
             void )  [private]
```

Definition at line 2320 of file cache_cntlr.cc.

References m_user_thread_sem, and Semaphore::wait().

Referenced by doPrefetch(), and processMemOpFromCore().

6.32.3.67 waitForUserThread()

```
void CacheCntlr::waitForUserThread (
              Semaphore * network_thread_sem = NULL )  [private]
```

Definition at line 2315 of file cache_cntlr.cc.

References m_network_thread_sem.

Referenced by handleMsgFromDramDirectory().

6.32.3.68 wakeUpNetworkThread()

```
void CacheCntlr::wakeUpNetworkThread (
            void ) [private]
```

Definition at line 2325 of file cache_cntlr.cc.

References m_network_thread_sem, and Semaphore::signal().

Referenced by doPrefetch(), and processMemOpFromCore().

6.32.3.69 wakeUpUserThread()

```
void CacheCntlr::wakeUpUserThread (
            Semaphore * user_thread_sem = NULL ) [private]
```

Definition at line 2310 of file cache_cntlr.cc.

References m_user_thread_sem.

Referenced by handleMsgFromDramDirectory().

6.32.3.70 walkUsageBits()

```
void CacheCntlr::walkUsageBits ( ) [private]
```

Definition at line 1072 of file cache_cntlr.cc.

References CacheBlockInfo::BitsUsedOffset, CacheBase::getAssociativity(), getCacheBlockSize(), CacheBase::getNumSets(), CacheBlockInfo::getOwner(), CacheBlockInfo::getUsage(), CacheBlockInfo::hasOption(), CacheBlockInfo::isValid(), ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, m_master, m_next_cache_cntlr, Cache::peekBlock(), and CacheBlockInfo::WARMUP.

6.32.3.71 writeCacheBlock()

```
void CacheCntlr::writeCacheBlock (
            IntPtr address,
            UInt32 offset,
            Byte * data_buf,
            UInt32 data_length,
            ShmemPerfModel::Thread_t thread_num ) [private]
```

Definition at line 1702 of file cache_cntlr.cc.

References __attribute__, Cache::accessSingleLine(), acquireStackLock(), getCacheBlockSize(), CacheBlockInfo::getCState(), ShmemPerfModel::getElapsedTime(), getShmemPerfModel(), LOG_ASSERT_ERROR, ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache, m_cache_writethrough, ParametricDramDirectoryMSI::CacheMasterCntlr::m_evicting_address, ParametricDramDirectoryMSI::CacheMasterCntlr::m_evicting_buf, m_master, m_next_cache_cntlr, CacheState::MODIFIED, MYLOG, releaseStackLock(), CacheBase::STORE, and writeCacheBlock().

Referenced by accessCache(), insertCacheBlock(), updateCacheBlock(), and writeCacheBlock().

6.32.4 Friends And Related Function Documentation

6.32.4.1 CacheCntlrList

```
friend class CacheCntlrList [friend]
```

Definition at line 412 of file cache_cntlr.h.

6.32.4.2 MemoryManager

```
friend class MemoryManager [friend]
```

Definition at line 413 of file cache_cntlr.h.

6.32.5 Member Data Documentation

6.32.5.1 backinval

```
UInt64 ParametricDramDirectoryMSI::CacheCntlr::backinval[ CacheState::NUM_CSTATE_STATES]
```

Definition at line 234 of file cache_cntlr.h.

6.32.5.2 coherency_downgrades

```
UInt64 ParametricDramDirectoryMSI::CacheCntlr::coherency_downgrades
```

Definition at line 241 of file cache_cntlr.h.

6.32.5.3 coherency_invalidates

```
UInt64 ParametricDramDirectoryMSI::CacheCntlr::coherency_invalidates
```

Definition at line 241 of file cache_cntlr.h.

6.32.5.4 coherency_upgrades

UInt64 ParametricDramDirectoryMSI::CacheCntlr::coherency_upgrades

Definition at line 241 of file cache_cntlr.h.

6.32.5.5 coherency_writebacks

UInt64 ParametricDramDirectoryMSI::CacheCntlr::coherency_writebacks

Definition at line 241 of file cache_cntlr.h.

6.32.5.6 evict

UInt64 ParametricDramDirectoryMSI::CacheCntlr::evict[**CacheState::NUM_CSTATE_STATES**]

Definition at line 233 of file cache_cntlr.h.

6.32.5.7 evict_prefetch

UInt64 ParametricDramDirectoryMSI::CacheCntlr::evict_prefetch

Definition at line 227 of file cache_cntlr.h.

6.32.5.8 evict_warmup

UInt64 ParametricDramDirectoryMSI::CacheCntlr::evict_warmup

Definition at line 235 of file cache_cntlr.h.

6.32.5.9 hits_prefetch

UInt64 ParametricDramDirectoryMSI::CacheCntlr::hits_prefetch

Definition at line 226 of file cache_cntlr.h.

6.32.5.10 hits_warmup

UInt64 ParametricDramDirectoryMSI::CacheCntlr::hits_warmup

Definition at line 235 of file cache_cntlr.h.

6.32.5.11 invalidate_prefetch

UInt64 ParametricDramDirectoryMSI::CacheCntlr::invalidate_prefetch

Definition at line 228 of file cache_cntlr.h.

6.32.5.12 invalidate_warmup

UInt64 ParametricDramDirectoryMSI::CacheCntlr::invalidate_warmup

Definition at line 235 of file cache_cntlr.h.

6.32.5.13 load_misses

UInt64 ParametricDramDirectoryMSI::CacheCntlr::load_misses

Definition at line 220 of file cache_cntlr.h.

6.32.5.14 load_misses_state

UInt64 ParametricDramDirectoryMSI::CacheCntlr::load_misses_state[**CacheState::NUM_CSTATE_ST↩ ATES**]

Definition at line 224 of file cache_cntlr.h.

6.32.5.15 load_overlapping_misses

UInt64 ParametricDramDirectoryMSI::CacheCntlr::load_overlapping_misses

Definition at line 221 of file cache_cntlr.h.

6.32.5.16 loads

UInt64 ParametricDramDirectoryMSI::CacheCntlr::loads

Definition at line 219 of file cache_cntlr.h.

6.32.5.17 loads_prefetch

UInt64 ParametricDramDirectoryMSI::CacheCntlr::loads_prefetch

Definition at line 225 of file cache_cntlr.h.

6.32.5.18 loads_state

UInt64 ParametricDramDirectoryMSI::CacheCntlr::loads_state[**CacheState::NUM_CSTATE_STATES**]

Definition at line 222 of file cache_cntlr.h.

6.32.5.19 loads_where

UInt64 ParametricDramDirectoryMSI::CacheCntlr::loads_where[**HitWhere::NUM_HITWHERES**]

Definition at line 223 of file cache_cntlr.h.

6.32.5.20 m_cache_block_size

UInt32 ParametricDramDirectoryMSI::CacheCntlr::m_cache_block_size [private]

Definition at line 258 of file cache_cntlr.h.

Referenced by CacheCntlr(), and getCacheBlockSize().

6.32.5.21 m_cache_writethrough

bool ParametricDramDirectoryMSI::CacheCntlr::m_cache_writethrough [private]

Definition at line 259 of file cache_cntlr.h.

Referenced by accessCache(), insertCacheBlock(), processMemOpFromCore(), updateCacheBlock(), and write↩ CacheBlock().

6.32.5.22 m_coherent

```
bool ParametricDramDirectoryMSI::CacheCntlr::m_coherent  [private]
```

Definition at line 214 of file cache_cntlr.h.

Referenced by insertCacheBlock(), and updateCacheBlock().

6.32.5.23 m_core_id

```
core_id_t ParametricDramDirectoryMSI::CacheCntlr::m_core_id  [private]
```

Definition at line 257 of file cache_cntlr.h.

Referenced by acquireLock(), acquireStackLock(), CacheCntlr(), copyDataFromNextLevel(), flushBlock(), handle↩ MsgFromDramDirectory(), initiateDirectoryAccess(), insertCacheBlock(), isMasterCache(), processMemOpFrom↩ Core(), processShmemReqFromPrevCache(), releaseLock(), releaseStackLock(), setDRAMDirectAccess(), train↩ Prefetcher(), updateCounters(), and ∼CacheCntlr().

6.32.5.24 m_core_id_master

```
core_id_t ParametricDramDirectoryMSI::CacheCntlr::m_core_id_master  [private]
```

Core (p. 377) id of the 'master' (actual) cache controller we're proxying

Definition at line 264 of file cache_cntlr.h.

Referenced by accessDRAM(), CacheCntlr(), insertCacheBlock(), invalidateCacheBlock(), isMasterCache(), isShared(), processExReqToDirectory(), processShReqToDirectory(), processUpgradeReqToDirectory(), and updateCounters().

6.32.5.25 m_dummy_shmem_perf

```
ShmemPerf ParametricDramDirectoryMSI::CacheCntlr::m_dummy_shmem_perf  [private]
```

Definition at line 274 of file cache_cntlr.h.

Referenced by flushBlock(), and insertCacheBlock().

6.32.5.26 m_l1_mshr

```
bool ParametricDramDirectoryMSI::CacheCntlr::m_l1_mshr  [private]
```

Definition at line 216 of file cache_cntlr.h.

Referenced by processMemOpFromCore().

6.32.5.27 m_last_level

```
CacheCntlr* ParametricDramDirectoryMSI::CacheCntlr::m_last_level  [private]
```

Definition at line 209 of file cache_cntlr.h.

Referenced by lastLevelCache().

6.32.5.28 m_last_remote_hit_where

```
volatile HitWhere::where_t ParametricDramDirectoryMSI::CacheCntlr::m_last_remote_hit_where
[private]
```

Definition at line 268 of file cache_cntlr.h.

Referenced by processShmemReqFromPrevCache(), and updateUncoreStatistics().

6.32.5.29 m_master

```
CacheMasterCntlr* ParametricDramDirectoryMSI::CacheCntlr::m_master  [private]
```

Definition at line 207 of file cache_cntlr.h.

Referenced by accessCache(), accessDRAM(), acquireLock(), acquireStackLock(), CacheCntlr(), cleanupMshr(), copyDataFromNextLevel(), createSetLocks(), disable(), enable(), flush(), flushBlock(), getCache(), getCache↩BlockInfo(), getLock(), handleMsgFromDramDirectory(), initiateDirectoryAccess(), insertCacheBlock(), invalidate↩CacheBlock(), isFirstLevel(), isInLowerLevelCache(), notifyPrevLevelEvict(), notifyPrevLevelInsert(), Prefetch(), printCache(), processMemOpFromCore(), processShmemReqFromPrevCache(), releaseLock(), releaseStack↩Lock(), retrieveCacheBlock(), setDRAMDirectAccess(), setPrevCacheCntlrs(), trainPrefetcher(), updateCache↩Block(), updateCounters(), walkUsageBits(), writeCacheBlock(), and ∼CacheCntlr().

6.32.5.30 m_mem_component

MemComponent::component_t ParametricDramDirectoryMSI::CacheCntlr::m_mem_component [private]

Definition at line 205 of file cache_cntlr.h.

Referenced by acquireLock(), acquireStackLock(), copyDataFromNextLevel(), insertCacheBlock(), invalidateCacheBlock(), processExRepFromDramDirectory(), processFlushReqFromDramDirectory(), processInvReqFromDramDirectory(), processMemOpFromCore(), processShmemReqFromPrevCache(), processShRepFromDramDirectory(), processUpgradeRepFromDramDirectory(), processWbReqFromDramDirectory(), releaseLock(), and releaseStackLock().

6.32.5.31 m_memory_manager

MemoryManager* ParametricDramDirectoryMSI::CacheCntlr::m_memory_manager [private]

Definition at line 206 of file cache_cntlr.h.

Referenced by getMemoryManager().

6.32.5.32 m_network_thread_sem

Semaphore* ParametricDramDirectoryMSI::CacheCntlr::m_network_thread_sem [private]

Definition at line 267 of file cache_cntlr.h.

Referenced by getNetworkThreadSemaphore(), handleMsgFromDramDirectory(), waitForUserThread(), and wakeUpNetworkThread().

6.32.5.33 m_next_cache_cntlr

CacheCntlr* ParametricDramDirectoryMSI::CacheCntlr::m_next_cache_cntlr [private]

Definition at line 208 of file cache_cntlr.h.

Referenced by accessCache(), copyDataFromNextLevel(), insertCacheBlock(), invalidateCacheBlock(), isLastLevel(), lastLevelCache(), Prefetch(), processMemOpFromCore(), processShmemReqFromPrevCache(), setNextCacheCntlr(), updateCacheBlock(), updateUsageBits(), walkUsageBits(), and writeCacheBlock().

6.32.5.34 m_next_level_read_bandwidth

ComponentBandwidthPerCycle ParametricDramDirectoryMSI::CacheCntlr::m_next_level_read_bandwidth [private]

Definition at line 261 of file cache_cntlr.h.

Referenced by copyDataFromNextLevel().

6.32.5.35 m_passthrough

bool ParametricDramDirectoryMSI::CacheCntlr::m_passthrough [private]

Definition at line 213 of file cache_cntlr.h.

Referenced by CacheCntlr(), processMemOpFromCore(), and processShmemReqFromPrevCache().

6.32.5.36 m_perfect

bool ParametricDramDirectoryMSI::CacheCntlr::m_perfect [private]

Definition at line 212 of file cache_cntlr.h.

Referenced by insertCacheBlock(), processMemOpFromCore(), processShmemReqFromPrevCache(), and updateUsageBits().

6.32.5.37 m_prefetch_on_prefetch_hit

bool ParametricDramDirectoryMSI::CacheCntlr::m_prefetch_on_prefetch_hit [private]

Definition at line 215 of file cache_cntlr.h.

Referenced by CacheCntlr(), and trainPrefetcher().

6.32.5.38 m_shared_cores

UInt32 ParametricDramDirectoryMSI::CacheCntlr::m_shared_cores [private]

Number of cores this cache is shared with

Definition at line 263 of file cache_cntlr.h.

Referenced by CacheCntlr(), isShared(), and processShmemReqFromPrevCache().

6.32.5.39 m_shmem_perf

ShmemPerf* ParametricDramDirectoryMSI::CacheCntlr::m_shmem_perf [private]

Definition at line 270 of file cache_cntlr.h.

Referenced by accessDRAM(), handleMsgFromDramDirectory(), initiateDirectoryAccess(), processExReqTo↩ Directory(), processShmemReqFromPrevCache(), processShReqToDirectory(), updateUncoreStatistics(), and ∼↩ CacheCntlr().

6.32.5.40 m_shmem_perf_global

ShmemPerf* ParametricDramDirectoryMSI::CacheCntlr::m_shmem_perf_global [private]

Definition at line 271 of file cache_cntlr.h.

Referenced by CacheCntlr(), updateUncoreStatistics(), and ∼CacheCntlr().

6.32.5.41 m_shmem_perf_model

ShmemPerfModel* ParametricDramDirectoryMSI::CacheCntlr::m_shmem_perf_model [private]

Definition at line 276 of file cache_cntlr.h.

Referenced by getShmemPerfModel().

6.32.5.42 m_shmem_perf_numrequests

UInt64 ParametricDramDirectoryMSI::CacheCntlr::m_shmem_perf_numrequests [private]

Definition at line 273 of file cache_cntlr.h.

Referenced by CacheCntlr(), and updateUncoreStatistics().

6.32.5.43 m_shmem_perf_totaltime

SubsecondTime ParametricDramDirectoryMSI::CacheCntlr::m_shmem_perf_totaltime [private]

Definition at line 272 of file cache_cntlr.h.

Referenced by CacheCntlr(), and updateUncoreStatistics().

6.32.5.44 m_shmem_req_source_map

```
std::unordered_map< IntPtr, MemComponent::component_t> ParametricDramDirectoryMSI::Cache↩
Cntlr::m_shmem_req_source_map [private]
```

Definition at line 211 of file cache_cntlr.h.

6.32.5.45 m_tag_directory_home_lookup

```
AddressHomeLookup* ParametricDramDirectoryMSI::CacheCntlr::m_tag_directory_home_lookup [private]
```

Definition at line 210 of file cache_cntlr.h.

Referenced by getHome().

6.32.5.46 m_user_thread_sem

```
Semaphore* ParametricDramDirectoryMSI::CacheCntlr::m_user_thread_sem [private]
```

Definition at line 266 of file cache_cntlr.h.

Referenced by getUserThreadSemaphore(), handleMsgFromDramDirectory(), waitForNetworkThread(), and wakeUpUserThread().

6.32.5.47 m_writeback_time

```
ComponentLatency ParametricDramDirectoryMSI::CacheCntlr::m_writeback_time [private]
```

Definition at line 260 of file cache_cntlr.h.

Referenced by incrementQBSLookupCost(), and updateCacheBlock().

6.32.5.48 mshr_latency

```
SubsecondTime ParametricDramDirectoryMSI::CacheCntlr::mshr_latency
```

Definition at line 239 of file cache_cntlr.h.

Referenced by processMemOpFromCore().

6.32.5.49 prefetches

UInt64 ParametricDramDirectoryMSI::CacheCntlr::prefetches

Definition at line 240 of file cache_cntlr.h.

6.32.5.50 qbs_query_latency

SubsecondTime ParametricDramDirectoryMSI::CacheCntlr::qbs_query_latency

Definition at line 238 of file cache_cntlr.h.

6.32.5.51 snoop_latency

SubsecondTime ParametricDramDirectoryMSI::CacheCntlr::snoop_latency

Definition at line 237 of file cache_cntlr.h.

6.32.5.52 stats

struct { ... } ParametricDramDirectoryMSI::CacheCntlr::stats [private]

Referenced by CacheCntlr(), doPrefetch(), incrementQBSLookupCost(), insertCacheBlock(), processMemOp↩FromCore(), processShmemReqFromPrevCache(), transition(), updateCacheBlock(), and updateCounters().

6.32.5.53 store_misses

UInt64 ParametricDramDirectoryMSI::CacheCntlr::store_misses

Definition at line 220 of file cache_cntlr.h.

6.32.5.54 store_misses_state

UInt64 ParametricDramDirectoryMSI::CacheCntlr::store_misses_state[**CacheState::NUM_CSTATE_S↩TATES**]

Definition at line 224 of file cache_cntlr.h.

6.32.5.55 store_overlapping_misses

UInt64 ParametricDramDirectoryMSI::CacheCntlr::store_overlapping_misses

Definition at line 221 of file cache_cntlr.h.

6.32.5.56 stores

UInt64 ParametricDramDirectoryMSI::CacheCntlr::stores

Definition at line 219 of file cache_cntlr.h.

6.32.5.57 stores_prefetch

UInt64 ParametricDramDirectoryMSI::CacheCntlr::stores_prefetch

Definition at line 225 of file cache_cntlr.h.

6.32.5.58 stores_state

UInt64 ParametricDramDirectoryMSI::CacheCntlr::stores_state[**CacheState::NUM_CSTATE_STATES**]

Definition at line 222 of file cache_cntlr.h.

6.32.5.59 stores_where

UInt64 ParametricDramDirectoryMSI::CacheCntlr::stores_where[**HitWhere::NUM_HITWHERES**]

Definition at line 223 of file cache_cntlr.h.

6.32.5.60 total_latency

SubsecondTime ParametricDramDirectoryMSI::CacheCntlr::total_latency

Definition at line 236 of file cache_cntlr.h.

Referenced by processMemOpFromCore().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_cntlr.h**
- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_cntlr.cc**

6.33 CacheCntlr Class Reference

`#include <cache_block_info.h>`

Inheritance diagram for CacheCntlr:

Public Member Functions

- virtual bool **isInLowerLevelCache** (**CacheBlockInfo** ∗block_info)
- virtual void **incrementQBSLookupCost** ()
- virtual void **flush** ()

6.33.1 Detailed Description

Definition at line 65 of file cache_block_info.h.

6.33.2 Member Function Documentation

6.33.2.1 flush()

`virtual void CacheCntlr::flush ( ) [inline], [virtual]`

Reimplemented in **ParametricDramDirectoryMSI::CacheCntlr** (p. 166).

Definition at line 70 of file cache_block_info.h.

Referenced by CacheSetKruger::getReplacementIndex().

6.33.2.2 incrementQBSLookupCost()

`virtual void CacheCntlr::incrementQBSLookupCost ( ) [inline], [virtual]`

Reimplemented in **ParametricDramDirectoryMSI::CacheCntlr** (p. 170).

Definition at line 69 of file cache_block_info.h.

Referenced by CacheSetSRRIP::getReplacementIndex(), and CacheSetLRU::getReplacementIndex().

6.33.2.3 isInLowerLevelCache()

```
virtual bool CacheCntlr::isInLowerLevelCache (
              CacheBlockInfo * block_info )  [inline], [virtual]
```

Reimplemented in **ParametricDramDirectoryMSI::CacheCntlr** (p. 172).

Definition at line 68 of file cache_block_info.h.

Referenced by CacheSetSRRIP::getReplacementIndex(), and CacheSetLRU::getReplacementIndex().

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/cache/ **cache_block_info.h**

6.34 ParametricDramDirectoryMSI::CacheCntlrList Class Reference

```
#include <cache_cntlr.h>
```

Inheritance diagram for ParametricDramDirectoryMSI::CacheCntlrList:

6.34.1 Detailed Description

Definition at line 122 of file cache_cntlr.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_cntlr.h**

6.35 ParametricDramDirectoryMSI::CacheDirectoryWaiter Class Reference

```
#include <cache_cntlr.h>
```

Public Member Functions

- **CacheDirectoryWaiter** (bool _exclusive, bool _isPrefetch, **CacheCntlr** *_cache_cntlr, **SubsecondTime** _t_issue)

Public Attributes

- bool **exclusive**
- bool **isPrefetch**
- **CacheCntlr** * **cache_cntlr**
- **SubsecondTime** **t_issue**

6.35.1 Detailed Description

Definition at line 130 of file cache_cntlr.h.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 CacheDirectoryWaiter()

```
ParametricDramDirectoryMSI::CacheDirectoryWaiter::CacheDirectoryWaiter (
            bool _exclusive,
            bool _isPrefetch,
             CacheCntlr * _cache_cntlr,
             SubsecondTime _t_issue )  [inline]
```

Definition at line 137 of file cache_cntlr.h.

6.35.3 Member Data Documentation

6.35.3.1 cache_cntlr

```
CacheCntlr* ParametricDramDirectoryMSI::CacheDirectoryWaiter::cache_cntlr
```

Definition at line 135 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory().

6.35.3.2 exclusive

```
bool ParametricDramDirectoryMSI::CacheDirectoryWaiter::exclusive
```

Definition at line 133 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory().

6.35.3.3 isPrefetch

```
bool ParametricDramDirectoryMSI::CacheDirectoryWaiter::isPrefetch
```

Definition at line 134 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory().

6.35.3.4 t_issue

```
SubsecondTime ParametricDramDirectoryMSI::CacheDirectoryWaiter::t_issue
```

Definition at line 136 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory().

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_cntlr.h**

6.36 FastNehalem::CacheLocked< assoc, size_kb > Class Template Reference

```
#include <fast_cache.h>
```

Inheritance diagram for FastNehalem::CacheLocked< assoc, size_kb >:

Public Member Functions

- **CacheLocked** (**Core** *core, String name, **MemComponent::component_t** mem_component, **UInt64** latency, **CacheBase** *next_level)
- **SubsecondTime access** (**Core::mem_op_t** mem_op_type, **IntPtr** tag)

Private Attributes

- **Lock lock**

6.36.1 Detailed Description

template<UInt32 assoc, UInt32 size_kb>
class FastNehalem::CacheLocked< assoc, size_kb >

Definition at line 113 of file fast_cache.h.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 CacheLocked()

```
template<UInt32 assoc, UInt32 size_kb>
FastNehalem::CacheLocked< assoc, size_kb >::  CacheLocked (
              Core * core,
             String name,
              MemComponent::component_t mem_component,
              UInt64 latency,
              CacheBase * next_level )  [inline]
```

Definition at line 118 of file fast_cache.h.

6.36.3 Member Function Documentation

6.36.3.1 access()

```
template<UInt32 assoc, UInt32 size_kb>
SubsecondTime  FastNehalem::CacheLocked< assoc, size_kb >::access (
              Core::mem_op_t mem_op_type,
              IntPtr tag )  [inline], [virtual]
```

Reimplemented from **FastNehalem::Cache< assoc, size_kb >** (p. 139).

Definition at line 121 of file fast_cache.h.

References FastNehalem::Cache< assoc, size_kb >::access(), and FastNehalem::CacheLocked< assoc, size_kb >::lock.

6.36.4 Member Data Documentation

6.36.4.1 lock

```
template<UInt32 assoc, UInt32 size_kb>
Lock FastNehalem::CacheLocked< assoc, size_kb >::lock [private]
```

Definition at line 116 of file fast_cache.h.

Referenced by FastNehalem::CacheLocked< assoc, size_kb >::access().

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/fast_nehalem/ **fast_cache.h**

6.37 ParametricDramDirectoryMSI::CacheMasterCntlr Class Reference

```
#include <cache_cntlr.h>
```

Private Member Functions

- void **createSetLocks** (**UInt32** cache_block_size, **UInt32** num_sets, **UInt32** core_offset, **UInt32** num_↩ cores)
- **SetLock** ∗ **getSetLock** (**IntPtr** addr)
- void **createATDs** (String name, String configName, **core_id_t** core_id, **UInt32** shared_cores, **UInt32** size, **UInt32** associativity, **UInt32** block_size, String replacement_policy, **CacheBase::hash_t** hash_function)
- void **accessATDs** (**Core::mem_op_t** mem_op_type, bool hit, **IntPtr** address, **UInt32** core_num)
- **CacheMasterCntlr** (String name, **core_id_t** core_id, **UInt32** outstanding_misses)
- ∼**CacheMasterCntlr** ()

Private Attributes

- **Cache** ∗ **m_cache**
- **Lock m_cache_lock**
- **Lock m_smt_lock**
- **CacheCntlrList m_prev_cache_cntlrs**
- **Prefetcher** ∗ **m_prefetcher**
- **DramCntlrInterface** ∗ **m_dram_cntlr**
- **ContentionModel** ∗ **m_dram_outstanding_writebacks**
- **Mshr mshr**
- **ContentionModel m_l1_mshr**
- **ContentionModel m_next_level_read_bandwidth**
- **CacheDirectoryWaiterMap m_directory_waiters**
- **IntPtr m_evicting_address**
- **Byte** ∗ **m_evicting_buf**
- std::vector< **ATD** ∗ > **m_atds**
- std::vector< **SetLock** > **m_setlocks**
- **UInt32 m_log_blocksize**
- **UInt32 m_num_sets**
- std::deque< **IntPtr** > **m_prefetch_list**
- **SubsecondTime m_prefetch_next**

Friends

- class **CacheCntlr**

6.37.1 Detailed Description

Definition at line 149 of file cache_cntlr.h.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 CacheMasterCntlr()

```
ParametricDramDirectoryMSI::CacheMasterCntlr::CacheMasterCntlr (
            String name,
             core_id_t core_id,
             UInt32 outstanding_misses )  [inline], [private]
```

Definition at line 183 of file cache_cntlr.h.

6.37.2.2 ~CacheMasterCntlr()

```
ParametricDramDirectoryMSI::CacheMasterCntlr::~CacheMasterCntlr ( )  [private]
```

Definition at line 115 of file cache_cntlr.cc.

References m_atds, and m_cache.

6.37.3 Member Function Documentation

6.37.3.1 accessATDs()

```
void ParametricDramDirectoryMSI::CacheMasterCntlr::accessATDs (
             Core::mem_op_t mem_op_type,
            bool hit,
             IntPtr address,
             UInt32 core_num )  [private]
```

Definition at line 109 of file cache_cntlr.cc.

References m_atds.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::updateCounters().

6.37.3.2 createATDs()

```
void ParametricDramDirectoryMSI::CacheMasterCntlr::createATDs (
            String name,
            String configName,
             core_id_t core_id,
             UInt32 shared_cores,
             UInt32 size,
             UInt32 associativity,
             UInt32 block_size,
            String replacement_policy,
             CacheBase::hash_t hash_function )  [private]
```

Definition at line 98 of file cache_cntlr.cc.

References m_atds.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr().

6.37.3.3 createSetLocks()

```
void ParametricDramDirectoryMSI::CacheMasterCntlr::createSetLocks (
             UInt32 cache_block_size,
             UInt32 num_sets,
             UInt32 core_offset,
             UInt32 num_cores )  [private]
```

Definition at line 84 of file cache_cntlr.cc.

References floorLog2(), m_log_blocksize, m_num_sets, and m_setlocks.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::createSetLocks().

6.37.3.4 getSetLock()

```
 SetLock * ParametricDramDirectoryMSI::CacheMasterCntlr::getSetLock (
             IntPtr addr )  [private]
```

Definition at line 92 of file cache_cntlr.cc.

References m_log_blocksize, m_num_sets, and m_setlocks.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::acquireLock(), ParametricDramDirectoryMSI::Cache↩ Cntlr::acquireStackLock(), ParametricDramDirectoryMSI::CacheCntlr::releaseLock(), and ParametricDram↩ DirectoryMSI::CacheCntlr::releaseStackLock().

6.37.4 Friends And Related Function Documentation

6.37.4.1 CacheCntlr

```
friend class CacheCntlr [friend]
```

Definition at line 198 of file cache_cntlr.h.

6.37.5 Member Data Documentation

6.37.5.1 m_atds

```
std::vector< ATD*> ParametricDramDirectoryMSI::CacheMasterCntlr::m_atds [private]
```

Definition at line 167 of file cache_cntlr.h.

Referenced by accessATDs(), createATDs(), and ~CacheMasterCntlr().

6.37.5.2 m_cache

```
Cache* ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache [private]
```

Definition at line 152 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::accessCache(), ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), ParametricDramDirectoryMSI::CacheCntlr::disable(), ParametricDramDirectoryMSI::CacheCntlr::enable(), ParametricDramDirectoryMSI::CacheCntlr::flush(), ParametricDramDirectoryMSI::CacheCntlr::flushBlock(), ParametricDramDirectoryMSI::CacheCntlr::getCache(), ParametricDramDirectoryMSI::CacheCntlr::getCacheBlockInfo(), ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), ParametricDramDirectoryMSI::CacheCntlr::invalidateCacheBlock(), ParametricDramDirectoryMSI::CacheCntlr::isInLowerLevelCache(), ParametricDramDirectoryMSI::CacheCntlr::printCache(), ParametricDramDirectoryMSI::CacheCntlr::retrieveCacheBlock(), ParametricDramDirectoryMSI::CacheCntlr::walkUsageBits(), ParametricDramDirectoryMSI::CacheCntlr::writeCacheBlock(), and ~CacheMasterCntlr().

6.37.5.3 m_cache_lock

```
Lock ParametricDramDirectoryMSI::CacheMasterCntlr::m_cache_lock [private]
```

Definition at line 153 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::getLock().

6.37.5.4 m_directory_waiters

CacheDirectoryWaiterMap ParametricDramDirectoryMSI::CacheMasterCntlr::m_directory_waiters [private]

Definition at line 163 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), and ParametricDramDirectoryMSI::CacheCntlr::initiateDirectoryAccess().

6.37.5.5 m_dram_cntlr

DramCntlrInterface* ParametricDramDirectoryMSI::CacheMasterCntlr::m_dram_cntlr [private]

Definition at line 157 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::accessDRAM(), ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache(), and ParametricDramDirectoryMSI::CacheCntlr::setDRAMDirectAccess().

6.37.5.6 m_dram_outstanding_writebacks

ContentionModel* ParametricDramDirectoryMSI::CacheMasterCntlr::m_dram_outstanding_writebacks [private]

Definition at line 158 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), and ParametricDramDirectoryMSI::CacheCntlr::setDRAMDirectAccess().

6.37.5.7 m_evicting_address

IntPtr ParametricDramDirectoryMSI::CacheMasterCntlr::m_evicting_address [private]

Definition at line 164 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), ParametricDramDirectoryMSI::CacheCntlr::notifyPrevLevelEvict(), and ParametricDramDirectoryMSI::CacheCntlr::writeCacheBlock().

6.37.5.8 m_evicting_buf

Byte* ParametricDramDirectoryMSI::CacheMasterCntlr::m_evicting_buf [private]

Definition at line 165 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), ParametricDramDirectoryMSI::↩ CacheCntlr::notifyPrevLevelEvict(), and ParametricDramDirectoryMSI::CacheCntlr::writeCacheBlock().

6.37.5.9 m_l1_mshr

ContentionModel ParametricDramDirectoryMSI::CacheMasterCntlr::m_l1_mshr [private]

Definition at line 161 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore().

6.37.5.10 m_log_blocksize

UInt32 ParametricDramDirectoryMSI::CacheMasterCntlr::m_log_blocksize [private]

Definition at line 170 of file cache_cntlr.h.

Referenced by createSetLocks(), and getSetLock().

6.37.5.11 m_next_level_read_bandwidth

ContentionModel ParametricDramDirectoryMSI::CacheMasterCntlr::m_next_level_read_bandwidth [private]

Definition at line 162 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::copyDataFromNextLevel().

6.37.5.12 m_num_sets

UInt32 ParametricDramDirectoryMSI::CacheMasterCntlr::m_num_sets [private]

Definition at line 171 of file cache_cntlr.h.

Referenced by createSetLocks(), and getSetLock().

6.37.5.13 m_prefetch_list

```
std::deque< IntPtr> ParametricDramDirectoryMSI::CacheMasterCntlr::m_prefetch_list [private]
```

Definition at line 173 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::Prefetch(), and ParametricDramDirectoryMSI::Cache↩ Cntlr::trainPrefetcher().

6.37.5.14 m_prefetch_next

```
SubsecondTime ParametricDramDirectoryMSI::CacheMasterCntlr::m_prefetch_next [private]
```

Definition at line 174 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::Prefetch(), and ParametricDramDirectoryMSI::Cache↩ Cntlr::trainPrefetcher().

6.37.5.15 m_prefetcher

```
Prefetcher* ParametricDramDirectoryMSI::CacheMasterCntlr::m_prefetcher [private]
```

Definition at line 156 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), ParametricDramDirectoryMSI::Cache↩ Cntlr::processMemOpFromCore(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrev↩ Cache(), and ParametricDramDirectoryMSI::CacheCntlr::trainPrefetcher().

6.37.5.16 m_prev_cache_cntlrs

```
CacheCntlrList ParametricDramDirectoryMSI::CacheMasterCntlr::m_prev_cache_cntlrs [private]
```

Definition at line 155 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), ParametricDramDirectoryMSI::↩ CacheCntlr::isFirstLevel(), ParametricDramDirectoryMSI::CacheCntlr::isInLowerLevelCache(), ParametricDram↩ DirectoryMSI::CacheCntlr::notifyPrevLevelEvict(), ParametricDramDirectoryMSI::CacheCntlr::notifyPrevLevel↩ Insert(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache(), ParametricDram↩ DirectoryMSI::CacheCntlr::setPrevCacheCntlrs(), and ParametricDramDirectoryMSI::CacheCntlr::updateCache↩ Block().

6.37.5.17 m_setlocks

```
std::vector< SetLock> ParametricDramDirectoryMSI::CacheMasterCntlr::m_setlocks [private]
```

Definition at line 169 of file cache_cntlr.h.

Referenced by createSetLocks(), and getSetLock().

6.37.5.18 m_smt_lock

```
Lock ParametricDramDirectoryMSI::CacheMasterCntlr::m_smt_lock [private]
```

Definition at line 154 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore().

6.37.5.19 mshr

```
Mshr ParametricDramDirectoryMSI::CacheMasterCntlr::mshr [private]
```

Definition at line 160 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::cleanupMshr(), ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache(), and ParametricDramDirectoryMSI::CacheCntlr::updateCounters().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_cntlr.h**
- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_cntlr.cc**

6.38 ParametricDramDirectoryMSI::CacheParameters Class Reference

```
#include <cache_cntlr.h>
```

Public Member Functions

- **CacheParameters** ()
- **CacheParameters** (String _configName, **UInt32** _size, **UInt32** _associativity, **UInt32** block_size, String _hash_function, String _replacement_policy, bool _perfect, bool _coherent, const **ComponentLatency** &_data_access_time, const **ComponentLatency** &_tags_access_time, const **ComponentLatency** &_writeback_time, const **ComponentBandwidthPerCycle** &_next_level_read_bandwidth, String _perf_model_type, bool _writethrough, **UInt32** _shared_cores, String _prefetcher, **UInt32** _outstanding_misses)

Public Attributes

- String **configName**
- **UInt32 size**
- **UInt32 num_sets**
- **UInt32 associativity**
- String **hash_function**
- String **replacement_policy**
- bool **perfect**
- bool **coherent**
- **ComponentLatency data_access_time**
- **ComponentLatency tags_access_time**
- **ComponentLatency writeback_time**
- **ComponentBandwidthPerCycle next_level_read_bandwidth**
- String **perf_model_type**
- bool **writethrough**
- **UInt32 shared_cores**
- String **prefetcher**
- **UInt32 outstanding_misses**

6.38.1 Detailed Description

Definition at line 76 of file cache_cntlr.h.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 CacheParameters() [1/2]

```
ParametricDramDirectoryMSI::CacheParameters::CacheParameters ( )  [inline]
```

Definition at line 97 of file cache_cntlr.h.

6.38.2.2 CacheParameters() [2/2]

```
ParametricDramDirectoryMSI::CacheParameters::CacheParameters (
            String _configName,
             UInt32 _size,
             UInt32 _associativity,
             UInt32 block_size,
            String _hash_function,
            String _replacement_policy,
            bool _perfect,
            bool _coherent,
            const  ComponentLatency & _data_access_time,
            const  ComponentLatency & _tags_access_time,
            const  ComponentLatency & _writeback_time,
```

```
const  ComponentBandwidthPerCycle & _next_level_read_bandwidth,
String _perf_model_type,
bool _writethrough,
 UInt32 _shared_cores,
String _prefetcher,
 UInt32 _outstanding_misses )  [inline]
```

Definition at line 102 of file cache_cntlr.h.

References associativity, k_KILO, LOG_ASSERT_ERROR, and num_sets.

6.38.3 Member Data Documentation

6.38.3.1 associativity

```
UInt32 ParametricDramDirectoryMSI::CacheParameters::associativity
```

Definition at line 82 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), CacheParameters(), and NucaCache::↩ NucaCache().

6.38.3.2 coherent

```
bool ParametricDramDirectoryMSI::CacheParameters::coherent
```

Definition at line 86 of file cache_cntlr.h.

6.38.3.3 configName

```
String ParametricDramDirectoryMSI::CacheParameters::configName
```

Definition at line 79 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr().

6.38.3.4 data_access_time

```
ComponentLatency ParametricDramDirectoryMSI::CacheParameters::data_access_time
```

Definition at line 87 of file cache_cntlr.h.

6.38.3.5 hash_function

String ParametricDramDirectoryMSI::CacheParameters::hash_function

Definition at line 83 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), and NucaCache::NucaCache().

6.38.3.6 next_level_read_bandwidth

ComponentBandwidthPerCycle ParametricDramDirectoryMSI::CacheParameters::next_level_read_↩ bandwidth

Definition at line 90 of file cache_cntlr.h.

6.38.3.7 num_sets

UInt32 ParametricDramDirectoryMSI::CacheParameters::num_sets

Definition at line 81 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), CacheParameters(), and NucaCache::↩ NucaCache().

6.38.3.8 outstanding_misses

UInt32 ParametricDramDirectoryMSI::CacheParameters::outstanding_misses

Definition at line 95 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr().

6.38.3.9 perf_model_type

String ParametricDramDirectoryMSI::CacheParameters::perf_model_type

Definition at line 91 of file cache_cntlr.h.

6.38.3.10 perfect

```
bool ParametricDramDirectoryMSI::CacheParameters::perfect
```

Definition at line 85 of file cache_cntlr.h.

6.38.3.11 prefetcher

```
String ParametricDramDirectoryMSI::CacheParameters::prefetcher
```

Definition at line 94 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr().

6.38.3.12 replacement_policy

```
String ParametricDramDirectoryMSI::CacheParameters::replacement_policy
```

Definition at line 84 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), and NucaCache::NucaCache().

6.38.3.13 shared_cores

```
UInt32 ParametricDramDirectoryMSI::CacheParameters::shared_cores
```

Definition at line 93 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::MemoryManager::MemoryManager().

6.38.3.14 size

```
UInt32 ParametricDramDirectoryMSI::CacheParameters::size
```

Definition at line 80 of file cache_cntlr.h.

6.38.3.15 tags_access_time

```
ComponentLatency ParametricDramDirectoryMSI::CacheParameters::tags_access_time
```

Definition at line 88 of file cache_cntlr.h.

6.38.3.16 writeback_time

```
ComponentLatency ParametricDramDirectoryMSI::CacheParameters::writeback_time
```

Definition at line 89 of file cache_cntlr.h.

6.38.3.17 writethrough

```
bool ParametricDramDirectoryMSI::CacheParameters::writethrough
```

Definition at line 92 of file cache_cntlr.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_cntlr.h**

6.39 CachePerfModel Class Reference

```
#include <cache_perf_model.h>
```

Inheritance diagram for CachePerfModel:

Public Types

- enum **CacheAccess_t** { **ACCESS_CACHE_DATA_AND_TAGS** = 0, **ACCESS_CACHE_DATA**, **ACCE↩ SS_CACHE_TAGS**, **NUM_CACHE_ACCESS_TYPES** }
- enum **PerfModel_t** { **CACHE_PERF_MODEL_PARALLEL** = 0, **CACHE_PERF_MODEL_SEQUENTIAL**, **NUM_CACHE_PERF_MODELS** }

Public Member Functions

- **CachePerfModel** (const **ComponentLatency** &cache_data_access_time, const **ComponentLatency** &cache_tags_access_time)
- virtual ~**CachePerfModel** ()
- virtual void **enable** ()=0
- virtual void **disable** ()=0
- virtual bool **isEnabled** ()=0
- virtual **SubsecondTime** **getLatency** (**CacheAccess_t** access)=0

Static Public Member Functions

- static **CachePerfModel** * **create** (String cache_perf_model_type, const **ComponentLatency** &cache_↩ data_access_time, const **ComponentLatency** &cache_tags_access_time)
- static **PerfModel_t** **parseModelType** (String model_type)

Protected Attributes

- **ComponentLatency** **m_cache_data_access_time**
- **ComponentLatency** **m_cache_tags_access_time**

6.39.1 Detailed Description

Definition at line 7 of file cache_perf_model.h.

6.39.2 Member Enumeration Documentation

6.39.2.1 CacheAccess_t

enum **CachePerfModel::CacheAccess_t**

Enumerator

ACCESS_CACHE_DATA_AND_TAGS	
ACCESS_CACHE_DATA	
ACCESS_CACHE_TAGS	
NUM_CACHE_ACCESS_TYPES	

Definition at line 10 of file cache_perf_model.h.

6.39.2.2 PerfModel_t

enum **CachePerfModel::PerfModel_t**

Enumerator

CACHE_PERF_MODEL_PARALLEL	
CACHE_PERF_MODEL_SEQUENTIAL	
NUM_CACHE_PERF_MODELS	

Definition at line 18 of file cache_perf_model.h.

6.39.3 Constructor & Destructor Documentation

6.39.3.1 CachePerfModel()

```
CachePerfModel::CachePerfModel (
            const  ComponentLatency & cache_data_access_time,
            const  ComponentLatency & cache_tags_access_time )
```

Definition at line 6 of file cache_perf_model.cc.

6.39.3.2 ~CachePerfModel()

```
CachePerfModel::~CachePerfModel ( )  [virtual]
```

Definition at line 11 of file cache_perf_model.cc.

6.39.4 Member Function Documentation

6.39.4.1 create()

```
CachePerfModel * CachePerfModel::create (
            String cache_perf_model_type,
            const  ComponentLatency & cache_data_access_time,
            const  ComponentLatency & cache_tags_access_time )  [static]
```

Definition at line 15 of file cache_perf_model.cc.

References CACHE_PERF_MODEL_PARALLEL, CACHE_PERF_MODEL_SEQUENTIAL, LOG_ASSERT_ER↩ ROR, and parseModelType().

Referenced by ParametricDramDirectoryMSI::MemoryManager::MemoryManager().

6.39.4.2 disable()

```
virtual void CachePerfModel::disable ( )  [pure virtual]
```

Implemented in **CachePerfModelParallel** (p. 219), and **CachePerfModelSequential** (p. 222).

6.39.4.3 enable()

```
virtual void CachePerfModel::enable ( )  [pure virtual]
```

Implemented in **CachePerfModelParallel** (p. 219), and **CachePerfModelSequential** (p. 222).

6.39.4.4 getLatency()

```
virtual  SubsecondTime CachePerfModel::getLatency (
             CacheAccess_t access )  [pure virtual]
```

Implemented in **CachePerfModelParallel** (p. 220), and **CachePerfModelSequential** (p. 222).

Referenced by ParametricDramDirectoryMSI::MemoryManager::getCost(), and ParametricDramDirectoryMSI::↩ MemoryManager::getL1HitLatency().

6.39.4.5 isEnabled()

```
virtual bool CachePerfModel::isEnabled ( )  [pure virtual]
```

Implemented in **CachePerfModelParallel** (p. 220), and **CachePerfModelSequential** (p. 222).

6.39.4.6 parseModelType()

```
CachePerfModel::PerfModel_t CachePerfModel::parseModelType (
            String model_type )  [static]
```

Definition at line 35 of file cache_perf_model.cc.

References CACHE_PERF_MODEL_PARALLEL, CACHE_PERF_MODEL_SEQUENTIAL, LOG_PRINT_ERROR, and NUM_CACHE_PERF_MODELS.

Referenced by create().

6.39.5 Member Data Documentation

6.39.5.1 m_cache_data_access_time

ComponentLatency CachePerfModel::m_cache_data_access_time [protected]

Definition at line 26 of file cache_perf_model.h.

Referenced by CachePerfModelParallel::getLatency(), and CachePerfModelSequential::getLatency().

6.39.5.2 m_cache_tags_access_time

ComponentLatency CachePerfModel::m_cache_tags_access_time [protected]

Definition at line 27 of file cache_perf_model.h.

Referenced by CachePerfModelParallel::getLatency(), and CachePerfModelSequential::getLatency().

The documentation for this class was generated from the following files:

- common/performance_model/ **cache_perf_model.h**
- common/performance_model/ **cache_perf_model.cc**

6.40 CachePerfModelParallel Class Reference

`#include <cache_perf_model_parallel.h>`

Inheritance diagram for CachePerfModelParallel:

Public Member Functions

- **CachePerfModelParallel** (const **ComponentLatency** &cache_data_access_time, const **ComponentLatency** &cache_tags_access_time)
- ~**CachePerfModelParallel** ()
- void **enable** ()
- void **disable** ()
- bool **isEnabled** ()
- **SubsecondTime** **getLatency** (**CacheAccess_t** access)

Private Attributes

- bool **m_enabled**

Additional Inherited Members

6.40.1 Detailed Description

Definition at line 6 of file cache_perf_model_parallel.h.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 CachePerfModelParallel()

```
CachePerfModelParallel::CachePerfModelParallel (
            const  ComponentLatency & cache_data_access_time,
            const  ComponentLatency & cache_tags_access_time )  [inline]
```

Definition at line 12 of file cache_perf_model_parallel.h.

6.40.2.2 ~CachePerfModelParallel()

```
CachePerfModelParallel::~CachePerfModelParallel ( )  [inline]
```

Definition at line 17 of file cache_perf_model_parallel.h.

6.40.3 Member Function Documentation

6.40.3.1 disable()

```
void CachePerfModelParallel::disable ( )  [inline], [virtual]
```

Implements **CachePerfModel** (p. 216).

Definition at line 20 of file cache_perf_model_parallel.h.

References m_enabled.

6.40.3.2 enable()

```
void CachePerfModelParallel::enable ( )  [inline], [virtual]
```

Implements **CachePerfModel** (p. 217).

Definition at line 19 of file cache_perf_model_parallel.h.

References m_enabled.

6.40.3.3 getLatency()

```
SubsecondTime CachePerfModelParallel::getLatency (
            CacheAccess_t access )  [inline], [virtual]
```

Implements **CachePerfModel** (p. 217).

Definition at line 23 of file cache_perf_model_parallel.h.

References CachePerfModel::ACCESS_CACHE_DATA, CachePerfModel::ACCESS_CACHE_DATA_AND_TAGS, CachePerfModel::ACCESS_CACHE_TAGS, ComponentLatency::getLatency(), CachePerfModel::m_cache_data_access_time, CachePerfModel::m_cache_tags_access_time, m_enabled, and SubsecondTime::Zero().

6.40.3.4 isEnabled()

```
bool CachePerfModelParallel::isEnabled ( )  [inline], [virtual]
```

Implements **CachePerfModel** (p. 217).

Definition at line 21 of file cache_perf_model_parallel.h.

References m_enabled.

6.40.4 Member Data Documentation

6.40.4.1 m_enabled

```
bool CachePerfModelParallel::m_enabled  [private]
```

Definition at line 9 of file cache_perf_model_parallel.h.

Referenced by disable(), enable(), getLatency(), and isEnabled().

The documentation for this class was generated from the following file:

- common/performance_model/ **cache_perf_model_parallel.h**

6.41 CachePerfModelSequential Class Reference

`#include <cache_perf_model_sequential.h>`

Inheritance diagram for CachePerfModelSequential:

Public Member Functions

- **CachePerfModelSequential** (const **ComponentLatency** &cache_data_access_time, const **ComponentLatency** &cache_tags_access_time)
- ~**CachePerfModelSequential** ()
- void **enable** ()
- void **disable** ()
- bool **isEnabled** ()
- **SubsecondTime** **getLatency** (**CacheAccess_t** access)

Private Attributes

- bool **m_enabled**

Additional Inherited Members

6.41.1 Detailed Description

Definition at line 6 of file cache_perf_model_sequential.h.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 CachePerfModelSequential()

```
CachePerfModelSequential::CachePerfModelSequential (
            const  ComponentLatency & cache_data_access_time,
            const  ComponentLatency & cache_tags_access_time )  [inline]
```

Definition at line 12 of file cache_perf_model_sequential.h.

6.41.2.2 ~CachePerfModelSequential()

```
CachePerfModelSequential::~CachePerfModelSequential ( )  [inline]
```

Definition at line 17 of file cache_perf_model_sequential.h.

6.41.3 Member Function Documentation

6.41.3.1 disable()

```
void CachePerfModelSequential::disable ( )  [inline], [virtual]
```

Implements **CachePerfModel** (p. 216).

Definition at line 20 of file cache_perf_model_sequential.h.

References m_enabled.

6.41.3.2 enable()

```
void CachePerfModelSequential::enable ( )  [inline], [virtual]
```

Implements **CachePerfModel** (p. 217).

Definition at line 19 of file cache_perf_model_sequential.h.

References m_enabled.

6.41.3.3 getLatency()

```
SubsecondTime CachePerfModelSequential::getLatency (
            CacheAccess_t access )  [inline], [virtual]
```

Implements **CachePerfModel** (p. 217).

Definition at line 23 of file cache_perf_model_sequential.h.

References CachePerfModel::ACCESS_CACHE_DATA, CachePerfModel::ACCESS_CACHE_DATA_AND_TAGS, CachePerfModel::ACCESS_CACHE_TAGS, ComponentLatency::getLatency(), CachePerfModel::m_cache_data_access_time, CachePerfModel::m_cache_tags_access_time, m_enabled, and SubsecondTime::Zero().

6.41.3.4 isEnabled()

```
bool CachePerfModelSequential::isEnabled ( )  [inline], [virtual]
```

Implements **CachePerfModel** (p. 217).

Definition at line 21 of file cache_perf_model_sequential.h.

References m_enabled.

6.41.4 Member Data Documentation

6.41.4.1 m_enabled

```
bool CachePerfModelSequential::m_enabled  [private]
```

Definition at line 9 of file cache_perf_model_sequential.h.

Referenced by disable(), enable(), getLatency(), and isEnabled().

The documentation for this class was generated from the following file:

- common/performance_model/ **cache_perf_model_sequential.h**

6.42 CacheSet Class Reference

```
#include <cache_set.h>
```

Inheritance diagram for CacheSet:

Public Member Functions

- **CacheSet** (**CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize)
- virtual ~**CacheSet** ()
- **UInt32** **getBlockSize** ()
- **UInt32** **getAssociativity** ()
- **Lock** & **getLock** ()
- void **read_line** (**UInt32** line_index, **UInt32** offset, **Byte** *out_buff, **UInt32** bytes, bool update_replacement)
- void **write_line** (**UInt32** line_index, **UInt32** offset, **Byte** *in_buff, **UInt32** bytes, bool update_replacement)
- **CacheBlockInfo** * **find** (**IntPtr** tag, **UInt32** *line_index=NULL)
- bool **invalidate** (**IntPtr** &tag)
- void **insert** (**CacheBlockInfo** *cache_block_info, **Byte** *fill_buff, bool *eviction, **CacheBlockInfo** *evict↩ _block_info, **Byte** *evict_buff, **CacheCntlr** *cntlr=NULL)
- **CacheBlockInfo** * **peekBlock** (**UInt32** way) const
- char * **getDataPtr** (**UInt32** line_index, **UInt32** offset=0)
- **UInt32** **getBlockSize** (void) const
- virtual **UInt32** **getReplacementIndex** (**CacheCntlr** *cntlr)=0
- virtual void **updateReplacementIndex** (**UInt32**)=0
- bool **isValidReplacement** (**UInt32** index)

Static Public Member Functions

- static **CacheSet** * **createCacheSet** (String cfgname, **core_id_t** core_id, String replacement_policy, **CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize, **CacheSetInfo** *set_info=N↩ ULL)
- static **CacheSetInfo** * **createCacheSetInfo** (String name, String cfgname, **core_id_t** core_id, String replacement_policy, **UInt32** associativity)
- static **CacheBase::ReplacementPolicy** **parsePolicyType** (String policy)
- static **UInt8** **getNumQBSAttempts** (**CacheBase::ReplacementPolicy**, String cfgname, **core_id_t** core↩ _id)

Protected Attributes

- **CacheBlockInfo** ** **m_cache_block_info_array**
- char * **m_blocks**
- **UInt32** **m_associativity**
- **UInt32** **m_blocksize**
- **Lock** **m_lock**

6.42.1 Detailed Description

Definition at line 22 of file cache_set.h.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 CacheSet()

```
CacheSet::CacheSet (
            CacheBase::cache_t cache_type,
            UInt32 associativity,
            UInt32 blocksize )
```

Definition at line 17 of file cache_set.cc.

References CacheBlockInfo::create(), m_associativity, m_blocks, m_blocksize, and m_cache_block_info_array.

6.42.2.2 ~CacheSet()

```
CacheSet::~CacheSet ( )  [virtual]
```

Definition at line 36 of file cache_set.cc.

References m_associativity, m_blocks, and m_cache_block_info_array.

6.42.3 Member Function Documentation

6.42.3.1 createCacheSet()

```
CacheSet * CacheSet::createCacheSet (
            String cfgname,
            core_id_t core_id,
            String replacement_policy,
            CacheBase::cache_t cache_type,
            UInt32 associativity,
            UInt32 blocksize,
            CacheSetInfo * set_info = NULL )  [static]
```

Definition at line 138 of file cache_set.cc.

References getNumQBSAttempts(), CacheBase::KRUGER, LOG_PRINT_ERROR, CacheBase::LRU, CacheBase::LRU_QBS, CacheBase::MRU, CacheBase::NMRU, CacheBase::NRU, parsePolicyType(), CacheBase::PLRU, CacheBase::RANDOM, CacheBase::ROUND_ROBIN, CacheBase::SRRIP, and CacheBase::SRRIP_QBS.

Referenced by ATD::ATD(), and Cache::Cache().

6.42.3.2 createCacheSetInfo()

```
CacheSetInfo * CacheSet::createCacheSetInfo (
            String name,
            String cfgname,
             core_id_t core_id,
            String replacement_policy,
             UInt32 associativity )  [static]
```

Definition at line 187 of file cache_set.cc.

References getNumQBSAttempts(), CacheBase::KRUGER, CacheBase::LRU, CacheBase::LRU_QBS, parse↩ PolicyType(), CacheBase::SRRIP, and CacheBase::SRRIP_QBS.

Referenced by ATD::ATD(), and Cache::Cache().

6.42.3.3 find()

```
CacheBlockInfo * CacheSet::find (
             IntPtr tag,
             UInt32 * line_index = NULL )
```

Definition at line 71 of file cache_set.cc.

References m_associativity, and m_cache_block_info_array.

Referenced by Cache::accessSingleLine(), Cache::insertSingleLine(), and Cache::peekSingleLine().

6.42.3.4 getAssociativity()

```
UInt32 CacheSet::getAssociativity ( )  [inline]
```

Definition at line 45 of file cache_set.h.

References m_associativity.

6.42.3.5 getBlockSize() [1/2]

```
UInt32 CacheSet::getBlockSize ( )  [inline]
```

Definition at line 44 of file cache_set.h.

References m_blocksize.

6.42.3.6 getBlockSize() [2/2]

```
UInt32 CacheSet::getBlockSize (
            void  ) const  [inline]
```

Definition at line 57 of file cache_set.h.

References m_blocksize.

6.42.3.7 getDataPtr()

```
char * CacheSet::getDataPtr (
             UInt32 line_index,
             UInt32 offset = 0 )
```

Definition at line 132 of file cache_set.cc.

References m_blocks, and m_blocksize.

6.42.3.8 getLock()

```
Lock& CacheSet::getLock ( )  [inline]
```

Definition at line 46 of file cache_set.h.

References m_lock.

Referenced by Cache::getSetLock().

6.42.3.9 getNumQBSAttempts()

```
UInt8 CacheSet::getNumQBSAttempts (
             CacheBase::ReplacementPolicy policy,
            String cfgname,
             core_id_t core_id )  [static]
```

Definition at line 205 of file cache_set.cc.

References CacheBase::LRU_QBS, and CacheBase::SRRIP_QBS.

Referenced by createCacheSet(), and createCacheSetInfo().

6.42.3.10 getReplacementIndex()

```
virtual UInt32 CacheSet::getReplacementIndex (
              CacheCntlr * cntlr )  [pure virtual]
```

Implemented in **CacheSetLRU** (p. 243), **CacheSetSRRIP** (p. 259), **CacheSetKruger** (p. 239), **CacheSetMRU** (p. 246), **CacheSetNMRU** (p. 248), **CacheSetNRU** (p. 250), **CacheSetPLRU** (p. 253), **CacheSetRandom** (p. 255), **CacheSetRoundRobin** (p. 257), and **CacheSetKruger** (p. 239).

Referenced by insert().

6.42.3.11 insert()

```
void CacheSet::insert (
              CacheBlockInfo * cache_block_info,
              Byte * fill_buff,
             bool * eviction,
              CacheBlockInfo * evict_block_info,
              Byte * evict_buff,
              CacheCntlr * cntlr = NULL )
```

Definition at line 100 of file cache_set.cc.

References CacheBlockInfo::clone(), getReplacementIndex(), m_associativity, m_blocks, m_blocksize, and m_cache_block_info_array.

Referenced by Cache::insertSingleLine().

6.42.3.12 invalidate()

```
bool CacheSet::invalidate (
              IntPtr & tag )
```

Definition at line 86 of file cache_set.cc.

References CacheBlockInfo::invalidate(), m_associativity, and m_cache_block_info_array.

Referenced by Cache::invalidateSingleLine().

6.42.3.13 isValidReplacement()

```
bool CacheSet::isValidReplacement (
              UInt32 index )
```

Definition at line 249 of file cache_set.cc.

References m_cache_block_info_array, and CacheState::SHARED_UPGRADING.

Referenced by CacheSetMRU::getReplacementIndex(), CacheSetNRU::getReplacementIndex(), CacheSetRoundRobin::getReplacementIndex(), CacheSetRandom::getReplacementIndex(), CacheSetPLRU::getReplacementIndex(), CacheSetNMRU::getReplacementIndex(), CacheSetSRRIP::getReplacementIndex(), and CacheSetLRU::getReplacementIndex().

6.42.3.14 parsePolicyType()

```
CacheBase::ReplacementPolicy CacheSet::parsePolicyType (
            String policy )  [static]
```

Definition at line 219 of file cache_set.cc.

References CacheBase::KRUGER, LOG_PRINT_ERROR, CacheBase::LRU, CacheBase::LRU_QBS, CacheBase::MRU, CacheBase::NMRU, CacheBase::NRU, CacheBase::PLRU, CacheBase::RANDOM, CacheBase::ROUND_ROBIN, CacheBase::SRRIP, and CacheBase::SRRIP_QBS.

Referenced by createCacheSet(), and createCacheSetInfo().

6.42.3.15 peekBlock()

```
CacheBlockInfo* CacheSet::peekBlock (
             UInt32 way ) const  [inline]
```

Definition at line 54 of file cache_set.h.

References m_cache_block_info_array.

Referenced by Cache::peekBlock().

6.42.3.16 read_line()

```
void CacheSet::read_line (
              UInt32 line_index,
              UInt32 offset,
              Byte * out_buff,
              UInt32 bytes,
             bool update_replacement )
```

Definition at line 45 of file cache_set.cc.

References m_blocks, m_blocksize, and updateReplacementIndex().

Referenced by Cache::accessSingleLine().

6.42.3.17 updateReplacementIndex()

```
virtual void CacheSet::updateReplacementIndex (
              UInt32  )  [pure virtual]
```

Implemented in **CacheSetLRU** (p. 244), **CacheSetSRRIP** (p. 259), **CacheSetKruger** (p. 240), **CacheSetMRU** (p. 246), **CacheSetNMRU** (p. 248), **CacheSetNRU** (p. 251), **CacheSetPLRU** (p. 253), **CacheSetRandom** (p. 255), **CacheSetRoundRobin** (p. 257), and **CacheSetKruger** (p. 240).

Referenced by read_line(), and write_line().

6.42.3.18 write_line()

```
void CacheSet::write_line (
              UInt32 line_index,
              UInt32 offset,
              Byte * in_buff,
              UInt32 bytes,
             bool update_replacement )
```

Definition at line 58 of file cache_set.cc.

References m_blocks, m_blocksize, and updateReplacementIndex().

Referenced by Cache::accessSingleLine().

6.42.4 Member Data Documentation

6.42.4.1 m_associativity

```
UInt32 CacheSet::m_associativity  [protected]
```

Definition at line 34 of file cache_set.h.

Referenced by CacheSet(), CacheSetKruger::CacheSetKruger(), CacheSetLRU::CacheSetLRU(), CacheSetMRU::CacheSetMRU(), CacheSetNMRU::CacheSetNMRU(), CacheSetNRU::CacheSetNRU(), CacheSetRoundRobin::CacheSetRoundRobin(), CacheSetSRRIP::CacheSetSRRIP(), find(), getAssociativity(), CacheSetNMRU::getReplacementIndex(), CacheSetNRU::getReplacementIndex(), CacheSetPLRU::getReplacementIndex(), CacheSetRandom::getReplacementIndex(), CacheSetRoundRobin::getReplacementIndex(), CacheSetMRU::getReplacementIndex(), CacheSetKruger::getReplacementIndex(), CacheSetSRRIP::getReplacementIndex(), CacheSetLRU::getReplacementIndex(), insert(), invalidate(), CacheSetKruger::moveToMRU(), CacheSetLRU::moveToMRU(), CacheSetKruger::printBlockStats(), CacheSetNMRU::updateReplacementIndex(), CacheSetPLRU::updateReplacementIndex(), CacheSetNRU::updateReplacementIndex(), CacheSetMRU::updateReplacementIndex(), and ~CacheSet().

6.42.4.2 m_blocks

```
char* CacheSet::m_blocks  [protected]
```

Definition at line 33 of file cache_set.h.

Referenced by CacheSet(), getDataPtr(), insert(), read_line(), write_line(), and ~CacheSet().

6.42.4.3 m_blocksize

```
UInt32 CacheSet::m_blocksize [protected]
```

Definition at line 35 of file cache_set.h.

Referenced by CacheSet(), getBlockSize(), getDataPtr(), insert(), read_line(), and write_line().

6.42.4.4 m_cache_block_info_array

```
CacheBlockInfo** CacheSet::m_cache_block_info_array [protected]
```

Definition at line 32 of file cache_set.h.

Referenced by CacheSet(), find(), CacheSetNRU::getReplacementIndex(), CacheSetPLRU::getReplacement↩Index(), CacheSetRandom::getReplacementIndex(), CacheSetMRU::getReplacementIndex(), CacheSetNMR↩U::getReplacementIndex(), CacheSetKruger::getReplacementIndex(), CacheSetSRRIP::getReplacementIndex(), CacheSetLRU::getReplacementIndex(), CacheSetKruger::injectTest(), insert(), invalidate(), CacheSetKruger::is↩ValidReplacement(), isValidReplacement(), CacheSetKruger::isValidReplacement2(), peekBlock(), CacheSet↩Kruger::printBlockStats(), and ~CacheSet().

6.42.4.5 m_lock

```
Lock CacheSet::m_lock [protected]
```

Definition at line 36 of file cache_set.h.

Referenced by getLock().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_set.h**
- common/core/memory_subsystem/cache/ **cache_set.cc**

6.43 FastNehalem::CacheSet< assoc > Class Template Reference

`#include <fast_cache.h>`

Public Member Functions

- bool **find** (**IntPtr** tag)

Private Attributes

- **IntPtr m_tags** [assoc]
- **UInt64 m_lru** [assoc]
- **UInt64 m_lru_max**

6.43.1 Detailed Description

template<UInt32 assoc>
class FastNehalem::CacheSet< assoc >

Definition at line 36 of file fast_cache.h.

6.43.2 Member Function Documentation

6.43.2.1 find()

```
template<UInt32 assoc>
bool  FastNehalem::CacheSet< assoc >::find (
              IntPtr tag )  [inline]
```

Definition at line 43 of file fast_cache.h.

References FastNehalem::CacheSet< assoc >::m_lru, FastNehalem::CacheSet< assoc >::m_lru_max, and FastNehalem::CacheSet< assoc >::m_tags.

6.43.3 Member Data Documentation

6.43.3.1 m_lru

```
template<UInt32 assoc>
UInt64  FastNehalem::CacheSet< assoc >::m_lru[assoc]  [private]
```

Definition at line 40 of file fast_cache.h.

Referenced by FastNehalem::CacheSet< assoc >::find().

6.43.3.2 m_lru_max

```
template<UInt32 assoc>
UInt64 FastNehalem::CacheSet< assoc >::m_lru_max [private]
```

Definition at line 41 of file fast_cache.h.

Referenced by FastNehalem::CacheSet< assoc >::find().

6.43.3.3 m_tags

```
template<UInt32 assoc>
IntPtr FastNehalem::CacheSet< assoc >::m_tags[assoc] [private]
```

Definition at line 39 of file fast_cache.h.

Referenced by FastNehalem::CacheSet< assoc >::find().

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/fast_nehalem/ **fast_cache.h**

6.44 CacheSetInfo Class Reference

`#include <cache_set.h>`

Inheritance diagram for CacheSetInfo:

Public Member Functions

- virtual ~**CacheSetInfo** ()

6.44.1 Detailed Description

Definition at line 15 of file cache_set.h.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 ~CacheSetInfo()

```
virtual CacheSetInfo::~CacheSetInfo ( )  [inline], [virtual]
```

Definition at line 18 of file cache_set.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/cache/ **cache_set.h**

6.45 CacheSetInfoLRU Class Reference

```
#include <cache_set_lru.h>
```

Inheritance diagram for CacheSetInfoLRU:

Public Member Functions

- **CacheSetInfoLRU** (String name, String cfgname, **core_id_t** core_id, **UInt32** associativity, **UInt8** num_attempts)
- virtual ~**CacheSetInfoLRU** ()
- void **increment** (**UInt32** index)
- void **incrementAttempt** (**UInt8** attempt)

Private Attributes

- const **UInt32** **m_associativity**
- **UInt64** * **m_access**
- **UInt64** * **m_attempts**

6.45.1 Detailed Description

Definition at line 6 of file cache_set_lru.h.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 CacheSetInfoLRU()

```
CacheSetInfoLRU::CacheSetInfoLRU (
            String name,
            String cfgname,
             core_id_t core_id,
             UInt32 associativity,
             UInt8 num_attempts )
```

Definition at line 98 of file cache_set_lru.cc.

References itostr(), m_access, m_associativity, m_attempts, and registerStatsMetric().

6.45.2.2 ~CacheSetInfoLRU()

```
CacheSetInfoLRU::~CacheSetInfoLRU ( )  [virtual]
```

Definition at line 120 of file cache_set_lru.cc.

References m_access, and m_attempts.

6.45.3 Member Function Documentation

6.45.3.1 increment()

```
void CacheSetInfoLRU::increment (
             UInt32 index )  [inline]
```

Definition at line 11 of file cache_set_lru.h.

References LOG_ASSERT_ERROR, m_access, and m_associativity.

Referenced by CacheSetKruger::updateReplacementIndex(), CacheSetSRRIP::updateReplacementIndex(), and CacheSetLRU::updateReplacementIndex().

6.45.3.2 incrementAttempt()

```
void CacheSetInfoLRU::incrementAttempt (
             UInt8 attempt )  [inline]
```

Definition at line 16 of file cache_set_lru.h.

References LOG_ASSERT_ERROR, and m_attempts.

Referenced by CacheSetKruger::getReplacementIndex(), CacheSetSRRIP::getReplacementIndex(), and CacheSetLRU::getReplacementIndex().

6.45.4 Member Data Documentation

6.45.4.1 m_access

```
UInt64* CacheSetInfoLRU::m_access  [private]
```

Definition at line 25 of file cache_set_lru.h.

Referenced by CacheSetInfoLRU(), increment(), and ~CacheSetInfoLRU().

6.45.4.2 m_associativity

```
const  UInt32 CacheSetInfoLRU::m_associativity  [private]
```

Definition at line 24 of file cache_set_lru.h.

Referenced by CacheSetInfoLRU(), and increment().

6.45.4.3 m_attempts

```
UInt64* CacheSetInfoLRU::m_attempts  [private]
```

Definition at line 26 of file cache_set_lru.h.

Referenced by CacheSetInfoLRU(), incrementAttempt(), and ~CacheSetInfoLRU().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_set_lru.h**
- common/core/memory_subsystem/cache/ **cache_set_lru.cc**

6.46 CacheSetKruger Class Reference

```
#include <cache_set_kruger.h>
```

Inheritance diagram for CacheSetKruger:

Public Member Functions

- **CacheSetKruger** (**CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize, **CacheSetInfoLRU** *set_info, **UInt8** num_attempts)
- virtual ~**CacheSetKruger** ()
- virtual **UInt32** **getReplacementIndex** (**CacheCntlr** *cntlr)
- void **updateReplacementIndex** (**UInt32** accessed_index)
- **CacheSetKruger** (**CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize)
- ~**CacheSetKruger** ()
- **UInt32** **getReplacementIndex** (**CacheCntlr** *cntlr)
- void **updateReplacementIndex** (**UInt32** accessed_index)

Protected Member Functions

- void **moveToMRU** (**UInt32** accessed_index)
- bool **isValidReplacement** (**UInt32** index)
- void **printBlockStats** ()

Protected Attributes

- const **UInt8** **m_num_attempts**
- **UInt8** * **m_lru_bits**
- **CacheSetInfoLRU** * **m_set_info**

Static Protected Attributes

- static const constexpr char *const **states** [ ] = {"INVALID", "SHARED", "SHARED_UPGRADING", "EXCLUSIVE", "OWNED", "MODIFIED"}

Private Member Functions

- void **printBlockStats** ()
- bool **isValidReplacement2** (**UInt32** index)
- void **injectTest** ()

Additional Inherited Members

6.46.1 Detailed Description

Definition at line 7 of file cache_set_kruger.h.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 CacheSetKruger() [1/2]

```
CacheSetKruger::CacheSetKruger (
              CacheBase::cache_t cache_type,
              UInt32 associativity,
              UInt32 blocksize,
              CacheSetInfoLRU * set_info,
              UInt8 num_attempts )
```

Definition at line 7 of file cache_set_kruger.cc.

References CacheSet::m_associativity, and m_lru_bits.

6.46.2.2 ~CacheSetKruger() [1/2]

```
CacheSetKruger::~CacheSetKruger ( )  [virtual]
```

Definition at line 17 of file cache_set_kruger.cc.

References m_lru_bits.

6.46.2.3 CacheSetKruger() [2/2]

```
CacheSetKruger::CacheSetKruger (
              CacheBase::cache_t cache_type,
              UInt32 associativity,
              UInt32 blocksize )
```

Definition at line 6 of file cache_set_kruger_2.cc.

References CacheSet::m_associativity, and m_lru_bits.

6.46.2.4 ~CacheSetKruger() [2/2]

```
CacheSetKruger::~CacheSetKruger ( )
```

6.46.3 Member Function Documentation

6.46.3.1 getReplacementIndex() [1/2]

```
UInt32 CacheSetKruger::getReplacementIndex (
            CacheCntlr * cntlr )  [virtual]
```

Implements **CacheSet** (p. 227).

6.46.3.2 getReplacementIndex() [2/2]

```
UInt32 CacheSetKruger::getReplacementIndex (
            CacheCntlr * cntlr )  [virtual]
```

Implements **CacheSet** (p. 227).

Definition at line 82 of file cache_set_kruger.cc.

References CacheCntlr::flush(), CacheSetInfoLRU::incrementAttempt(), isValidReplacement(), LOG_ASSERT_↩ ERROR, CacheSet::m_associativity, CacheSet::m_cache_block_info_array, m_lru_bits, m_set_info, CacheState↩ ::MODIFIED, and moveToMRU().

6.46.3.3 injectTest()

```
void CacheSetKruger::injectTest ( )  [private]
```

Definition at line 104 of file cache_set_kruger_2.cc.

References CacheSet::m_cache_block_info_array, CacheState::MODIFIED, and CacheBlockInfo::setCState().

6.46.3.4 isValidReplacement()

```
bool CacheSetKruger::isValidReplacement (
            UInt32 index )  [protected]
```

Definition at line 136 of file cache_set_kruger.cc.

References CacheBlockInfo::getCState(), CacheSet::m_cache_block_info_array, CacheState::MODIFIED, and CacheState::SHARED_UPGRADING.

Referenced by getReplacementIndex().

6.46.3.5 isValidReplacement2()

```
bool CacheSetKruger::isValidReplacement2 (
            UInt32 index )  [private]
```

Definition at line 99 of file cache_set_kruger_2.cc.

References CacheBlockInfo::getCState(), CacheSet::m_cache_block_info_array, and CacheState::MODIFIED.

6.46.3.6 moveToMRU()

```
void CacheSetKruger::moveToMRU (
            UInt32 accessed_index )  [protected]
```

Definition at line 126 of file cache_set_kruger.cc.

References CacheSet::m_associativity, and m_lru_bits.

Referenced by getReplacementIndex(), and updateReplacementIndex().

6.46.3.7 printBlockStats() [1/2]

```
void CacheSetKruger::printBlockStats ( )  [private]
```

6.46.3.8 printBlockStats() [2/2]

```
void CacheSetKruger::printBlockStats ( )  [protected]
```

Definition at line 142 of file cache_set_kruger.cc.

References CacheSet::m_associativity, CacheSet::m_cache_block_info_array, m_lru_bits, and states.

6.46.3.9 updateReplacementIndex() [1/2]

```
void CacheSetKruger::updateReplacementIndex (
            UInt32 accessed_index )  [virtual]
```

Implements **CacheSet** (p. 229).

6.46.3.10 updateReplacementIndex() [2/2]

```
void CacheSetKruger::updateReplacementIndex (
             UInt32 accessed_index )  [virtual]
```

Implements **CacheSet** (p. 229).

Definition at line 120 of file cache_set_kruger.cc.

References CacheSetInfoLRU::increment(), m_lru_bits, m_set_info, and moveToMRU().

6.46.4 Member Data Documentation

6.46.4.1 m_lru_bits

```
UInt8 * CacheSetKruger::m_lru_bits  [protected]
```

Definition at line 21 of file cache_set_kruger.h.

Referenced by CacheSetKruger(), getReplacementIndex(), moveToMRU(), printBlockStats(), updateReplacementIndex(), and ~CacheSetKruger().

6.46.4.2 m_num_attempts

```
const  UInt8 CacheSetKruger::m_num_attempts  [protected]
```

Definition at line 20 of file cache_set_kruger.h.

6.46.4.3 m_set_info

```
CacheSetInfoLRU* CacheSetKruger::m_set_info  [protected]
```

Definition at line 22 of file cache_set_kruger.h.

Referenced by getReplacementIndex(), and updateReplacementIndex().

6.46.4.4 states

```
static const constexpr char *const CacheSetKruger::states = {"INVALID", "SHARED", "SHARED_UPG↩
RADING", "EXCLUSIVE", "OWNED", "MODIFIED"}  [static], [constexpr], [protected]
```

Definition at line 18 of file cache_set_kruger.h.

Referenced by printBlockStats().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_set_kruger.h**
- common/core/memory_subsystem/cache/ **cache_set_kruger_2.h**
- common/core/memory_subsystem/cache/ **cache_set_kruger.cc**
- common/core/memory_subsystem/cache/ **cache_set_kruger_2.cc**

6.47 CacheSetLRU Class Reference

```
#include <cache_set_lru.h>
```

Inheritance diagram for CacheSetLRU:

Public Member Functions

- **CacheSetLRU** (**CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize, **CacheSetInfoLRU** ∗set_info, **UInt8** num_attempts)
- virtual ∼**CacheSetLRU** ()
- virtual **UInt32** **getReplacementIndex** (**CacheCntlr** ∗cntlr)
- void **updateReplacementIndex** (**UInt32** accessed_index)

Protected Member Functions

- void **moveToMRU** (**UInt32** accessed_index)

Protected Attributes

- const **UInt8** **m_num_attempts**
- **UInt8** ∗ **m_lru_bits**
- **CacheSetInfoLRU** ∗ **m_set_info**

Additional Inherited Members

6.47.1 Detailed Description

Definition at line 29 of file cache_set_lru.h.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 CacheSetLRU()

```
CacheSetLRU::CacheSetLRU (
              CacheBase::cache_t cache_type,
              UInt32 associativity,
              UInt32 blocksize,
              CacheSetInfoLRU * set_info,
              UInt8 num_attempts )
```

Definition at line 7 of file cache_set_lru.cc.

References CacheSet::m_associativity, and m_lru_bits.

6.47.2.2 ~CacheSetLRU()

```
CacheSetLRU::~CacheSetLRU ( )  [virtual]
```

Definition at line 19 of file cache_set_lru.cc.

References m_lru_bits.

6.47.3 Member Function Documentation

6.47.3.1 getReplacementIndex()

```
UInt32 CacheSetLRU::getReplacementIndex (
              CacheCntlr * cntlr )  [virtual]
```

Implements **CacheSet** (p. 227).

Definition at line 25 of file cache_set_lru.cc.

References CacheSetInfoLRU::incrementAttempt(), CacheCntlr::incrementQBSLookupCost(), CacheCntlr::isIn↩LowerLevelCache(), CacheSet::isValidReplacement(), LOG_ASSERT_ERROR, LOG_PRINT_ERROR, Cache↩Set::m_associativity, CacheSet::m_cache_block_info_array, m_lru_bits, m_num_attempts, m_set_info, and move↩ToMRU().

6.47.3.2 moveToMRU()

```
void CacheSetLRU::moveToMRU (
            UInt32 accessed_index )  [protected]
```

Definition at line 88 of file cache_set_lru.cc.

References CacheSet::m_associativity, and m_lru_bits.

Referenced by getReplacementIndex(), and updateReplacementIndex().

6.47.3.3 updateReplacementIndex()

```
void CacheSetLRU::updateReplacementIndex (
            UInt32 accessed_index )  [virtual]
```

Implements **CacheSet** (p. 229).

Definition at line 81 of file cache_set_lru.cc.

References CacheSetInfoLRU::increment(), m_lru_bits, m_set_info, and moveToMRU().

6.47.4 Member Data Documentation

6.47.4.1 m_lru_bits

```
UInt8* CacheSetLRU::m_lru_bits  [protected]
```

Definition at line 41 of file cache_set_lru.h.

Referenced by CacheSetLRU(), getReplacementIndex(), moveToMRU(), updateReplacementIndex(), and ~CacheSetLRU().

6.47.4.2 m_num_attempts

```
const UInt8 CacheSetLRU::m_num_attempts  [protected]
```

Definition at line 40 of file cache_set_lru.h.

Referenced by getReplacementIndex().

6.47.4.3 m_set_info

```
CacheSetInfoLRU* CacheSetLRU::m_set_info  [protected]
```

Definition at line 42 of file cache_set_lru.h.

Referenced by getReplacementIndex(), and updateReplacementIndex().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_set_lru.h**
- common/core/memory_subsystem/cache/ **cache_set_lru.cc**

6.48 CacheSetMRU Class Reference

```
#include <cache_set_mru.h>
```

Inheritance diagram for CacheSetMRU:

Public Member Functions

- **CacheSetMRU** (**CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize)
- ~**CacheSetMRU** ()
- **UInt32** **getReplacementIndex** (**CacheCntlr** *cntlr)
- void **updateReplacementIndex** (**UInt32** accessed_index)

Private Attributes

- **UInt8** * **m_lru_bits**

Additional Inherited Members

6.48.1 Detailed Description

Definition at line 6 of file cache_set_mru.h.

6.48.2 Constructor & Destructor Documentation

6.48.2.1 CacheSetMRU()

```
CacheSetMRU::CacheSetMRU (
              CacheBase::cache_t cache_type,
              UInt32 associativity,
              UInt32 blocksize )
```

Definition at line 6 of file cache_set_mru.cc.

References CacheSet::m_associativity, and m_lru_bits.

6.48.2.2 ~CacheSetMRU()

```
CacheSetMRU::~CacheSetMRU ( )
```

Definition at line 16 of file cache_set_mru.cc.

References m_lru_bits.

6.48.3 Member Function Documentation

6.48.3.1 getReplacementIndex()

```
UInt32 CacheSetMRU::getReplacementIndex (
              CacheCntlr * cntlr )  [virtual]
```

Implements **CacheSet** (p. 227).

Definition at line 22 of file cache_set_mru.cc.

References CacheSet::isValidReplacement(), LOG_PRINT_ERROR, CacheSet::m_associativity, CacheSet::m_↩cache_block_info_array, m_lru_bits, and updateReplacementIndex().

6.48.3.2 updateReplacementIndex()

```
void CacheSetMRU::updateReplacementIndex (
              UInt32 accessed_index )  [virtual]
```

Implements **CacheSet** (p. 229).

Definition at line 53 of file cache_set_mru.cc.

References CacheSet::m_associativity, and m_lru_bits.

Referenced by getReplacementIndex().

6.48.4 Member Data Documentation

6.48.4.1 m_lru_bits

```
UInt8* CacheSetMRU::m_lru_bits  [private]
```

Definition at line 17 of file cache_set_mru.h.

Referenced by CacheSetMRU(), getReplacementIndex(), updateReplacementIndex(), and ~CacheSetMRU().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_set_mru.h**
- common/core/memory_subsystem/cache/ **cache_set_mru.cc**

6.49 CacheSetNMRU Class Reference

`#include <cache_set_nmru.h>`

Inheritance diagram for CacheSetNMRU:

Public Member Functions

- **CacheSetNMRU** (**CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize)
- ~**CacheSetNMRU** ()
- **UInt32** **getReplacementIndex** (**CacheCntlr** *cntlr)
- void **updateReplacementIndex** (**UInt32** accessed_index)

Private Attributes

- **UInt8** * **m_lru_bits**
- **UInt8** **m_replacement_pointer**

Additional Inherited Members

6.49.1 Detailed Description

Definition at line 6 of file cache_set_nmru.h.

6.49.2 Constructor & Destructor Documentation

6.49.2.1 CacheSetNMRU()

```
CacheSetNMRU::CacheSetNMRU (
            CacheBase::cache_t cache_type,
            UInt32 associativity,
            UInt32 blocksize )
```

Definition at line 6 of file cache_set_nmru.cc.

References CacheSet::m_associativity, m_lru_bits, and m_replacement_pointer.

6.49.2.2 ~CacheSetNMRU()

```
CacheSetNMRU::~CacheSetNMRU ( )
```

Definition at line 18 of file cache_set_nmru.cc.

References m_lru_bits.

6.49.3 Member Function Documentation

6.49.3.1 getReplacementIndex()

```
UInt32 CacheSetNMRU::getReplacementIndex (
            CacheCntlr * cntlr )  [virtual]
```

Implements **CacheSet** (p. 227).

Definition at line 24 of file cache_set_nmru.cc.

References CacheSet::isValidReplacement(), LOG_PRINT_ERROR, CacheSet::m_associativity, CacheSet::m_↩ cache_block_info_array, m_lru_bits, m_replacement_pointer, and updateReplacementIndex().

6.49.3.2 updateReplacementIndex()

```
void CacheSetNMRU::updateReplacementIndex (
            UInt32 accessed_index )  [virtual]
```

Implements **CacheSet** (p. 229).

Definition at line 55 of file cache_set_nmru.cc.

References CacheSet::m_associativity, and m_lru_bits.

Referenced by getReplacementIndex().

6.49.4 Member Data Documentation

6.49.4.1 m_lru_bits

```
UInt8* CacheSetNMRU::m_lru_bits  [private]
```

Definition at line 17 of file cache_set_nmru.h.

Referenced by CacheSetNMRU(), getReplacementIndex(), updateReplacementIndex(), and ~CacheSetNMRU().

6.49.4.2 m_replacement_pointer

```
UInt8 CacheSetNMRU::m_replacement_pointer  [private]
```

Definition at line 18 of file cache_set_nmru.h.

Referenced by CacheSetNMRU(), and getReplacementIndex().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_set_nmru.h**
- common/core/memory_subsystem/cache/ **cache_set_nmru.cc**

6.50 CacheSetNRU Class Reference

```
#include <cache_set_nru.h>
```

Inheritance diagram for CacheSetNRU:

Public Member Functions

- **CacheSetNRU** (**CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize)
- ~**CacheSetNRU** ()
- **UInt32** **getReplacementIndex** (**CacheCntlr** *cntlr)
- void **updateReplacementIndex** (**UInt32** accessed_index)

Private Attributes

- **UInt8 * m_lru_bits**
- **UInt8 m_num_bits_set**
- **UInt8 m_replacement_pointer**

Additional Inherited Members

6.50.1 Detailed Description

Definition at line 6 of file cache_set_nru.h.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 CacheSetNRU()

```
CacheSetNRU::CacheSetNRU (
              CacheBase::cache_t cache_type,
              UInt32 associativity,
              UInt32 blocksize )
```

Definition at line 6 of file cache_set_nru.cc.

References CacheSet::m_associativity, m_lru_bits, m_num_bits_set, and m_replacement_pointer.

6.50.2.2 ~CacheSetNRU()

```
CacheSetNRU::~CacheSetNRU ( )
```

Definition at line 19 of file cache_set_nru.cc.

References m_lru_bits.

6.50.3 Member Function Documentation

6.50.3.1 getReplacementIndex()

```
UInt32 CacheSetNRU::getReplacementIndex (
            CacheCntlr * cntlr )  [virtual]
```

Implements **CacheSet** (p. 227).

Definition at line 25 of file cache_set_nru.cc.

References CacheSet::isValidReplacement(), LOG_PRINT_ERROR, CacheSet::m_associativity, CacheSet::m_↩ cache_block_info_array, m_lru_bits, m_replacement_pointer, and updateReplacementIndex().

6.50.3.2 updateReplacementIndex()

```
void CacheSetNRU::updateReplacementIndex (
            UInt32 accessed_index )  [virtual]
```

Implements **CacheSet** (p. 229).

Definition at line 69 of file cache_set_nru.cc.

References CacheSet::m_associativity, m_lru_bits, and m_num_bits_set.

Referenced by getReplacementIndex().

6.50.4 Member Data Documentation

6.50.4.1 m_lru_bits

```
UInt8* CacheSetNRU::m_lru_bits  [private]
```

Definition at line 17 of file cache_set_nru.h.

Referenced by CacheSetNRU(), getReplacementIndex(), updateReplacementIndex(), and ∼CacheSetNRU().

6.50.4.2 m_num_bits_set

```
UInt8 CacheSetNRU::m_num_bits_set  [private]
```

Definition at line 18 of file cache_set_nru.h.

Referenced by CacheSetNRU(), and updateReplacementIndex().

6.50.4.3 m_replacement_pointer

```
UInt8 CacheSetNRU::m_replacement_pointer  [private]
```

Definition at line 19 of file cache_set_nru.h.

Referenced by CacheSetNRU(), and getReplacementIndex().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_set_nru.h**
- common/core/memory_subsystem/cache/ **cache_set_nru.cc**

6.51 CacheSetPLRU Class Reference

```
#include <cache_set_plru.h>
```

Inheritance diagram for CacheSetPLRU:

Public Member Functions

- **CacheSetPLRU** (**CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize)
- ∼**CacheSetPLRU** ()
- **UInt32 getReplacementIndex** (**CacheCntlr** ∗cntlr)
- void **updateReplacementIndex** (**UInt32** accessed_index)

Private Attributes

- **UInt8 b** [8]

Additional Inherited Members

6.51.1 Detailed Description

Definition at line 6 of file cache_set_plru.h.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 CacheSetPLRU()

```
CacheSetPLRU::CacheSetPLRU (
              CacheBase::cache_t cache_type,
              UInt32 associativity,
              UInt32 blocksize )
```

Definition at line 6 of file cache_set_plru.cc.

References b, and LOG_ASSERT_ERROR.

6.51.2.2 ~CacheSetPLRU()

```
CacheSetPLRU::~CacheSetPLRU ( )
```

Definition at line 17 of file cache_set_plru.cc.

6.51.3 Member Function Documentation

6.51.3.1 getReplacementIndex()

```
UInt32 CacheSetPLRU::getReplacementIndex (
              CacheCntlr * cntlr )  [virtual]
```

Implements **CacheSet** (p. 227).

Definition at line 22 of file cache_set_plru.cc.

References b, CacheSet::isValidReplacement(), LOG_ASSERT_ERROR, LOG_PRINT_ERROR, CacheSet::m_↩associativity, CacheSet::m_cache_block_info_array, and updateReplacementIndex().

6.51.3.2 updateReplacementIndex()

```
void CacheSetPLRU::updateReplacementIndex (
              UInt32 accessed_index )  [virtual]
```

Implements **CacheSet** (p. 229).

Definition at line 91 of file cache_set_plru.cc.

References b, LOG_PRINT_ERROR, and CacheSet::m_associativity.

Referenced by getReplacementIndex().

6.51.4 Member Data Documentation

6.51.4.1 b

```
UInt8 CacheSetPLRU::b[8]  [private]
```

Definition at line 17 of file cache_set_plru.h.

Referenced by CacheSetPLRU(), getReplacementIndex(), and updateReplacementIndex().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_set_plru.h**
- common/core/memory_subsystem/cache/ **cache_set_plru.cc**

6.52 CacheSetRandom Class Reference

```
#include <cache_set_random.h>
```

Inheritance diagram for CacheSetRandom:

Public Member Functions

- **CacheSetRandom** (**CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize)
- ∼**CacheSetRandom** ()
- **UInt32 getReplacementIndex** (**CacheCntlr** ∗cntlr)
- void **updateReplacementIndex** (**UInt32** accessed_index)

Private Attributes

- **Random m_rand**

Additional Inherited Members

6.52.1 Detailed Description

Definition at line 6 of file cache_set_random.h.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 CacheSetRandom()

```
CacheSetRandom::CacheSetRandom (
            CacheBase::cache_t cache_type,
            UInt32 associativity,
            UInt32 blocksize )
```

Definition at line 8 of file cache_set_random.cc.

References m_rand, and Random::seed().

6.52.2.2 ~CacheSetRandom()

```
CacheSetRandom::~CacheSetRandom ( )
```

Definition at line 16 of file cache_set_random.cc.

6.52.3 Member Function Documentation

6.52.3.1 getReplacementIndex()

```
UInt32 CacheSetRandom::getReplacementIndex (
            CacheCntlr * cntlr )  [virtual]
```

Implements **CacheSet** (p. 227).

Definition at line 21 of file cache_set_random.cc.

References CacheSet::isValidReplacement(), CacheSet::m_associativity, CacheSet::m_cache_block_info_array, m_rand, and Random::next().

6.52.3.2 updateReplacementIndex()

```
void CacheSetRandom::updateReplacementIndex (
            UInt32 accessed_index )  [virtual]
```

Implements **CacheSet** (p. 229).

Definition at line 44 of file cache_set_random.cc.

6.52.4 Member Data Documentation

6.52.4.1 m_rand

Random CacheSetRandom::m_rand [private]

Definition at line 17 of file cache_set_random.h.

Referenced by CacheSetRandom(), and getReplacementIndex().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_set_random.h**
- common/core/memory_subsystem/cache/ **cache_set_random.cc**

6.53 CacheSetRoundRobin Class Reference

`#include <cache_set_round_robin.h>`

Inheritance diagram for CacheSetRoundRobin:

Public Member Functions

- **CacheSetRoundRobin** (**CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize)
- ∼**CacheSetRoundRobin** ()
- **UInt32 getReplacementIndex** (**CacheCntlr** ∗cntlr)
- void **updateReplacementIndex** (**UInt32** accessed_index)

Private Attributes

- **UInt32 m_replacement_index**

Additional Inherited Members

6.53.1 Detailed Description

Definition at line 6 of file cache_set_round_robin.h.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 CacheSetRoundRobin()

```
CacheSetRoundRobin::CacheSetRoundRobin (
            CacheBase::cache_t cache_type,
            UInt32 associativity,
            UInt32 blocksize )
```

Definition at line 3 of file cache_set_round_robin.cc.

References CacheSet::m_associativity, and m_replacement_index.

6.53.2.2 ~CacheSetRoundRobin()

```
CacheSetRoundRobin::~CacheSetRoundRobin ( )
```

Definition at line 11 of file cache_set_round_robin.cc.

6.53.3 Member Function Documentation

6.53.3.1 getReplacementIndex()

```
UInt32 CacheSetRoundRobin::getReplacementIndex (
            CacheCntlr * cntlr )  [virtual]
```

Implements **CacheSet** (p. 227).

Definition at line 15 of file cache_set_round_robin.cc.

References CacheSet::isValidReplacement(), CacheSet::m_associativity, and m_replacement_index.

6.53.3.2 updateReplacementIndex()

```
void CacheSetRoundRobin::updateReplacementIndex (
            UInt32 accessed_index )  [virtual]
```

Implements **CacheSet** (p. 229).

Definition at line 27 of file cache_set_round_robin.cc.

6.53.4 Member Data Documentation

6.53.4.1 m_replacement_index

UInt32 CacheSetRoundRobin::m_replacement_index [private]

Definition at line 17 of file cache_set_round_robin.h.

Referenced by CacheSetRoundRobin(), and getReplacementIndex().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_set_round_robin.h**
- common/core/memory_subsystem/cache/ **cache_set_round_robin.cc**

6.54 CacheSetSRRIP Class Reference

#include <cache_set_srrip.h>

Inheritance diagram for CacheSetSRRIP:

Public Member Functions

- **CacheSetSRRIP** (String cfgname, **core_id_t** core_id, **CacheBase::cache_t** cache_type, **UInt32** associativity, **UInt32** blocksize, **CacheSetInfoLRU** *set_info, **UInt8** num_attempts)
- ~**CacheSetSRRIP** ()
- **UInt32 getReplacementIndex** (**CacheCntlr** *cntlr)
- void **updateReplacementIndex** (**UInt32** accessed_index)

Private Attributes

- const **UInt8 m_rrip_numbits**
- const **UInt8 m_rrip_max**
- const **UInt8 m_rrip_insert**
- const **UInt8 m_num_attempts**
- **UInt8** * **m_rrip_bits**
- **UInt8 m_replacement_pointer**
- **CacheSetInfoLRU** * **m_set_info**

Additional Inherited Members

6.54.1 Detailed Description

Definition at line 7 of file cache_set_srrip.h.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 CacheSetSRRIP()

```
CacheSetSRRIP::CacheSetSRRIP (
            String cfgname,
             core_id_t core_id,
             CacheBase::cache_t cache_type,
             UInt32 associativity,
             UInt32 blocksize,
             CacheSetInfoLRU * set_info,
             UInt8 num_attempts )
```

Definition at line 8 of file cache_set_srrip.cc.

References CacheSet::m_associativity, m_rrip_bits, and m_rrip_insert.

6.54.2.2 ~CacheSetSRRIP()

```
CacheSetSRRIP::~CacheSetSRRIP ( )
```

Definition at line 25 of file cache_set_srrip.cc.

References m_rrip_bits.

6.54.3 Member Function Documentation

6.54.3.1 getReplacementIndex()

```
UInt32 CacheSetSRRIP::getReplacementIndex (
             CacheCntlr * cntlr )  [virtual]
```

Implements **CacheSet** (p. 227).

Definition at line 31 of file cache_set_srrip.cc.

References CacheSetInfoLRU::incrementAttempt(), CacheCntlr::incrementQBSLookupCost(), CacheCntlr::isInLowerLevelCache(), CacheSet::isValidReplacement(), LOG_ASSERT_ERROR, LOG_PRINT_ERROR, CacheSet::m_associativity, CacheSet::m_cache_block_info_array, m_num_attempts, m_replacement_pointer, m_rrip_bits, m_rrip_insert, m_rrip_max, and m_set_info.

6.54.3.2 updateReplacementIndex()

```
void CacheSetSRRIP::updateReplacementIndex (
             UInt32 accessed_index )  [virtual]
```

Implements **CacheSet** (p. 229).

Definition at line 99 of file cache_set_srrip.cc.

References CacheSetInfoLRU::increment(), m_rrip_bits, and m_set_info.

6.54.4 Member Data Documentation

6.54.4.1 m_num_attempts

```
const  UInt8 CacheSetSRRIP::m_num_attempts  [private]
```

Definition at line 22 of file cache_set_srrip.h.

Referenced by getReplacementIndex().

6.54.4.2 m_replacement_pointer

```
 UInt8 CacheSetSRRIP::m_replacement_pointer  [private]
```

Definition at line 24 of file cache_set_srrip.h.

Referenced by getReplacementIndex().

6.54.4.3 m_rrip_bits

```
 UInt8* CacheSetSRRIP::m_rrip_bits  [private]
```

Definition at line 23 of file cache_set_srrip.h.

Referenced by CacheSetSRRIP(), getReplacementIndex(), updateReplacementIndex(), and ~CacheSetSRRIP().

6.54.4.4 m_rrip_insert

```
const UInt8 CacheSetSRRIP::m_rrip_insert [private]
```

Definition at line 21 of file cache_set_srrip.h.

Referenced by CacheSetSRRIP(), and getReplacementIndex().

6.54.4.5 m_rrip_max

```
const UInt8 CacheSetSRRIP::m_rrip_max [private]
```

Definition at line 20 of file cache_set_srrip.h.

Referenced by getReplacementIndex().

6.54.4.6 m_rrip_numbits

```
const UInt8 CacheSetSRRIP::m_rrip_numbits [private]
```

Definition at line 19 of file cache_set_srrip.h.

6.54.4.7 m_set_info

```
CacheSetInfoLRU* CacheSetSRRIP::m_set_info [private]
```

Definition at line 25 of file cache_set_srrip.h.

Referenced by getReplacementIndex(), and updateReplacementIndex().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **cache_set_srrip.h**
- common/core/memory_subsystem/cache/ **cache_set_srrip.cc**

6.55 CacheState Class Reference

```
#include <cache_state.h>
```

Public Types

- enum **cstate_t** {
  CSTATE_FIRST = 0, **INVALID** = CSTATE_FIRST, **SHARED**, **SHARED_UPGRADING**,
  EXCLUSIVE, **OWNED**, **MODIFIED**, **NUM_CSTATE_STATES**,
  INVALID_COLD = NUM_CSTATE_STATES, **INVALID_EVICT**, **INVALID_COHERENCY**, **NUM_CSTAT↩**
  E_SPECIAL_STATES }

Public Member Functions

- **CacheState** (**cstate_t** state= **INVALID**)
- ∼**CacheState** ()
- bool **readable** ()
- bool **writable** ()

Private Attributes

- **cstate_t cstate**

6.55.1 Detailed Description

Definition at line 8 of file cache_state.h.

6.55.2 Member Enumeration Documentation

6.55.2.1 cstate_t

enum **CacheState::cstate_t**

Enumerator

CSTATE_FIRST	
INVALID	
SHARED	
SHARED_UPGRADING	
EXCLUSIVE	
OWNED	
MODIFIED	
NUM_CSTATE_STATES	
INVALID_COLD	
INVALID_EVICT	
INVALID_COHERENCY	
NUM_CSTATE_SPECIAL_STATES	

Definition at line 11 of file cache_state.h.

6.55.3 Constructor & Destructor Documentation

6.55.3.1 CacheState()

```
CacheState::CacheState (
            cstate_t state =  INVALID )  [inline]
```

Definition at line 28 of file cache_state.h.

6.55.3.2 ~CacheState()

```
CacheState::~CacheState ( )  [inline]
```

Definition at line 29 of file cache_state.h.

6.55.4 Member Function Documentation

6.55.4.1 readable()

```
bool CacheState::readable ( )  [inline]
```

Definition at line 31 of file cache_state.h.

References cstate, EXCLUSIVE, MODIFIED, OWNED, and SHARED.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::operationPermissibleinCache().

6.55.4.2 writable()

```
bool CacheState::writable ( )  [inline]
```

Definition at line 36 of file cache_state.h.

References cstate, and MODIFIED.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::operationPermissibleinCache().

6.55.5 Member Data Documentation

6.55.5.1 cstate

```
cstate_t CacheState::cstate  [private]
```

Definition at line 42 of file cache_state.h.

Referenced by readable(), and writable().

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/cache/ **cache_state.h**

6.56 CacheEfficiencyTracker::Callbacks Struct Reference

`#include <cache_efficiency_tracker.h>`

Public Member Functions

- **Callbacks** ()
- **UInt64 call_get_owner** (**core_id_t** core_id, **UInt64** address) const
- void **call_notify_access** (**UInt64** owner, **Core::mem_op_t** mem_op_type, **HitWhere::where_t** hit_where) const
- void **call_notify_evict** (bool on_roi_end, **UInt64** owner, **UInt64** evictor, **CacheBlockInfo::BitsUsedType** bits_used, **UInt32** bits_total) const

Public Attributes

- **CacheEfficiencyTracker::CallbackGetOwner get_owner_func**
- **CacheEfficiencyTracker::CallbackNotifyAccess notify_access_func**
- **CacheEfficiencyTracker::CallbackNotifyEvict notify_evict_func**
- **UInt64 user_arg**

6.56.1 Detailed Description

Definition at line 14 of file cache_efficiency_tracker.h.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 Callbacks()

```
CacheEfficiencyTracker::Callbacks::Callbacks ( )  [inline]
```

Definition at line 21 of file cache_efficiency_tracker.h.

6.56.3 Member Function Documentation

6.56.3.1 call_get_owner()

```
UInt64 CacheEfficiencyTracker::Callbacks::call_get_owner (
            core_id_t core_id,
            UInt64 address ) const  [inline]
```

Definition at line 23 of file cache_efficiency_tracker.h.

References get_owner_func, and user_arg.

6.56.3.2 call_notify_access()

```
void CacheEfficiencyTracker::Callbacks::call_notify_access (
            UInt64 owner,
            Core::mem_op_t mem_op_type,
            HitWhere::where_t hit_where ) const  [inline]
```

Definition at line 24 of file cache_efficiency_tracker.h.

References notify_access_func, and user_arg.

6.56.3.3 call_notify_evict()

```
void CacheEfficiencyTracker::Callbacks::call_notify_evict (
            bool on_roi_end,
            UInt64 owner,
            UInt64 evictor,
            CacheBlockInfo::BitsUsedType bits_used,
            UInt32 bits_total ) const  [inline]
```

Definition at line 26 of file cache_efficiency_tracker.h.

References notify_evict_func, and user_arg.

6.56.4 Member Data Documentation

6.56.4.1 get_owner_func

CacheEfficiencyTracker::CallbackGetOwner CacheEfficiencyTracker::Callbacks::get_owner_func

Definition at line 16 of file cache_efficiency_tracker.h.

Referenced by call_get_owner(), and Config::setCacheEfficiencyCallbacks().

6.56.4.2 notify_access_func

CacheEfficiencyTracker::CallbackNotifyAccess CacheEfficiencyTracker::Callbacks::notify_↩
access_func

Definition at line 17 of file cache_efficiency_tracker.h.

Referenced by call_notify_access(), and Config::setCacheEfficiencyCallbacks().

6.56.4.3 notify_evict_func

CacheEfficiencyTracker::CallbackNotifyEvict CacheEfficiencyTracker::Callbacks::notify_evict↩
_func

Definition at line 18 of file cache_efficiency_tracker.h.

Referenced by call_notify_evict(), Config::hasCacheEfficiencyCallbacks(), and Config::setCacheEfficiency↩
Callbacks().

6.56.4.4 user_arg

UInt64 CacheEfficiencyTracker::Callbacks::user_arg

Definition at line 19 of file cache_efficiency_tracker.h.

Referenced by call_get_owner(), call_notify_access(), call_notify_evict(), and Config::setCacheEfficiency↩
Callbacks().

The documentation for this struct was generated from the following file:

- common/system/ **cache_efficiency_tracker.h**

6.57 CheetahManager Class Reference

#include <cheetah_manager.h>

Classes

- class **CheetahStats**

Public Member Functions

- **CheetahManager** (**core_id_t** core_id)
- ~**CheetahManager** ()
- void **access** (**Core::mem_op_t** mem_op_type, **IntPtr** address)

Private Types

- enum **cheetah_types_t** {
  CHEETAH_LOCAL, **CHEETAH_BY2**, **CHEETAH_BY4**, **CHEETAH_BY8**,
  CHEETAH_GLOBAL, **NUM_CHEETAH_TYPES** }

Private Attributes

- const **UInt32** **m_min_bits**
- const **UInt32** **m_max_bits_local**
- const **UInt32** **m_max_bits_global**
- **CheetahModel** * **m_cheetah** [**NUM_CHEETAH_TYPES**]
- **IntPtr** **m_address_buffer** [**ADDRESS_BUFFER_SIZE**]
- **UInt32** **m_address_buffer_size**

Static Private Attributes

- static const char * **cheetah_names** [ ] = { "local", "by-2", "by-4", "by-8", "global" }
- static **CheetahStats** * **s_cheetah_stats** = NULL
- static std::vector< std::vector< **CheetahModel** * > > **s_cheetah_models**
- static const **UInt32** **ADDRESS_BUFFER_SIZE** = 256

6.57.1 Detailed Description

Definition at line 9 of file cheetah_manager.h.

6.57.2 Member Enumeration Documentation

6.57.2.1 cheetah_types_t

enum **CheetahManager::cheetah_types_t** [private]

Enumerator

CHEETAH_LOCAL	
CHEETAH_BY2	
CHEETAH_BY4	
CHEETAH_BY8	
CHEETAH_GLOBAL	
NUM_CHEETAH_TYPES	

Definition at line 12 of file cheetah_manager.h.

6.57.3 Constructor & Destructor Documentation

6.57.3.1 CheetahManager()

```
CheetahManager::CheetahManager (
            core_id_t core_id )
```

Definition at line 13 of file cheetah_manager.cc.

References CHEETAH_BY2, CHEETAH_BY4, CHEETAH_BY8, CHEETAH_GLOBAL, CHEETAH_LOCAL, CheetahModel::getMinSize(), LOG_ASSERT_ERROR, m_cheetah, m_max_bits_global, m_max_bits_local, m_↩ min_bits, s_cheetah_models, and s_cheetah_stats.

6.57.3.2 ~CheetahManager()

```
CheetahManager::~CheetahManager ( )
```

Definition at line 45 of file cheetah_manager.cc.

6.57.4 Member Function Documentation

6.57.4.1 access()

```
void CheetahManager::access (
            Core::mem_op_t mem_op_type,
            IntPtr address )
```

Definition at line 49 of file cheetah_manager.cc.

References ADDRESS_BUFFER_SIZE, m_address_buffer, m_address_buffer_size, m_cheetah, and NUM_CH↩ EETAH_TYPES.

Referenced by Core::accessMemoryFast(), and Core::initiateMemoryAccess().

6.57.5 Member Data Documentation

6.57.5.1 ADDRESS_BUFFER_SIZE

```
const UInt32 CheetahManager::ADDRESS_BUFFER_SIZE = 256 [static], [private]
```

Definition at line 45 of file cheetah_manager.h.

Referenced by access().

6.57.5.2 cheetah_names

```
const char * CheetahManager::cheetah_names = { "local", "by-2", "by-4", "by-8", "global" }
[static], [private]
```

Definition at line 20 of file cheetah_manager.h.

Referenced by CheetahManager::CheetahStats::CheetahStats().

6.57.5.3 m_address_buffer

```
IntPtr CheetahManager::m_address_buffer[ ADDRESS_BUFFER_SIZE] [private]
```

Definition at line 46 of file cheetah_manager.h.

Referenced by access().

6.57.5.4 m_address_buffer_size

```
UInt32 CheetahManager::m_address_buffer_size [private]
```

Definition at line 47 of file cheetah_manager.h.

Referenced by access().

6.57.5.5 m_cheetah

CheetahModel* CheetahManager::m_cheetah[**NUM_CHEETAH_TYPES**] [private]

Definition at line 43 of file cheetah_manager.h.

Referenced by access(), and CheetahManager().

6.57.5.6 m_max_bits_global

const **UInt32** CheetahManager::m_max_bits_global [private]

Definition at line 42 of file cheetah_manager.h.

Referenced by CheetahManager().

6.57.5.7 m_max_bits_local

const **UInt32** CheetahManager::m_max_bits_local [private]

Definition at line 41 of file cheetah_manager.h.

Referenced by CheetahManager().

6.57.5.8 m_min_bits

const **UInt32** CheetahManager::m_min_bits [private]

Definition at line 40 of file cheetah_manager.h.

Referenced by CheetahManager().

6.57.5.9 s_cheetah_models

std::vector< std::vector< **CheetahModel** * > > CheetahManager::s_cheetah_models [static], [private]

Definition at line 38 of file cheetah_manager.h.

Referenced by CheetahManager(), and CheetahManager::CheetahStats::update().

6.57.5.10 s_cheetah_stats

```
CheetahManager::CheetahStats * CheetahManager::s_cheetah_stats = NULL  [static], [private]
```

Definition at line 37 of file cheetah_manager.h.

Referenced by CheetahManager().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cheetah/ **cheetah_manager.h**
- common/core/memory_subsystem/cheetah/ **cheetah_manager.cc**

6.58 CheetahModel Class Reference

`#include <cheetah_model.h>`

Public Member Functions

- **CheetahModel** (bool locked, unsigned min_size_bits, unsigned max_size_bits)
- ~**CheetahModel** ()
- void **accesses** (**IntPtr** *addrs, int count)
- void **updateStats** (std::vector< **UInt64** > &stats)

Static Public Member Functions

- static unsigned **getMinSize** ()

Private Member Functions

- void **access** (**IntPtr** addr)

Private Attributes

- const unsigned **m_min_sets_log2**
- const unsigned **m_max_sets_log2**
- **CheetahSACLRU** **cheetah**
- bool **m_locked**
- **Lock** **m_lock**

Static Private Attributes

- static const unsigned **associativity_log2** = 4
- static const unsigned **line_size_log2** = 6

6.58.1 Detailed Description

Definition at line 10 of file cheetah_model.h.

6.58.2 Constructor & Destructor Documentation

6.58.2.1 CheetahModel()

```
CheetahModel::CheetahModel (
            bool locked,
            unsigned min_size_bits,
            unsigned max_size_bits )
```

Definition at line 3 of file cheetah_model.cc.

6.58.2.2 ~CheetahModel()

```
CheetahModel::~CheetahModel ( )
```

Definition at line 11 of file cheetah_model.cc.

6.58.3 Member Function Documentation

6.58.3.1 access()

```
void CheetahModel::access (
             IntPtr addr )  [private]
```

Definition at line 37 of file cheetah_model.cc.

References cheetah, and CheetahSACLRU::sacnmul_woarr().

Referenced by accesses().

6.58.3.2 accesses()

```
void CheetahModel::accesses (
              IntPtr * addrs,
             int count )
```

Definition at line 25 of file cheetah_model.cc.

References access(), TLock< T_LockCreator >::acquire(), m_lock, m_locked, and TLock< T_LockCreator >::release().

6.58.3.3 getMinSize()

```
static unsigned CheetahModel::getMinSize ( )  [inline], [static]
```

Definition at line 24 of file cheetah_model.h.

References associativity_log2, and line_size_log2.

Referenced by CheetahManager::CheetahManager().

6.58.3.4 updateStats()

```
void CheetahModel::updateStats (
             std::vector<  UInt64 > & stats )
```

Definition at line 15 of file cheetah_model.cc.

References associativity_log2, cheetah, CheetahSACLRU::hits(), line_size_log2, m_max_sets_log2, m_min_sets_log2, and CheetahSACLRU::numentries().

6.58.4 Member Data Documentation

6.58.4.1 associativity_log2

```
const unsigned CheetahModel::associativity_log2 = 4  [static], [private]
```

Definition at line 13 of file cheetah_model.h.

Referenced by getMinSize(), and updateStats().

6.58.4.2 cheetah

```
CheetahSACLRU CheetahModel::cheetah  [private]
```

Definition at line 17 of file cheetah_model.h.

Referenced by access(), and updateStats().

6.58.4.3 line_size_log2

```
const unsigned CheetahModel::line_size_log2 = 6  [static], [private]
```

Definition at line 14 of file cheetah_model.h.

Referenced by getMinSize(), and updateStats().

6.58.4.4 m_lock

```
Lock CheetahModel::m_lock  [private]
```

Definition at line 19 of file cheetah_model.h.

Referenced by accesses().

6.58.4.5 m_locked

```
bool CheetahModel::m_locked  [private]
```

Definition at line 18 of file cheetah_model.h.

Referenced by accesses().

6.58.4.6 m_max_sets_log2

```
const unsigned CheetahModel::m_max_sets_log2  [private]
```

Definition at line 16 of file cheetah_model.h.

Referenced by updateStats().

6.58.4.7 m_min_sets_log2

```
const unsigned CheetahModel::m_min_sets_log2  [private]
```

Definition at line 15 of file cheetah_model.h.

Referenced by updateStats().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cheetah/ **cheetah_model.h**
- common/core/memory_subsystem/cheetah/ **cheetah_model.cc**

6.59 CheetahSACLRU Class Reference

```
#include <saclru.h>
```

Public Member Functions

- **CheetahSACLRU** (int _N, int _B, int _A, int _L)
- ~**CheetahSACLRU** ()
- void **outpr_saclru** (FILE *fd)
- void **init_saclru** (void)
- void **sacnmul_woarr** (intptr_t addr)
- void **flush** (void)
- uint64_t **hits** (unsigned sets_log2, unsigned assoc)
- uint64_t **numentries** (void)

Private Attributes

- const unsigned int **N**
- const unsigned int **B**
- const unsigned int **A**
- const unsigned int **L**
- const unsigned int **SAVE_INTERVAL**
- const unsigned int **P_INTERVAL**
- unsigned **TWO_PWR_N**
- unsigned **MAX_DEPTH**
- int **TWO_POWER_MAX_DEPTH**
- int **SET_MASK**
- int **DIFF_SET_MASK**
- int **BASE**
- int **SIZE_OF_TREE**
- int **BASE_PWR_MAX_DEPTH_PLUS_ONE**
- uint64_t * **arr**
- uint64_t ** **sac_hits**
- int64_t * **hitarr**
- uint64_t * **base_pwr_array**
- uint64_t * **rm_arr**
- uint64_t **t_entries**
- unsigned long **next_save_time**
- uint64_t * **depths**
- uint64_t **hitarr0**
- uint64_t **tag**

6.59.1 Detailed Description

Definition at line 7 of file saclru.h.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 CheetahSACLRU()

```
CheetahSACLRU::CheetahSACLRU (
            int _N,
            int _B,
            int _A,
            int _L )
```

Definition at line 35 of file saclru.cc.

References init_saclru().

6.59.2.2 ~CheetahSACLRU()

```
CheetahSACLRU::~CheetahSACLRU ( )
```

Definition at line 43 of file saclru.cc.

6.59.3 Member Function Documentation

6.59.3.1 flush()

```
void CheetahSACLRU::flush (
            void  )
```

Definition at line 56 of file saclru.cc.

References hitarr0, MAX_DEPTH, and sac_hits.

Referenced by hits(), and outpr_saclru().

6.59.3.2 hits()

```
uint64_t CheetahSACLRU::hits (
            unsigned sets_log2,
            unsigned assoc )
```

Definition at line 109 of file saclru.cc.

References A, B, flush(), hitarr0, and sac_hits.

Referenced by CheetahModel::updateStats().

6.59.3.3 init_saclru()

```
void CheetahSACLRU::init_saclru (
            void  )
```

Definition at line 135 of file saclru.cc.

References A, arr, B, B80000000, BASE, base_pwr_array, BASE_PWR_MAX_DEPTH_PLUS_ONE, depths, D↩IFF_SET_MASK, fatal(), hitarr, idim2(), MAX_DEPTH, MEM_AVAIL_HITARR, N, next_save_time, ONE, power(), rm_arr, sac_hits, SAVE_INTERVAL, SET_MASK, SIZE_OF_TREE, TWO, TWO_POWER_MAX_DEPTH, and T↩WO_PWR_N.

Referenced by CheetahSACLRU().

6.59.3.4 numentries()

```
uint64_t CheetahSACLRU::numentries (
            void  )  [inline]
```

Definition at line 58 of file saclru.h.

References t_entries.

Referenced by CheetahModel::updateStats().

6.59.3.5 outpr_saclru()

```
void CheetahSACLRU::outpr_saclru (
            FILE * fd )
```

Definition at line 72 of file saclru.cc.

References A, flush(), L, MAX_DEPTH, ONE, sac_hits, t_entries, and TWO_PWR_N.

Referenced by sacnmul_woarr().

6.59.3.6 sacnmul_woarr()

```
void CheetahSACLRU::sacnmul_woarr (
            intptr_t addr )
```

Definition at line 240 of file saclru.cc.

References A, arr, BASE, depths, DIFF_SET_MASK, hitarr0, L, MAX_DEPTH, next_save_time, ONE, outpr_↩ saclru(), P_INTERVAL, rm_arr, sac_hits, SAVE_INTERVAL, SET_MASK, SIZE_OF_TREE, t_entries, tag, and T↩ WO_PWR_N.

Referenced by CheetahModel::access().

6.59.4 Member Data Documentation

6.59.4.1 A

```
const unsigned int CheetahSACLRU::A  [private]
```

Definition at line 12 of file saclru.h.

Referenced by hits(), init_saclru(), outpr_saclru(), and sacnmul_woarr().

6.59.4.2 arr

```
uint64_t* CheetahSACLRU::arr  [private]
```

Definition at line 30 of file saclru.h.

Referenced by init_saclru(), and sacnmul_woarr().

6.59.4.3 B

```
const unsigned int CheetahSACLRU::B  [private]
```

Definition at line 11 of file saclru.h.

Referenced by hits(), and init_saclru().

6.59.4.4 BASE

```
int CheetahSACLRU::BASE  [private]
```

Definition at line 23 of file saclru.h.

Referenced by init_saclru(), and sacnmul_woarr().

6.59.4.5 base_pwr_array

```
uint64_t* CheetahSACLRU::base_pwr_array  [private]
```

Definition at line 34 of file saclru.h.

Referenced by init_saclru().

6.59.4.6 BASE_PWR_MAX_DEPTH_PLUS_ONE

```
int CheetahSACLRU::BASE_PWR_MAX_DEPTH_PLUS_ONE  [private]
```

Definition at line 28 of file saclru.h.

Referenced by init_saclru().

6.59.4.7 depths

```
uint64_t* CheetahSACLRU::depths  [private]
```

Definition at line 40 of file saclru.h.

Referenced by init_saclru(), and sacnmul_woarr().

6.59.4.8 DIFF_SET_MASK

```
int CheetahSACLRU::DIFF_SET_MASK  [private]
```

Definition at line 22 of file saclru.h.

Referenced by init_saclru(), and sacnmul_woarr().

6.59.4.9 hitarr

```
int64_t* CheetahSACLRU::hitarr  [private]
```

Definition at line 32 of file saclru.h.

Referenced by init_saclru().

6.59.4.10 hitarr0

```
uint64_t CheetahSACLRU::hitarr0  [private]
```

Definition at line 41 of file saclru.h.

Referenced by flush(), hits(), and sacnmul_woarr().

6.59.4.11 L

```
const unsigned int CheetahSACLRU::L  [private]
```

Definition at line 13 of file saclru.h.

Referenced by outpr_saclru(), and sacnmul_woarr().

6.59.4.12 MAX_DEPTH

```
unsigned CheetahSACLRU::MAX_DEPTH  [private]
```

Definition at line 19 of file saclru.h.

Referenced by flush(), init_saclru(), outpr_saclru(), and sacnmul_woarr().

6.59.4.13 N

```
const unsigned int CheetahSACLRU::N  [private]
```

Definition at line 10 of file saclru.h.

Referenced by init_saclru().

6.59.4.14 next_save_time

```
unsigned long CheetahSACLRU::next_save_time  [private]
```

Definition at line 39 of file saclru.h.

Referenced by init_saclru(), and sacnmul_woarr().

6.59.4.15 P_INTERVAL

```
const unsigned int CheetahSACLRU::P_INTERVAL  [private]
```

Definition at line 16 of file saclru.h.

Referenced by sacnmul_woarr().

6.59.4.16 rm_arr

```
uint64_t* CheetahSACLRU::rm_arr  [private]
```

Definition at line 35 of file saclru.h.

Referenced by init_saclru(), and sacnmul_woarr().

6.59.4.17 sac_hits

```
uint64_t** CheetahSACLRU::sac_hits  [private]
```

Definition at line 31 of file saclru.h.

Referenced by flush(), hits(), init_saclru(), outpr_saclru(), and sacnmul_woarr().

6.59.4.18 SAVE_INTERVAL

```
const unsigned int CheetahSACLRU::SAVE_INTERVAL  [private]
```

Definition at line 15 of file saclru.h.

Referenced by init_saclru(), and sacnmul_woarr().

6.59.4.19 SET_MASK

```
int CheetahSACLRU::SET_MASK  [private]
```

Definition at line 21 of file saclru.h.

Referenced by init_saclru(), and sacnmul_woarr().

6.59.4.20 SIZE_OF_TREE

```
int CheetahSACLRU::SIZE_OF_TREE  [private]
```

Definition at line 24 of file saclru.h.

Referenced by init_saclru(), and sacnmul_woarr().

6.59.4.21 t_entries

```
uint64_t CheetahSACLRU::t_entries  [private]
```

Definition at line 37 of file saclru.h.

Referenced by numentries(), outpr_saclru(), and sacnmul_woarr().

6.59.4.22 tag

```
uint64_t CheetahSACLRU::tag  [private]
```

Definition at line 42 of file saclru.h.

Referenced by sacnmul_woarr().

6.59.4.23 TWO_POWER_MAX_DEPTH

```
int CheetahSACLRU::TWO_POWER_MAX_DEPTH  [private]
```

Definition at line 20 of file saclru.h.

Referenced by init_saclru().

6.59.4.24 TWO_PWR_N

```
unsigned CheetahSACLRU::TWO_PWR_N  [private]
```

Definition at line 18 of file saclru.h.

Referenced by init_saclru(), outpr_saclru(), and sacnmul_woarr().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cheetah/ **saclru.h**
- common/core/memory_subsystem/cheetah/ **saclru.cc**

6.60 CheetahManager::CheetahStats Class Reference

Public Member Functions

- **CheetahStats** (**UInt32** min_bits, **UInt32** max_bits_local, **UInt32** max_bits_global)

Private Member Functions

- void **update** ()

Static Private Member Functions

- static **SInt64** **hook_update** (**UInt64** user, **UInt64** args)

Private Attributes

- const **UInt32** **m_min_bits**
- const **UInt32** **m_max_bits_local**
- const **UInt32** **m_max_bits_global**
- std::vector< std::vector< **UInt64** > > **m_stats**

6.60.1 Detailed Description

Definition at line 22 of file cheetah_manager.h.

6.60.2 Constructor & Destructor Documentation

6.60.2.1 CheetahStats()

```
CheetahManager::CheetahStats::CheetahStats (
            UInt32 min_bits,
            UInt32 max_bits_local,
            UInt32 max_bits_global )
```

Definition at line 61 of file cheetah_manager.cc.

References CheetahManager::CHEETAH_GLOBAL, CheetahManager::cheetah_names, HookType::HOOK_PR↩E_STAT_WRITE, hook_update(), m_stats, CheetahManager::NUM_CHEETAH_TYPES, HooksManager::ORDE↩R_NOTIFY_PRE, and registerStatsMetric().

6.60.3 Member Function Documentation

6.60.3.1 hook_update()

```
static SInt64 CheetahManager::CheetahStats::hook_update (
            UInt64 user,
            UInt64 args ) [inline], [static], [private]
```

Definition at line 30 of file cheetah_manager.h.

Referenced by CheetahStats().

6.60.3.2 update()

```
void CheetahManager::CheetahStats::update ( ) [private]
```

Definition at line 77 of file cheetah_manager.cc.

References CheetahManager::NUM_CHEETAH_TYPES, and CheetahManager::s_cheetah_models.

6.60.4 Member Data Documentation

6.60.4.1 m_max_bits_global

```
const UInt32 CheetahManager::CheetahStats::m_max_bits_global [private]
```

Definition at line 27 of file cheetah_manager.h.

6.60.4.2 m_max_bits_local

```
const UInt32 CheetahManager::CheetahStats::m_max_bits_local [private]
```

Definition at line 26 of file cheetah_manager.h.

6.60.4.3 m_min_bits

```
const UInt32 CheetahManager::CheetahStats::m_min_bits [private]
```

Definition at line 25 of file cheetah_manager.h.

6.60.4.4 m_stats

```
std::vector<std::vector< UInt64> > CheetahManager::CheetahStats::m_stats [private]
```

Definition at line 28 of file cheetah_manager.h.

Referenced by CheetahStats().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cheetah/ **cheetah_manager.h**
- common/core/memory_subsystem/cheetah/ **cheetah_manager.cc**

6.61 CircularLog Class Reference

```
#include <circular_log.h>
```

Classes

- struct **event_t**

Public Member Functions

- **CircularLog** (String filename)
- ~**CircularLog** ()
- void **insert** (const char *type, const char *msg,...) **__attribute__**((format(printf
- void **UInt64 getTime** () const

Static Public Member Functions

- static void **init** (String filename)
- static void **enableCallbacks** ()
- static void **fini** ()
- static void **dump** ()

Static Public Attributes

- static **CircularLog** * **g_singleton** = NULL

Private Member Functions

- void **writeLog** ()
- void **writeEntry** (FILE *fp, int idx)

Static Private Member Functions

- static **SInt64** **hook_sigusr1** (**UInt64**, **UInt64**)

Private Attributes

- const String **m_filename**
- **event_t** *const **m_buffer**
- **Lock** **m_lock**
- **UInt64** **m_eventnum**
- **UInt64** **m_time_zero**

Static Private Attributes

- static const **UInt64** **BUFFER_SIZE** = 1024*1024

6.61.1 Detailed Description

Definition at line 18 of file circular_log.h.

6.61.2 Constructor & Destructor Documentation

6.61.2.1 CircularLog()

```
CircularLog::CircularLog (
            String filename )
```

Definition at line 39 of file circular_log.cc.

Referenced by init().

6.61.2.2 ~CircularLog()

```
CircularLog::~CircularLog ( )
```

Definition at line 47 of file circular_log.cc.

References m_buffer, and writeLog().

6.61.3 Member Function Documentation

6.61.3.1 dump()

```
void CircularLog::dump ( )  [static]
```

Definition at line 30 of file circular_log.cc.

References g_singleton, m_lock, and writeLog().

Referenced by hook_sigusr1().

6.61.3.2 enableCallbacks()

```
void CircularLog::enableCallbacks ( )  [static]
```

Definition at line 15 of file circular_log.cc.

References g_singleton, HookType::HOOK_SIGUSR1, and hook_sigusr1().

Referenced by Simulator::start().

6.61.3.3 fini()

```
void CircularLog::fini ( )  [static]
```

Definition at line 21 of file circular_log.cc.

References g_singleton.

Referenced by Log::log(), and Log::~Log().

6.61.3.4 getTime()

```
void  UInt64 CircularLog::getTime ( ) const  [inline]
```

Definition at line 32 of file circular_log.h.

References m_time_zero, and rdtsc().

Referenced by insert().

6.61.3.5 hook_sigusr1()

```
static  SInt64 CircularLog::hook_sigusr1 (
              UInt64 ,
              UInt64  )  [inline], [static], [private]
```

Definition at line 42 of file circular_log.h.

References dump().

Referenced by enableCallbacks().

6.61.3.6 init()

```
void CircularLog::init (
             String filename )  [static]
```

Definition at line 10 of file circular_log.cc.

References CircularLog(), and g_singleton.

Referenced by Log::Log().

6.61.3.7 insert()

```
void CircularLog::insert (
             const char * type,
             const char * msg,
              ...  )
```

Definition at line 53 of file circular_log.cc.

References BUFFER_SIZE, getTime(), m_buffer, m_eventnum, CircularLog::event_t::msg, CircularLog::event_t↩ ::time, and CircularLog::event_t::type.

6.61.3.8 writeEntry()

```
void CircularLog::writeEntry (
            FILE * fp,
            int idx )  [private]
```

Definition at line 87 of file circular_log.cc.

References CircularLog::event_t::args, m_buffer, CircularLog::event_t::msg, CircularLog::event_t::time, and CircularLog::event_t::type.

Referenced by writeLog().

6.61.3.9 writeLog()

```
void CircularLog::writeLog ( )  [private]
```

Definition at line 68 of file circular_log.cc.

References BUFFER_SIZE, m_eventnum, m_filename, and writeEntry().

Referenced by dump(), and ~CircularLog().

6.61.4 Member Data Documentation

6.61.4.1 BUFFER_SIZE

```
const  UInt64 CircularLog::BUFFER_SIZE = 1024*1024  [static], [private]
```

Definition at line 47 of file circular_log.h.

Referenced by insert(), and writeLog().

6.61.4.2 g_singleton

```
 CircularLog * CircularLog::g_singleton = NULL  [static]
```

Definition at line 26 of file circular_log.h.

Referenced by dump(), enableCallbacks(), fini(), and init().

6.61.4.3 m_buffer

```
event_t* const CircularLog::m_buffer  [private]
```

Definition at line 50 of file circular_log.h.

Referenced by insert(), writeEntry(), and ~CircularLog().

6.61.4.4 m_eventnum

```
UInt64 CircularLog::m_eventnum  [private]
```

Definition at line 52 of file circular_log.h.

Referenced by insert(), and writeLog().

6.61.4.5 m_filename

```
const String CircularLog::m_filename  [private]
```

Definition at line 49 of file circular_log.h.

Referenced by writeLog().

6.61.4.6 m_lock

```
Lock CircularLog::m_lock  [private]
```

Definition at line 51 of file circular_log.h.

Referenced by dump().

6.61.4.7 m_time_zero

```
UInt64 CircularLog::m_time_zero  [private]
```

Definition at line 53 of file circular_log.h.

Referenced by getTime().

The documentation for this class was generated from the following files:

- common/misc/ **circular_log.h**
- common/misc/ **circular_log.cc**

6.62 CircularQueue< T > Class Template Reference

`#include <circular_queue.h>`

Inheritance diagram for CircularQueue< T >:

Classes

- class **iterator**

Public Types

- typedef T **value_type**

Public Member Functions

- **CircularQueue** (**UInt32** **size**=63)
- **CircularQueue** (const **CircularQueue** &queue)
- ~**CircularQueue** ()
- void **push** (const T &t)
- void **pushCircular** (const T &t)
- T & **next** (void)
- T **pop** (void)
- T & **front** (void)
- const T & **front** (void) const
- T & **back** (void)
- const T & **back** (void) const
- bool **full** (void) const
- bool **empty** (void) const
- **UInt32** **size** (void) const
- **iterator** **begin** (void)
- **iterator** **end** (void)
- T & **operator[ ]** (**UInt32** idx) const
- T & **at** (**UInt32** idx) const

Private Attributes

- const **UInt32** **m_size**
- volatile **UInt32** **m_first**
- **UInt8** **padding1** [60]
- volatile **UInt32** **m_last**
- **UInt8** **padding2** [60]
- T *const **m_queue**

6.62.1 Detailed Description

template<class T>
class CircularQueue< T >

Definition at line 7 of file circular_queue.h.

6.62.2 Member Typedef Documentation

6.62.2.1 value_type

```
template<class T >
typedef T  CircularQueue< T >::  value_type
```

Definition at line 18 of file circular_queue.h.

6.62.3 Constructor & Destructor Documentation

6.62.3.1 CircularQueue() [1/2]

```
template<class T >
CircularQueue< T >::  CircularQueue (
              UInt32 size = 63 )
```

Definition at line 54 of file circular_queue.h.

6.62.3.2 CircularQueue() [2/2]

```
template<class T >
CircularQueue< T >::  CircularQueue (
             const  CircularQueue< T > & queue )
```

Definition at line 65 of file circular_queue.h.

6.62.3.3 ~CircularQueue()

```
template<class T >
CircularQueue< T >::~ CircularQueue
```

Definition at line 75 of file circular_queue.h.

6.62.4 Member Function Documentation

6.62.4.1 at()

```
template<class T >
T&  CircularQueue< T >::at (
              UInt32 idx ) const  [inline]
```

Definition at line 50 of file circular_queue.h.

Referenced by RobTimer::countOutstandingMemop(), RobSmtTimer::countOutstandingMemop(), RobTimer::doDispatch(), RobTimer::doIssue(), RobTimer::findCpiComponent(), RobSmtTimer::findCpiComponent(), CircularQueue< T >::iterator::operator*(), CircularQueue< T >::iterator::operator->(), RobTimer::printRob(), RobSmtTimer::printRob(), RobSmtTimer::tryDispatch(), and RobSmtTimer::tryIssue().

6.62.4.2 back() [1/2]

```
template<class T >
const T &  CircularQueue< T >::back (
             void  )
```

Definition at line 136 of file circular_queue.h.

6.62.4.3 back() [2/2]

```
template<class T >
const T&  CircularQueue< T >::back (
             void  ) const
```

6.62.4.4 begin()

```
template<class T >
iterator  CircularQueue< T >::begin (
             void  )  [inline]
```

Definition at line 47 of file circular_queue.h.

Referenced by RobTimer::~RobTimer().

6.62.4.5 empty()

```
template<class T >
bool  CircularQueue< T >::empty (
            void  ) const
```

Definition at line 159 of file circular_queue.h.

Referenced by MTCircularQueue< T >::push_locked().

6.62.4.6 end()

```
template<class T >
iterator  CircularQueue< T >::end (
            void  )  [inline]
```

Definition at line 48 of file circular_queue.h.

6.62.4.7 front() [1/2]

```
template<class T >
const T &  CircularQueue< T >::front (
            void  )
```

Definition at line 120 of file circular_queue.h.

Referenced by RobTimer::doCommit(), RobSmtTimer::doCommit(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), PerformanceModel::iterate(), RobSmtTimer::setDependencies(), and RobTimer::simulate().

6.62.4.8 front() [2/2]

```
template<class T >
const T&  CircularQueue< T >::front (
            void  ) const
```

6.62.4.9 full()

```
template<class T >
bool  CircularQueue< T >::full (
            void  ) const
```

Definition at line 152 of file circular_queue.h.

Referenced by MTCircularQueue< T >::pop_locked().

6.62.4.10 next()

```
template<class T >
T &  CircularQueue< T >::next (
             void  )
```

Definition at line 100 of file circular_queue.h.

Referenced by RobSmtTimer::pushInstructions(), and RobTimer::simulate().

6.62.4.11 operator[]()

```
template<class T >
T&  CircularQueue< T >::operator[] (
              UInt32 idx ) const  [inline]
```

Definition at line 49 of file circular_queue.h.

6.62.4.12 pop()

```
template<class T >
T  CircularQueue< T >::pop (
             void  )
```

Definition at line 110 of file circular_queue.h.

Referenced by RobTimer::doCommit(), RobSmtTimer::doCommit(), PerformanceModel::iterate(), and MTCircular↩ Queue< T >::pop_locked().

6.62.4.13 push()

```
template<class T >
void  CircularQueue< T >::push (
             const T & t )
```

Definition at line 82 of file circular_queue.h.

Referenced by MTCircularQueue< T >::push_locked(), PerformanceModel::queueInstruction(), and Performance↩ Model::queuePseudoInstruction().

6.62.4.14 pushCircular()

```
template<class T >
void  CircularQueue< T >::pushCircular (
            const T & t )
```

Definition at line 91 of file circular_queue.h.

6.62.4.15 size()

```
template<class T >
UInt32  CircularQueue< T >::size (
            void  ) const
```

Definition at line 166 of file circular_queue.h.

Referenced by CircularQueue< MemoryDependencies::Producer >::at(), RobSmtTimer::canExecute(), CircularQueue< MemoryDependencies::Producer >::CircularQueue(), RobTimer::doCommit(), RobSmtTimer::doCommit(), RobTimer::doDispatch(), CircularQueue< MemoryDependencies::Producer >::end(), RobTimer::execute(), RobTimer::findEntryBySequenceNumber(), RobSmtTimer::findEntryBySequenceNumber(), PerformanceModel::iterate(), RobTimer::printRob(), RobSmtTimer::printRob(), RobSmtTimer::setDependencies(), RobTimer::simulate(), RobSmtTimer::threadNumSurplusInstructions(), and RobSmtTimer::tryDispatch().

6.62.5 Member Data Documentation

6.62.5.1 m_first

```
template<class T >
volatile  UInt32  CircularQueue< T >::m_first  [private]
```

Definition at line 11 of file circular_queue.h.

6.62.5.2 m_last

```
template<class T >
volatile  UInt32  CircularQueue< T >::m_last  [private]
```

Definition at line 13 of file circular_queue.h.

Referenced by CircularQueue< MemoryDependencies::Producer >::operator[ ]().

6.62.5.3 m_queue

```
template<class T >
T* const  CircularQueue< T >::m_queue  [private]
```

Definition at line 15 of file circular_queue.h.

Referenced by CircularQueue< MemoryDependencies::Producer >::operator[ ]().

6.62.5.4 m_size

```
template<class T >
const  UInt32  CircularQueue< T >::m_size  [private]
```

Definition at line 10 of file circular_queue.h.

Referenced by CircularQueue< MemoryDependencies::Producer >::operator[ ]().

6.62.5.5 padding1

```
template<class T >
UInt8  CircularQueue< T >::padding1[60]  [private]
```

Definition at line 12 of file circular_queue.h.

6.62.5.6 padding2

```
template<class T >
UInt8  CircularQueue< T >::padding2[60]  [private]
```

Definition at line 14 of file circular_queue.h.

The documentation for this class was generated from the following file:

- common/misc/ **circular_queue.h**

6.63 ClockSkewMinimizationClient Class Reference

`#include <clock_skew_minimization_object.h>`

Inheritance diagram for ClockSkewMinimizationClient:

Public Member Functions

- virtual ~**ClockSkewMinimizationClient** ()
- virtual void **enable** ()=0
- virtual void **disable** ()=0
- virtual void **synchronize** (**SubsecondTime** time= **SubsecondTime::Zero**(), bool ignore_time=false, bool abort_func(void *)=NULL, void *abort_arg=NULL)=0

Static Public Member Functions

- static **ClockSkewMinimizationClient** * **create** (**Core** *core)

Protected Member Functions

- **ClockSkewMinimizationClient** ()

Additional Inherited Members

6.63.1 Detailed Description

Definition at line 23 of file clock_skew_minimization_object.h.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 ClockSkewMinimizationClient()

```
ClockSkewMinimizationClient::ClockSkewMinimizationClient ( )  [inline], [protected]
```

Definition at line 26 of file clock_skew_minimization_object.h.

6.63.2.2 ~ClockSkewMinimizationClient()

```
virtual ClockSkewMinimizationClient::~ClockSkewMinimizationClient ( )  [inline], [virtual]
```

Definition at line 29 of file clock_skew_minimization_object.h.

6.63.3 Member Function Documentation

6.63.3.1 create()

```
ClockSkewMinimizationClient * ClockSkewMinimizationClient::create (
            Core * core )  [static]
```

Definition at line 20 of file clock_skew_minimization_object.cc.

References ClockSkewMinimizationObject::BARRIER, and LOG_PRINT_ERROR.

Referenced by Core::Core().

6.63.3.2 disable()

```
virtual void ClockSkewMinimizationClient::disable ( )  [pure virtual]
```

Implemented in **BarrierSyncClient** (p. 90).

6.63.3.3 enable()

```
virtual void ClockSkewMinimizationClient::enable ( )  [pure virtual]
```

Implemented in **BarrierSyncClient** (p. 90).

6.63.3.4 synchronize()

```
virtual void ClockSkewMinimizationClient::synchronize (
             SubsecondTime time =  SubsecondTime::Zero(),
            bool ignore_time = false,
            bool  abort_funcvoid * = NULL,
            void * abort_arg = NULL )  [pure virtual]
```

Implemented in **BarrierSyncClient** (p. 90).

Referenced by SmtTimer::simulate(), and PerformanceModel::synchronize().

The documentation for this class was generated from the following files:

- common/system/ **clock_skew_minimization_object.h**
- common/system/ **clock_skew_minimization_object.cc**

6.64 ClockSkewMinimizationManager Class Reference

`#include <clock_skew_minimization_object.h>`

Inheritance diagram for ClockSkewMinimizationManager:

Public Member Functions

- virtual ~**ClockSkewMinimizationManager** ()
- virtual void **processSyncMsg** (**Byte** *msg)=0

Static Public Member Functions

- static **ClockSkewMinimizationManager** * **create** ()

Protected Member Functions

- **ClockSkewMinimizationManager** ()

Additional Inherited Members

6.64.1 Detailed Description

Definition at line 37 of file clock_skew_minimization_object.h.

6.64.2 Constructor & Destructor Documentation

6.64.2.1 ClockSkewMinimizationManager()

`ClockSkewMinimizationManager::ClockSkewMinimizationManager ( ) [inline], [protected]`

Definition at line 40 of file clock_skew_minimization_object.h.

6.64.2.2 ~ClockSkewMinimizationManager()

```
virtual ClockSkewMinimizationManager::~ClockSkewMinimizationManager ( )  [inline], [virtual]
```

Definition at line 43 of file clock_skew_minimization_object.h.

6.64.3 Member Function Documentation

6.64.3.1 create()

```
ClockSkewMinimizationManager * ClockSkewMinimizationManager::create ( )  [static]
```

Definition at line 36 of file clock_skew_minimization_object.cc.

References ClockSkewMinimizationObject::BARRIER, and LOG_PRINT_ERROR.

Referenced by Simulator::start().

6.64.3.2 processSyncMsg()

```
virtual void ClockSkewMinimizationManager::processSyncMsg (
             Byte * msg )  [pure virtual]
```

The documentation for this class was generated from the following files:

- common/system/ **clock_skew_minimization_object.h**
- common/system/ **clock_skew_minimization_object.cc**

6.65 ClockSkewMinimizationObject Class Reference

```
#include <clock_skew_minimization_object.h>
```

Inheritance diagram for ClockSkewMinimizationObject:

Public Types

- enum **Scheme** { **NONE** = 0, **BARRIER**, **NUM_SCHEMES** }

Static Public Member Functions

- static **Scheme** **parseScheme** (String scheme)

6.65.1 Detailed Description

Definition at line 10 of file clock_skew_minimization_object.h.

6.65.2 Member Enumeration Documentation

6.65.2.1 Scheme

enum **ClockSkewMinimizationObject::Scheme**

Enumerator

NONE	
BARRIER	
NUM_SCHEMES	

Definition at line 13 of file clock_skew_minimization_object.h.

6.65.3 Member Function Documentation

6.65.3.1 parseScheme()

ClockSkewMinimizationObject::Scheme ClockSkewMinimizationObject::parseScheme (
String *scheme*) [static]

Definition at line 9 of file clock_skew_minimization_object.cc.

References BARRIER, and config::Error().

Referenced by Config::Config().

The documentation for this class was generated from the following files:

- common/system/ **clock_skew_minimization_object.h**
- common/system/ **clock_skew_minimization_object.cc**

6.66 ClockSkewMinimizationServer Class Reference

`#include <clock_skew_minimization_object.h>`

Inheritance diagram for ClockSkewMinimizationServer:

Public Member Functions

- virtual ~**ClockSkewMinimizationServer** ()
- virtual void **synchronize** (**thread_id_t** thread_id, **SubsecondTime** time)=0
- virtual void **release** ()=0
- virtual void **advance** ()=0
- virtual void **setDisable** (bool disable)
- virtual void **setGroup** (**core_id_t** core_id, **core_id_t** master_core_id)=0
- virtual void **setFastForward** (bool fastforward, **SubsecondTime** next_barrier_time= **SubsecondTime::MaxTime**())=0
- virtual **SubsecondTime** **getGlobalTime** (bool upper_bound=false)
- virtual void **setBarrierInterval** (**SubsecondTime** barrier_interval)=0
- virtual **SubsecondTime** **getBarrierInterval** () const =0
- virtual void **printState** (void)

Static Public Member Functions

- static **ClockSkewMinimizationServer** * **create** ()

Protected Member Functions

- **ClockSkewMinimizationServer** ()

Additional Inherited Members

6.66.1 Detailed Description

Definition at line 49 of file clock_skew_minimization_object.h.

6.66.2 Constructor & Destructor Documentation

6.66.2.1 ClockSkewMinimizationServer()

```
ClockSkewMinimizationServer::ClockSkewMinimizationServer ( )  [inline], [protected]
```

Definition at line 52 of file clock_skew_minimization_object.h.

6.66.2.2 ~ClockSkewMinimizationServer()

```
virtual ClockSkewMinimizationServer::~ClockSkewMinimizationServer ( )  [inline], [virtual]
```

Definition at line 55 of file clock_skew_minimization_object.h.

6.66.3 Member Function Documentation

6.66.3.1 advance()

```
virtual void ClockSkewMinimizationServer::advance ( )  [pure virtual]
```

Implemented in **BarrierSyncServer** (p. 93).

6.66.3.2 create()

```
ClockSkewMinimizationServer * ClockSkewMinimizationServer::create ( )  [static]
```

Definition at line 52 of file clock_skew_minimization_object.cc.

References ClockSkewMinimizationObject::BARRIER, and LOG_PRINT_ERROR.

Referenced by Simulator::start().

6.66.3.3 getBarrierInterval()

```
virtual SubsecondTime ClockSkewMinimizationServer::getBarrierInterval ( ) const  [pure virtual]
```

Implemented in **BarrierSyncServer** (p. 94).

6.66.3.4 getGlobalTime()

```
SubsecondTime ClockSkewMinimizationServer::getGlobalTime (
            bool upper_bound = false )  [virtual]
```

Reimplemented in **BarrierSyncServer** (p. 94).

Definition at line 68 of file clock_skew_minimization_object.cc.

References LOG_PRINT_ERROR.

6.66.3.5 printState()

```
virtual void ClockSkewMinimizationServer::printState (
            void  )  [inline], [virtual]
```

Reimplemented in **BarrierSyncServer** (p. 96).

Definition at line 68 of file clock_skew_minimization_object.h.

6.66.3.6 release()

```
virtual void ClockSkewMinimizationServer::release ( )  [pure virtual]
```

Implemented in **BarrierSyncServer** (p. 96).

6.66.3.7 setBarrierInterval()

```
virtual void ClockSkewMinimizationServer::setBarrierInterval (
             SubsecondTime barrier_interval )  [pure virtual]
```

Implemented in **BarrierSyncServer** (p. 97).

6.66.3.8 setDisable()

```
virtual void ClockSkewMinimizationServer::setDisable (
            bool disable )  [inline], [virtual]
```

Reimplemented in **BarrierSyncServer** (p. 97).

Definition at line 61 of file clock_skew_minimization_object.h.

Referenced by Simulator::setInstrumentationMode().

6.66.3.9 setFastForward()

```
virtual void ClockSkewMinimizationServer::setFastForward (
            bool fastforward,
             SubsecondTime next_barrier_time =  SubsecondTime::MaxTime() )  [pure virtual]
```

Implemented in **BarrierSyncServer** (p. 97).

6.66.3.10 setGroup()

```
virtual void ClockSkewMinimizationServer::setGroup (
             core_id_t core_id,
             core_id_t master_core_id )  [pure virtual]
```

Implemented in **BarrierSyncServer** (p. 97).

6.66.3.11 synchronize()

```
virtual void ClockSkewMinimizationServer::synchronize (
             thread_id_t thread_id,
             SubsecondTime time )  [pure virtual]
```

Implemented in **BarrierSyncServer** (p. 98).

The documentation for this class was generated from the following files:

- common/system/ **clock_skew_minimization_object.h**
- common/system/ **clock_skew_minimization_object.cc**

6.67 CoherencyProtocol Class Reference

`#include <coherency_protocol.h>`

Public Types

- enum **type_t** { **MSI**, **MESI**, **MESIF** }

6.67.1 Detailed Description

Definition at line 4 of file coherency_protocol.h.

6.67.2 Member Enumeration Documentation

6.67.2.1 type_t

```
enum CoherencyProtocol::type_t
```

Enumerator

MSI	
MESI	
MESIF	

Definition at line 7 of file coherency_protocol.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/directory_schemes/ **coherency_protocol.h**

6.68 ComponentBandwidth Class Reference

`#include <subsecond_time.h>`

Public Member Functions

- **ComponentBandwidth** (float bw_in_bits_per_ns)
- **SubsecondTime getLatency** (uint64_t bits_transmitted) const
- **SubsecondTime getRoundedLatency** (uint64_t bits_transmitted) const

Private Member Functions

- **ComponentBandwidth** ()

Private Attributes

- uint64_t **m_bw_in_bits_per_us**

Friends

- std::ostream & **operator**<< (std::ostream &os, const **ComponentBandwidth** &time)

6.68.1 Detailed Description

Definition at line 386 of file subsecond_time.h.

6.68.2 Constructor & Destructor Documentation

6.68.2.1 ComponentBandwidth() [1/2]

```
ComponentBandwidth::ComponentBandwidth (
            float bw_in_bits_per_ns )  [inline]
```

Definition at line 389 of file subsecond_time.h.

6.68.2.2 ComponentBandwidth() [2/2]

```
ComponentBandwidth::ComponentBandwidth ( )  [inline], [private]
```

Definition at line 412 of file subsecond_time.h.

6.68.3 Member Function Documentation

6.68.3.1 getLatency()

```
SubsecondTime ComponentBandwidth::getLatency (
            uint64_t bits_transmitted ) const  [inline]
```

Definition at line 395 of file subsecond_time.h.

References m_bw_in_bits_per_us, and SubsecondTime::US().

Referenced by NetworkModelBusGlobal::useBus().

6.68.3.2 getRoundedLatency()

```
SubsecondTime ComponentBandwidth::getRoundedLatency (
            uint64_t bits_transmitted ) const  [inline]
```

Definition at line 401 of file subsecond_time.h.

References m_bw_in_bits_per_us, and SubsecondTime::US().

Referenced by NucaCache::accessDataArray(), DramCache::accessDataArray(), DramCache::DramCache(), DramPerfModelConstant::DramPerfModelConstant(), DramPerfModelNormal::DramPerfModelNormal(), DramPerfModelReadWrite::DramPerfModelReadWrite(), DramPerfModelConstant::getAccessLatency(), DramPerfModelNormal::getAccessLatency(), DramPerfModelReadWrite::getAccessLatency(), and NucaCache::NucaCache().

6.68.4 Friends And Related Function Documentation

6.68.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const  ComponentBandwidth & time )  [friend]
```

Definition at line 415 of file subsecond_time.h.

6.68.5 Member Data Documentation

6.68.5.1 m_bw_in_bits_per_us

```
uint64_t ComponentBandwidth::m_bw_in_bits_per_us  [private]
```

Definition at line 409 of file subsecond_time.h.

Referenced by getLatency(), getRoundedLatency(), and operator<<().

The documentation for this class was generated from the following file:

- common/misc/ **subsecond_time.h**

6.69 ComponentBandwidthPerCycle Class Reference

```
#include <subsecond_time.h>
```

Public Member Functions

- **ComponentBandwidthPerCycle** (const **ComponentPeriod** *period, uint64_t bw_in_bits_per_cycle)
- **ComponentBandwidthPerCycle** ()
- bool **isInfinite** () const
- **SubsecondTime** **getLatency** (uint64_t bits_transmitted) const
- **SubsecondTime** **getRoundedLatency** (uint64_t bits_transmitted) const
- **SubsecondTime** **getPeriod** (void) const

Private Attributes

- const **ComponentPeriod** * **m_period**
- uint64_t **m_bw_in_bits_per_cycle**

Friends

- std::ostream & **operator<<** (std::ostream &os, const **ComponentBandwidthPerCycle** &time)

6.69.1 Detailed Description

Definition at line 425 of file subsecond_time.h.

6.69.2 Constructor & Destructor Documentation

6.69.2.1 ComponentBandwidthPerCycle() [1/2]

```
ComponentBandwidthPerCycle::ComponentBandwidthPerCycle (
            const  ComponentPeriod * period,
            uint64_t bw_in_bits_per_cycle )  [inline]
```

Definition at line 428 of file subsecond_time.h.

6.69.2.2 ComponentBandwidthPerCycle() [2/2]

```
ComponentBandwidthPerCycle::ComponentBandwidthPerCycle ( )  [inline]
```

Definition at line 434 of file subsecond_time.h.

6.69.3 Member Function Documentation

6.69.3.1 getLatency()

```
SubsecondTime ComponentBandwidthPerCycle::getLatency (
            uint64_t bits_transmitted ) const  [inline]
```

Definition at line 445 of file subsecond_time.h.

References m_bw_in_bits_per_cycle.

6.69.3.2 getPeriod()

```
SubsecondTime ComponentBandwidthPerCycle::getPeriod (
            void  ) const  [inline]
```

Definition at line 455 of file subsecond_time.h.

Referenced by NetworkModelEMeshHopByHop::createQueueModels().

6.69.3.3 getRoundedLatency()

```
SubsecondTime ComponentBandwidthPerCycle::getRoundedLatency (
            uint64_t bits_transmitted ) const  [inline]
```

Definition at line 450 of file subsecond_time.h.

References m_bw_in_bits_per_cycle.

Referenced by NetworkModelEMeshHopByHop::computeProcessingTime(), NetworkModelEMeshHopCounter↩ ::computeSerializationLatency(), and ParametricDramDirectoryMSI::CacheCntlr::copyDataFromNextLevel().

6.69.3.4 isInfinite()

```
bool ComponentBandwidthPerCycle::isInfinite ( ) const  [inline]
```

Definition at line 438 of file subsecond_time.h.

References m_bw_in_bits_per_cycle.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::copyDataFromNextLevel().

6.69.4 Friends And Related Function Documentation

6.69.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const  ComponentBandwidthPerCycle & time )  [friend]
```

Definition at line 467 of file subsecond_time.h.

6.69.5 Member Data Documentation

6.69.5.1 m_bw_in_bits_per_cycle

```
uint64_t ComponentBandwidthPerCycle::m_bw_in_bits_per_cycle  [private]
```

Definition at line 463 of file subsecond_time.h.

Referenced by getLatency(), getRoundedLatency(), isInfinite(), and operator<<().

6.69.5.2 m_period

```
const ComponentPeriod* ComponentBandwidthPerCycle::m_period [private]
```

Definition at line 462 of file subsecond_time.h.

Referenced by operator<<().

The documentation for this class was generated from the following file:

- common/misc/ **subsecond_time.h**

6.70 ComponentLatency Class Reference

```
#include <subsecond_time.h>
```

Public Member Functions

- **ComponentLatency** (const **ComponentPeriod** *period, uint64_t fixed_cycle_latency)
- **SubsecondTime getLatency** () const
- **SubsecondTime getPeriod** (void) const
- **ComponentLatency** & **operator+=** (uint64_t rhs)

Private Member Functions

- **ComponentLatency** ()

Private Attributes

- const **ComponentPeriod** * **m_period**
- uint64_t **m_fixed_cycle_latency**

Friends

- std::ostream & **operator<<** (std::ostream &os, const **ComponentLatency** &time)

6.70.1 Detailed Description

Definition at line 479 of file subsecond_time.h.

6.70.2 Constructor & Destructor Documentation

6.70.2.1 ComponentLatency() [1/2]

```
ComponentLatency::ComponentLatency (
            const  ComponentPeriod * period,
            uint64_t fixed_cycle_latency )  [inline]
```

Definition at line 482 of file subsecond_time.h.

6.70.2.2 ComponentLatency() [2/2]

```
ComponentLatency::ComponentLatency ( )  [inline], [private]
```

Definition at line 508 of file subsecond_time.h.

6.70.3 Member Function Documentation

6.70.3.1 getLatency()

```
SubsecondTime ComponentLatency::getLatency ( ) const  [inline]
```

Definition at line 487 of file subsecond_time.h.

References m_fixed_cycle_latency.

Referenced by FastNehalem::Dram::access(), FastNehalem::Cache< assoc, size_kb >::access(), NucaCache::accessDataArray(), ParametricDramDirectoryMSI::MemoryManager::accessTLB(), NetworkModelEMeshHopByHop::computeLatency(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), CachePerfModelSequential::getLatency(), CachePerfModelParallel::getLatency(), FastforwardPerformanceModel::handleBranchMispredict(), ParametricDramDirectoryMSI::CacheCntlr::incrementQBSLookupCost(), NetworkModelEMeshHopByHop::processReceivedPacket(), NucaCache::read(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::replaceDirectoryEntry(), NetworkModelEMeshHopCounter::routePacket(), NetworkModelMagic::routePacket(), ParametricDramDirectoryMSI::CacheCntlr::updateCacheBlock(), and NucaCache::write().

6.70.3.2 getPeriod()

```
SubsecondTime ComponentLatency::getPeriod (
            void  ) const  [inline]
```

Definition at line 491 of file subsecond_time.h.

6.70.3.3 operator+=()

```
ComponentLatency& ComponentLatency::operator+= (
            uint64_t rhs )  [inline]
```

Definition at line 497 of file subsecond_time.h.

References m_fixed_cycle_latency.

6.70.4 Friends And Related Function Documentation

6.70.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const  ComponentLatency & time )  [friend]
```

Definition at line 513 of file subsecond_time.h.

6.70.5 Member Data Documentation

6.70.5.1 m_fixed_cycle_latency

```
uint64_t ComponentLatency::m_fixed_cycle_latency  [private]
```

Definition at line 506 of file subsecond_time.h.

Referenced by getLatency(), operator+=(), and operator<<().

6.70.5.2 m_period

```
const  ComponentPeriod* ComponentLatency::m_period  [private]
```

Definition at line 505 of file subsecond_time.h.

Referenced by operator<<().

The documentation for this class was generated from the following file:

- common/misc/ **subsecond_time.h**

6.71 ComponentPeriod Class Reference

`#include <subsecond_time.h>`

Public Member Functions

- **ComponentPeriod** (const **ComponentPeriod** &_p)
- void **setPeriodFromFreqHz** (uint64_t freq_in_hz)
- **SubsecondTime** **getPeriod** (void) const
- **UInt64** **getPeriodInFreqMHz** (void) const
- **ComponentPeriod** & **operator*=** (uint64_t rhs)
- **operator SubsecondTime** () const

Static Public Member Functions

- static **ComponentPeriod** **fromFreqHz** (uint64_t freq_in_hz)

Private Member Functions

- **ComponentPeriod** ()
- **ComponentPeriod** (uint64_t _time)
- **ComponentPeriod** (**SubsecondTime** &_time)

Private Attributes

- **SubsecondTime** **m_period**

Friends

- std::ostream & **operator<<** (std::ostream &os, const **ComponentPeriod** &period)

6.71.1 Detailed Description

Definition at line 289 of file subsecond_time.h.

6.71.2 Constructor & Destructor Documentation

6.71.2.1 ComponentPeriod() [1/4]

```
ComponentPeriod::ComponentPeriod (
            const  ComponentPeriod & _p )  [inline]
```

Definition at line 293 of file subsecond_time.h.

6.71.2.2 ComponentPeriod() [2/4]

```
ComponentPeriod::ComponentPeriod ( )  [inline], [private]
```

Definition at line 334 of file subsecond_time.h.

Referenced by fromFreqHz().

6.71.2.3 ComponentPeriod() [3/4]

```
ComponentPeriod::ComponentPeriod (
            uint64_t _time )  [inline], [private]
```

Definition at line 336 of file subsecond_time.h.

6.71.2.4 ComponentPeriod() [4/4]

```
ComponentPeriod::ComponentPeriod (
             SubsecondTime & _time )  [inline], [private]
```

Definition at line 339 of file subsecond_time.h.

6.71.3 Member Function Documentation

6.71.3.1 fromFreqHz()

```
static  ComponentPeriod ComponentPeriod::fromFreqHz (
            uint64_t freq_in_hz )  [inline], [static]
```

Definition at line 297 of file subsecond_time.h.

References ComponentPeriod(), and SubsecondTime::SEC().

Referenced by DvfsManager::DvfsManager(), NetworkModelBusGlobal::NetworkModelBusGlobal(), and MagicServer::setFrequency().

6.71.3.2 getPeriod()

```
SubsecondTime ComponentPeriod::getPeriod (
            void ) const [inline]
```

Definition at line 309 of file subsecond_time.h.

References m_period.

Referenced by DynamicInstruction::accessMemory(), PeriodicSampling::callbackDetailed(), IntervalTimer::dispatchWindow(), getFrequency(), MicroOpPerformanceModel::handleInstruction(), TraceThread::handleInstructionCountFunc(), IntervalTimer::issueMemOp(), operator<<(), DvfsManager::setCoreDomain(), and LoopTracer::traceInstruction().

6.71.3.3 getPeriodInFreqMHz()

```
UInt64 ComponentPeriod::getPeriodInFreqMHz (
            void ) const [inline]
```

Definition at line 311 of file subsecond_time.h.

References m_period, SubsecondTime::m_time, and SubsecondTime::US_1.

Referenced by MagicServer::getFrequency().

6.71.3.4 operator SubsecondTime()

```
ComponentPeriod::operator SubsecondTime ( ) const [inline]
```

Definition at line 326 of file subsecond_time.h.

References m_period.

6.71.3.5 operator*=()

```
ComponentPeriod& ComponentPeriod::operator*= (
            uint64_t rhs ) [inline]
```

Definition at line 317 of file subsecond_time.h.

References m_period.

6.71.3.6 setPeriodFromFreqHz()

```
void ComponentPeriod::setPeriodFromFreqHz (
            uint64_t freq_in_hz )  [inline]
```

Definition at line 304 of file subsecond_time.h.

References m_period, and SubsecondTime::SEC().

6.71.4 Friends And Related Function Documentation

6.71.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const  ComponentPeriod & period )  [friend]
```

Definition at line 355 of file subsecond_time.h.

6.71.5 Member Data Documentation

6.71.5.1 m_period

```
SubsecondTime ComponentPeriod::m_period  [private]
```

Definition at line 343 of file subsecond_time.h.

Referenced by ComponentTime::addLatency(), getPeriod(), getPeriodInFreqMHz(), operator SubsecondTime(), operator*=(), ComponentTime::operator+=(), operator<<(), and setPeriodFromFreqHz().

The documentation for this class was generated from the following file:

- common/misc/ **subsecond_time.h**

6.72 ComponentTime Class Reference

```
#include <subsecond_time.h>
```

Public Member Functions

- void **addCycleLatency** (uint64_t num_cycles)
- void **addLatency** (**SubsecondTime** time_to_add)
- void **addLatency** (const **ComponentTime** &time_to_add)
- **ComponentTime** & **operator+=** (const **ComponentTime** &rhs)
- **ComponentTime** & **operator+=** (const **SubsecondTime** &rhs)
- **ComponentTime** **operator+** (const **SubsecondTime** &rhs)
- **ComponentTime** **operator+** (uint64_t rhs)
- void **reset** ()
- **ComponentTime** **getLatencyGenerator** (void) const
- **SubsecondTime** **getElapsedTime** (void) const
- **UInt64** **getCycleCount** (void) const
- **SubsecondTime** **getPeriod** (void) const
- void **setElapsedTime** (**SubsecondTime** new_time)
- **ComponentTime** (const **ComponentPeriod** *_base_period)
- **operator const ComponentPeriod** * () const
- **operator const SubsecondTime** () const

Private Member Functions

- **ComponentTime** ()

Private Attributes

- const **ComponentPeriod** * **m_period**
- **SubsecondTime** **m_time**

Friends

- std::ostream & **operator**<< (std::ostream &os, const **ComponentTime** &time)

6.72.1 Detailed Description

Definition at line 527 of file subsecond_time.h.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 ComponentTime() [1/2]

```
ComponentTime::ComponentTime (
            const  ComponentPeriod * _base_period )  [inline]
```

Definition at line 601 of file subsecond_time.h.

6.72.2.2 ComponentTime() [2/2]

```
ComponentTime::ComponentTime ( )  [inline], [private]
```

Definition at line 625 of file subsecond_time.h.

Referenced by getLatencyGenerator().

6.72.3 Member Function Documentation

6.72.3.1 addCycleLatency()

```
void ComponentTime::addCycleLatency (
            uint64_t num_cycles )  [inline]
```

Definition at line 533 of file subsecond_time.h.

References m_time.

Referenced by MicroOpPerformanceModel::handleInstruction(), and operator+().

6.72.3.2 addLatency() [1/2]

```
void ComponentTime::addLatency (
            const  ComponentTime & time_to_add )  [inline]
```

Definition at line 542 of file subsecond_time.h.

References ComponentPeriod::m_period, m_period, and m_time.

6.72.3.3 addLatency() [2/2]

```
void ComponentTime::addLatency (
             SubsecondTime time_to_add )  [inline]
```

Definition at line 538 of file subsecond_time.h.

References m_time.

Referenced by OneIPCPerformanceModel::handleInstruction(), MicroOpPerformanceModel::handleInstruction(), PerformanceModel::incrementElapsedTime(), PerformanceModel::incrementIdleElapsedTime(), and operator+().

6.72.3.4 getCycleCount()

```
UInt64 ComponentTime::getCycleCount (
            void ) const [inline]
```

Definition at line 587 of file subsecond_time.h.

References SubsecondTime::divideRounded(), m_period, and m_time.

Referenced by IntervalPerformanceModel::notifyElapsedTimeUpdate().

6.72.3.5 getElapsedTime()

```
SubsecondTime ComponentTime::getElapsedTime (
            void ) const [inline]
```

Definition at line 583 of file subsecond_time.h.

References m_time.

Referenced by RobTimer::doIssue(), PerformanceModel::getElapsedTime(), PerformanceModel::getNonIdle↩ ElapsedTime(), OneIPCPerformanceModel::handleInstruction(), MicroOpPerformanceModel::handleInstruction(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), makeStatsValue< ComponentTime >(), Rob↩ PerformanceModel::notifyElapsedTimeUpdate(), RobSmtPerformanceModel::notifyElapsedTimeUpdate(), Rob↩ SmtTimer::synchronize(), and RobSmtTimer::tryIssue().

6.72.3.6 getLatencyGenerator()

```
ComponentTime ComponentTime::getLatencyGenerator (
            void ) const [inline]
```

Definition at line 579 of file subsecond_time.h.

References ComponentTime(), and m_period.

Referenced by OneIPCPerformanceModel::handleInstruction(), and MicroOpPerformanceModel::handle↩ Instruction().

6.72.3.7 getPeriod()

```
SubsecondTime ComponentTime::getPeriod (
            void ) const [inline]
```

Definition at line 591 of file subsecond_time.h.

Referenced by RobSmtTimer::doDispatch(), RobTimer::execute(), RobSmtTimer::executeCycle(), Rob↩ Timer::issueInstruction(), RobSmtTimer::issueInstruction(), RobPerformanceModel::simulate(), and RobSmt↩ PerformanceModel::simulate().

6.72.3.8 operator const ComponentPeriod ∗()

```
ComponentTime::operator const  ComponentPeriod * ( ) const  [inline]
```

Definition at line 608 of file subsecond_time.h.

References m_period.

6.72.3.9 operator const SubsecondTime()

```
ComponentTime::operator const  SubsecondTime ( ) const  [inline]
```

Definition at line 615 of file subsecond_time.h.

References m_time.

6.72.3.10 operator+() [1/2]

```
ComponentTime ComponentTime::operator+ (
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 561 of file subsecond_time.h.

References addLatency().

6.72.3.11 operator+() [2/2]

```
ComponentTime ComponentTime::operator+ (
            uint64_t rhs )  [inline]
```

Definition at line 567 of file subsecond_time.h.

References addCycleLatency().

6.72.3.12 operator+=() [1/2]

```
ComponentTime& ComponentTime::operator+= (
            const  ComponentTime & rhs )  [inline]
```

Definition at line 550 of file subsecond_time.h.

References ComponentPeriod::m_period, m_period, and m_time.

6.72.3.13 operator+=() [2/2]

```
ComponentTime& ComponentTime::operator+= (
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 556 of file subsecond_time.h.

References m_time.

6.72.3.14 reset()

```
void ComponentTime::reset ( )  [inline]
```

Definition at line 574 of file subsecond_time.h.

References m_time.

6.72.3.15 setElapsedTime()

```
void ComponentTime::setElapsedTime (
              SubsecondTime new_time )  [inline]
```

Definition at line 595 of file subsecond_time.h.

References m_time.

Referenced by RobContentionBoomV1::initCycle(), RobContentionNehalem::initCycle(), RobTimer::synchronize(), and RobSmtTimer::synchronize().

6.72.4 Friends And Related Function Documentation

6.72.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const  ComponentTime & time )  [friend]
```

Definition at line 631 of file subsecond_time.h.

6.72.5 Member Data Documentation

6.72.5.1 m_period

```
const ComponentPeriod* ComponentTime::m_period [private]
```

Definition at line 622 of file subsecond_time.h.

Referenced by addLatency(), getCycleCount(), getLatencyGenerator(), operator const ComponentPeriod *(), operator+=(), and operator<<().

6.72.5.2 m_time

```
SubsecondTime ComponentTime::m_time [private]
```

Definition at line 623 of file subsecond_time.h.

Referenced by addCycleLatency(), addLatency(), getCycleCount(), getElapsedTime(), operator const SubsecondTime(), operator+=(), operator<<(), reset(), and setElapsedTime().

The documentation for this class was generated from the following file:

- common/misc/ **subsecond_time.h**

6.73 ConditionVariable Class Reference

```
#include <cond.h>
```

Public Member Functions

- **ConditionVariable** ()
- ~**ConditionVariable** ()
- void **wait** (**Lock** &_lock, **UInt64** timeout_ns=0)
- void **signal** ()
- void **broadcast** ()

Private Attributes

- int **m_futx**
- **Lock** **m_lock**

6.73.1 Detailed Description

Definition at line 11 of file cond.h.

6.73.2 Constructor & Destructor Documentation

6.73.2.1 ConditionVariable()

```
ConditionVariable::ConditionVariable ( )
```

Definition at line 10 of file cond.cc.

References TotalTimer::getTimerByStacktrace(), and itostr().

6.73.2.2 ~ConditionVariable()

```
ConditionVariable::~ConditionVariable ( )
```

Definition at line 18 of file cond.cc.

6.73.3 Member Function Documentation

6.73.3.1 broadcast()

```
void ConditionVariable::broadcast ( )
```

Definition at line 63 of file cond.cc.

References TLock< T_LockCreator >::acquire(), FUTEX_PRIVATE_FLAG, m_futx, m_lock, and TLock< T_Lock↩ Creator >::release().

Referenced by Network::netPullFromTransport(), SmTransport::SmNode::send(), and Barrier::wait().

6.73.3.2 signal()

```
void ConditionVariable::signal ( )
```

Definition at line 52 of file cond.cc.

References TLock< T_LockCreator >::acquire(), FUTEX_PRIVATE_FLAG, m_futx, m_lock, and TLock< T_Lock↩ Creator >::release().

Referenced by SmtTimer::barrierRelease(), SmtTimer::disable(), Thread::signal(), and SmtTimer::threadMigrate().

6.73.3.3 wait()

```
void ConditionVariable::wait (
              Lock & _lock,
              UInt64 timeout_ns = 0 )
```

Definition at line 25 of file cond.cc.

References TLock< T_LockCreator >::acquire(), FUTEX_PRIVATE_FLAG, m_futx, m_lock, and TLock< T_Lock↩ Creator >::release().

Referenced by SmtTimer::barrier(), Network::netRecv(), Barrier::wait(), and Thread::wait().

6.73.4 Member Data Documentation

6.73.4.1 m_futx

```
int ConditionVariable::m_futx  [private]
```

Definition at line 23 of file cond.h.

Referenced by broadcast(), signal(), and wait().

6.73.4.2 m_lock

```
Lock ConditionVariable::m_lock  [private]
```

Definition at line 24 of file cond.h.

Referenced by broadcast(), signal(), and wait().

The documentation for this class was generated from the following files:

- common/misc/ **cond.h**
- common/misc/ **cond.cc**

6.74 SimCond::CondWaiter Class Reference

Public Member Functions

- **CondWaiter** (**thread_id_t** thread_id, **SimMutex** ∗mutex)

Public Attributes

- **thread_id_t m_thread_id**
- **SimMutex** * **m_mutex**

6.74.1 Detailed Description

Definition at line 56 of file sync_server.h.

6.74.2 Constructor & Destructor Documentation

6.74.2.1 CondWaiter()

```
SimCond::CondWaiter::CondWaiter (
              thread_id_t thread_id,
              SimMutex * mutex )  [inline]
```

Definition at line 59 of file sync_server.h.

6.74.3 Member Data Documentation

6.74.3.1 m_mutex

```
SimMutex* SimCond::CondWaiter::m_mutex
```

Definition at line 62 of file sync_server.h.

Referenced by SimCond::broadcast(), and SimCond::signal().

6.74.3.2 m_thread_id

```
thread_id_t SimCond::CondWaiter::m_thread_id
```

Definition at line 61 of file sync_server.h.

Referenced by SimCond::broadcast(), and SimCond::signal().

The documentation for this class was generated from the following file:

- common/system/ **sync_server.h**

6.75 config::Config Class Reference

Config (p. 328): A class for managing the interface to persistent configuration entries defined at runtime. This class is used to manage a configuration interface. It is the base class for which different back ends will derive from.

`#include <config.hpp>`

Inheritance diagram for config::Config:

Public Member Functions

- **Config** (bool case_sensitive=false)
- **Config** (const **Section** &root, bool case_sensitive=false)
- virtual ~**Config** ()
- virtual void **saveAs** (const String &path)

  A function for saving the entire configuration tree to the specified path. This function is responsible for walking through the entire configuration tree (starting at the root) and outputing an appropriate text representation than can then be re-read by an appropriate configuration class. Note: In the case of a write-through situation (as is the case with the windows registry), this function is unnecessary.
- void **load** (const String &path)

  *A function to convert from external representation to in-memory representation This function sets the member variable m_path and then calls the **loadConfig()** (p. 338) function which must be implemented by the derived back-end implementation.*
- void **Save** ()

  *A function which will save the in-memory representation to the external representation that was specified with the **load()** (p. 337) function.*
- void **clear** ()
- bool **hasKey** (const String &path, **UInt64** index=UINT64_MAX)
- virtual void **set** (const String &path, const String &new_value)

  A function that will save a given value to key at the specified path.
- virtual void **set** (const String &path, **SInt64** new_value)
- virtual void **set** (const String &path, double new_value)
- const **Section** & **getSection** (const String &path)

  Returns a reference to the section at the given path.
- const **Section** & **getRoot** ()

  Returns a reference to the root section of the configuration tree.
- const **Section** & **addSection** (const String &path)

  ***addSection()** (p. 331) adds the specified path as a new section, creating each entry in the path along the way.*
- bool **getBool** (const String &path)

  Look up the key at the given path, and return the value of that key as a bool.
- bool **getBoolArray** (const String &path, **UInt64** index)
- bool **getBoolDefault** (const String &path, bool defaultValue)
- **SInt64** **getInt** (const String &path)

  Look up the key at the given path, and return the value of that key as a bool.
- **SInt64** **getIntArray** (const String &path, **UInt64** index)
- const String **getString** (const String &path)

Look up the key at the given path, and return the value of that key as a bool.

- const String **getStringArray** (const String &path, **UInt64** index)
- const String **get** (const String &path)

  *Same as **getString()** (p. 336)*
- double **getFloat** (const String &path)

  Look up the key at the given path, and return the value of that key as a bool.
- double **getFloatArray** (const String &path, **UInt64** index)
- String **showTree** (const **Section** &current, int depth=0)

  Returns a string representation of the tree starting at the specified section.
- String **showFullTree** ()

  Returns a string representation of the loaded configuration tree.
- const **Key** & **addKey** (const String &path, const String &new_key, **UInt64** index=UINT64_MAX)
- const **Key** & **addKey** (const String &path, const **SInt64** new_key, **UInt64** index=UINT64_MAX)
- const **Key** & **addKey** (const String &path, const double new_key, **UInt64** index=UINT64_MAX)

Protected Member Functions

- virtual void **loadConfig** ()=0
- **Section** & **getSection_unsafe** (String const &path)
- **Section** & **getRoot_unsafe** ()
- **Key** & **getKey_unsafe** (String const &path)

Protected Attributes

- bool **m_case_sensitive**
- **Section** **m_root**
- String **m_path**

Private Member Functions

- template<class V >
  const **Key** & **addKeyInternal** (const String &path, const V &new_key, **UInt64** index)
- const **Key** & **getKey** (const String &path, **UInt64** index)
- template<class V >
  const **Key** & **getKey** (const String &path, const V &default_val, **UInt64** index)

Static Private Member Functions

- static **PathPair** **splitPath** (const String &path)
- static **PathPair** **splitPathElements** (const String &path, **PathElementList** &path_elements)
- static bool **isLeaf** (const String &path)

6.75.1 Detailed Description

Config (p. 328): A class for managing the interface to persistent configuration entries defined at runtime. This class is used to manage a configuration interface. It is the base class for which different back ends will derive from.

Definition at line 37 of file config.hpp.

6.75.2 Constructor & Destructor Documentation

6.75.2.1 Config() [1/2]

```
config::Config::Config (
            bool case_sensitive = false )  [inline]
```

Definition at line 40 of file config.hpp.

6.75.2.2 Config() [2/2]

```
config::Config::Config (
            const  Section & root,
            bool case_sensitive = false )  [inline]
```

Definition at line 41 of file config.hpp.

6.75.2.3 ~Config()

```
virtual config::Config::~Config ( )  [inline], [virtual]
```

Definition at line 42 of file config.hpp.

6.75.3 Member Function Documentation

6.75.3.1 addKey() [1/3]

```
const  Key& config::Config::addKey (
            const String & path,
            const double new_key,
             UInt64 index = UINT64_MAX )  [inline]
```

Definition at line 161 of file config.hpp.

References addKeyInternal().

6.75.3.2 addKey() [2/3]

```
const  Key& config::Config::addKey (
             const String & path,
             const  SInt64 new_key,
              UInt64 index = UINT64_MAX )  [inline]
```

Definition at line 158 of file config.hpp.

References addKeyInternal().

6.75.3.3 addKey() [3/3]

```
const  Key& config::Config::addKey (
             const String & path,
             const String & new_key,
              UInt64 index = UINT64_MAX )  [inline]
```

addKey() (p. 331) Adds the specified path as a new key (with the given value), creating each entry in the path along the way.

Parameters

path	The path to the new key
new_key	The value for the newly created key

Definition at line 155 of file config.hpp.

References addKeyInternal().

Referenced by set().

6.75.3.4 addKeyInternal()

```
template<class V >
const  Key & Config::addKeyInternal (
             const String & path,
             const V & new_key,
              UInt64 index )  [private]
```

Definition at line 167 of file config.cpp.

References config::Section::addKey(), getSection_unsafe(), isLeaf(), m_root, and splitPath().

Referenced by addKey().

6.75.3.5 addSection()

```
const  Section & Config::addSection (
             const String & path )
```

addSection() (p. 331) adds the specified path as a new section, creating each entry in the path along the way.

Definition at line 135 of file config.cpp.

References config::Section::addSubsection(), getSection_unsafe(), and splitPath().

6.75.3.6 clear()

```
void Config::clear ( )
```

Definition at line 76 of file config.cpp.

References config::Section::clear(), and m_root.

6.75.3.7 get()

```
const String config::Config::get (
             const String & path )  [inline]
```

Same as **getString()** (p. 336)

Definition at line 133 of file config.hpp.

References getString().

6.75.3.8 getBool()

```
bool config::Config::getBool (
             const String & path )  [inline]
```

Look up the key at the given path, and return the value of that key as a bool.

Parameters

path	- Path for key to look up

Exceptions

KeyNotFound *(p. 674)*	is thrown if the specified path doesn't exist.

Definition at line 105 of file config.hpp.

References getBoolArray().

Referenced by getBoolDefault(), main(), and Simulator::start().

6.75.3.9 getBoolArray()

```
bool Config::getBoolArray (
             const String & path,
              UInt64 index )
```

Definition at line 223 of file config.cpp.

References config::Key::getBool(), and getKey().

Referenced by getBool().

6.75.3.10 getBoolDefault()

```
bool config::Config::getBoolDefault (
             const String & path,
             bool defaultValue )  [inline]
```

Definition at line 110 of file config.hpp.

References getBool(), and hasKey().

6.75.3.11 getFloat()

```
double config::Config::getFloat (
             const String & path )  [inline]
```

Look up the key at the given path, and return the value of that key as a bool.

Parameters

path	- Path for key to look up

Exceptions

KeyNotFound *(p. 674)*	is thrown if the specified path doesn't exist.

Definition at line 139 of file config.hpp.

References getFloatArray().

6.75.3.12 getFloatArray()

```
double Config::getFloatArray (
            const String & path,
             UInt64 index )
```

Definition at line 238 of file config.cpp.

References config::Key::getFloat(), and getKey().

Referenced by getFloat().

6.75.3.13 getInt()

```
SInt64 config::Config::getInt (
            const String & path )  [inline]
```

Look up the key at the given path, and return the value of that key as a bool.

Parameters

path	- Path for key to look up

Exceptions

KeyNotFound *(p. 674)*	is thrown if the specified path doesn't exist.

Definition at line 122 of file config.hpp.

References getIntArray().

6.75.3.14 getIntArray()

```
SInt64 Config::getIntArray (
            const String & path,
             UInt64 index )
```

Definition at line 228 of file config.cpp.

References config::Key::getInt(), and getKey().

Referenced by BranchPredictor::create(), and getInt().

6.75.3.15 getKey() [1/2]

```
template<class V >
const  Key& config::Config::getKey (
             const String & path,
             const V & default_val,
              UInt64 index )  [private]
```

6.75.3.16 getKey() [2/2]

```
const  Key & Config::getKey (
             const String & path,
              UInt64 index )  [private]
```

Definition at line 102 of file config.cpp.

References config::Error(), config::Section::getKey(), getSection_unsafe(), config::Section::hasKey(), isLeaf(), m↩
_root, and splitPath().

Referenced by getBoolArray(), getFloatArray(), getIntArray(), and getStringArray().

6.75.3.17 getKey_unsafe()

```
 Key& config::Config::getKey_unsafe (
             String const & path )  [protected]
```

6.75.3.18 getRoot()

```
const  Section& config::Config::getRoot ( )  [inline]
```

Returns a reference to the root section of the configuration tree.

Definition at line 93 of file config.hpp.

References m_root.

6.75.3.19 getRoot_unsafe()

```
 Section& config::Config::getRoot_unsafe ( )  [inline], [protected]
```

Definition at line 170 of file config.hpp.

References m_root.

6.75.3.20 getSection()

```
const  Section & Config::getSection (
             const String & path )
```

Returns a reference to the section at the given path.

Definition at line 33 of file config.cpp.

References getSection_unsafe().

6.75.3.21 getSection_unsafe()

```
 Section & Config::getSection_unsafe (
             String const & path )  [protected]
```

Definition at line 38 of file config.cpp.

References config::Section::addSubsection(), config::Section::getSection_unsafe(), config::Section::hasSection(), isLeaf(), m_root, and splitPathElements().

Referenced by addKeyInternal(), addSection(), config::ConfigFile::evalTree(), getKey(), getSection(), and hasKey().

6.75.3.22 getString()

```
const String config::Config::getString (
             const String & path )  [inline]
```

Look up the key at the given path, and return the value of that key as a bool.

Parameters

path	- Path for key to look up

Exceptions

KeyNotFound *(p. 674)*	is thrown if the specified path doesn't exist.

Definition at line 129 of file config.hpp.

References getStringArray().

Referenced by get(), MemoryManagerBase::getCoreListWithMemoryControllers(), Config::getNearest↩ AcceptableCoreCount(), Config::getNetworkModels(), and main().

6.75.3.23 getStringArray()

```
const String Config::getStringArray (
             const String & path,
              UInt64 index )
```

Definition at line 233 of file config.cpp.

References getKey(), and config::Key::getString().

Referenced by BranchPredictor::create(), and getString().

6.75.3.24 hasKey()

```
bool Config::hasKey (
             const String & path,
              UInt64 index = UINT64_MAX )
```

Definition at line 81 of file config.cpp.

References getSection_unsafe(), config::Section::hasKey(), isLeaf(), m_root, and splitPath().

Referenced by getBoolDefault().

6.75.3.25 isLeaf()

```
bool Config::isLeaf (
             const String & path )  [static], [private]
```

Definition at line 27 of file config.cpp.

Referenced by addKeyInternal(), getKey(), getSection_unsafe(), and hasKey().

6.75.3.26 load()

```
void Config::load (
             const String & path )
```

A function to convert from external representation to in-memory representation This function sets the member variable m_path and then calls the **loadConfig()** (p. 338) function which must be implemented by the derived back-end implementation.

Parameters

path	- This is the path where we load the configuration from. This may either be a registry path for the case of the ConfigRegistry interface or a file path in the case of a **ConfigFile** (p. 362) interface.

Exceptions

FileNotFound *(p. 566)*	- This exception is thrown if the **ConfigFile** (p. 362) interface is instructed to load a file which does not exist.
parseError	- This exception is thrown on a malformed file with the **ConfigFile** (p. 362) interface.

Definition at line 70 of file config.cpp.

References loadConfig(), and m_path.

Referenced by handle_generic_arg(), and main().

6.75.3.27 loadConfig()

```
virtual void config::Config::loadConfig ( )  [protected], [pure virtual]
```

Implemented in **config::ConfigFile** (p. 366).

Referenced by load().

6.75.3.28 Save()

```
void config::Config::Save ( )  [inline]
```

A function which will save the in-memory representation to the external representation that was specified with the **load()** (p. 337) function.

Definition at line 69 of file config.hpp.

References m_path, and saveAs().

6.75.3.29 saveAs()

```
virtual void config::Config::saveAs (
            const String & path )  [inline], [virtual]
```

A function for saving the entire configuration tree to the specified path. This function is responsible for walking through the entire configuration tree (starting at the root) and outputing an appropriate text representation than can then be re-read by an appropriate configuration class. Note: In the case of a write-through situation (as is the case with the windows registry), this function is unnecessary.

Reimplemented in **config::ConfigFile** (p. 367).

Definition at line 53 of file config.hpp.

Referenced by Save(), and Simulator::start().

6.75.3.30 set() [1/3]

```
void Config::set (
            const String & path,
            const String & new_value )  [virtual]
```

A function that will save a given value to key at the specified path.

Definition at line 207 of file config.cpp.

References addKey().

6.75.3.31 set() [2/3]

```
void Config::set (
            const String & path,
            double new_value )  [virtual]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 217 of file config.cpp.

References addKey().

6.75.3.32 set() [3/3]

```
void Config::set (
            const String & path,
             SInt64 new_value )  [virtual]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 212 of file config.cpp.

References addKey().

6.75.3.33 showFullTree()

```
String config::Config::showFullTree ( )  [inline]
```

Returns a string representation of the loaded configuration tree.

Definition at line 148 of file config.hpp.

References m_root, and showTree().

6.75.3.34 showTree()

```
String Config::showTree (
            const  Section & current,
            int depth = 0 )
```

Returns a string representation of the tree starting at the specified section.

Parameters

current	The root of the tree for which we are creating a string representation.

Definition at line 179 of file config.cpp.

References config::Section::getKeys(), config::Section::getName(), and config::Section::getSubsections().

Referenced by showFullTree().

6.75.3.35 splitPath()

```
std::pair< String, String > Config::splitPath (
            const String & path )  [static], [private]
```

Definition at line 143 of file config.cpp.

References splitPathElements().

Referenced by addKeyInternal(), addSection(), getKey(), and hasKey().

6.75.3.36 splitPathElements()

```
std::pair< String, String > Config::splitPathElements (
            const String & path,
             PathElementList & path_elements )  [static], [private]
```

Definition at line 150 of file config.cpp.

Referenced by getSection_unsafe(), and splitPath().

6.75.4 Member Data Documentation

6.75.4.1 m_case_sensitive

```
bool config::Config::m_case_sensitive  [protected]
```

Definition at line 164 of file config.hpp.

6.75.4.2 m_path

```
String config::Config::m_path  [protected]
```

Definition at line 166 of file config.hpp.

Referenced by load(), config::ConfigFile::loadConfig(), and Save().

6.75.4.3 m_root

```
Section config::Config::m_root  [protected]
```

Definition at line 165 of file config.hpp.

Referenced by addKeyInternal(), clear(), getKey(), getRoot(), getRoot_unsafe(), getSection_unsafe(), hasKey(), config::ConfigFile::loadConfigFromString(), config::ConfigFile::parse(), config::ConfigFile::saveAs(), and showFullTree().

The documentation for this class was generated from the following files:

- common/config/ **config.hpp**
- common/config/ **config.cpp**

6.76 Config Class Reference

```
#include <config.h>
```

Public Types

- enum **SimulationMode** { **PINTOOL** = 0, **STANDALONE**, **NUM_SIMULATION_MODES** }
- enum **SimulationROI** { **ROI_FULL**, **ROI_MAGIC**, **ROI_SCRIPT** }
- typedef std::unordered_map< **UInt32**, **core_id_t** > **CommToCoreMap**

Public Member Functions

- **Config** (**SimulationMode** mode)
- ∼**Config** ()
- void **loadFromFile** (char ∗filename)
- void **loadFromCmdLine** ()
- **UInt32** **getTotalCores** ()
- **UInt32** **getApplicationCores** ()
- void **updateCommToCoreMap** (**UInt32** comm_id, **core_id_t** core_id)
- **UInt32** **getCoreFromCommId** (**UInt32** comm_id)
- void **getNetworkModels** (**UInt32** ∗) const
- **UInt32** **getCoreIDLength** ()
- **SimulationMode** **getSimulationMode** ()
- **UInt32** **getNumHostCores** () const
- bool **getEnableSMCSupport** () const
- void **forceEnableSMCSupport** ()
- bool **getIssueMemopsAtFunctional** () const
- bool **getEnableICacheModeling** () const
- **SimulationROI** **getSimulationROI** () const
- bool **getEnableProgressTrace** () const
- bool **getEnableSync** () const
- bool **getEnableSyncReport** () const
- bool **getOSEmuPthreadReplace** () const
- **UInt32** **getOSEmuNprocs** () const
- bool **getOSEmuClockReplace** () const
- time_t **getOSEmuTimeStart** () const
- **ClockSkewMinimizationObject::Scheme** **getClockSkewMinimizationScheme** () const
- **UInt64** **getHPIInstructionsPerCore** () const
- **UInt64** **getHPIInstructionsGlobal** () const
- bool **getEnableSpinLoopDetection** () const
- bool **suppressStdout** () const
- bool **suppressStderr** () const
- bool **getEnablePinPlay** () const
- bool **getEnableSyscallEmulation** () const
- bool **getBBVsEnabled** () const
- void **setBBVsEnabled** (bool enable)
- const **CacheEfficiencyTracker::Callbacks** & **getCacheEfficiencyCallbacks** () const
- bool **hasCacheEfficiencyCallbacks** () const
- void **setCacheEfficiencyCallbacks** (**CacheEfficiencyTracker::CallbackGetOwner** get_owner_func, **CacheEfficiencyTracker::CallbackNotifyAccess** notify_access_func, **CacheEfficiencyTracker::↩ CallbackNotifyEvict** notify_evict_func, **UInt64** user_arg)
- String **getOutputDirectory** () const
- String **formatOutputFileName** (String filename) const
- void **logCoreMap** ()
- bool **getCircularLogEnabled** () const

Static Public Member Functions

- static **Config** ∗ **getSingleton** ()

Static Private Member Functions

- static **SimulationMode** **parseSimulationMode** (String mode)
- static **UInt32** **computeCoreIDLength** (**UInt32** core_count)
- static **UInt32** **getNearestAcceptableCoreCount** (**UInt32** core_count)

Private Attributes

- **UInt32 m_total_cores**
- **UInt32 m_core_id_length**
- **CommToCoreMap m_comm_to_core_map**
- **SimulationMode m_simulation_mode**

Static Private Attributes

- static **Config** * **m_singleton**
- static String **m_knob_output_directory**
- static **UInt32 m_knob_total_cores**
- static **UInt32 m_knob_num_host_cores**
- static bool **m_knob_enable_smc_support**
- static bool **m_knob_issue_memops_at_functional**
- static bool **m_knob_enable_icache_modeling**
- static **SimulationROI m_knob_roi**
- static bool **m_knob_enable_progress_trace**
- static bool **m_knob_enable_sync**
- static bool **m_knob_enable_sync_report**
- static bool **m_knob_osemu_pthread_replace**
- static **UInt32 m_knob_osemu_nprocs**
- static bool **m_knob_osemu_clock_replace**
- static time_t **m_knob_osemu_time_start**
- static bool **m_knob_bbvs**
- static **ClockSkewMinimizationObject::Scheme m_knob_clock_skew_minimization_scheme**
- static **UInt64 m_knob_hpi_percore**
- static **UInt64 m_knob_hpi_global**
- static bool **m_knob_enable_spinloopdetection**
- static bool **m_suppress_stdout**
- static bool **m_suppress_stderr**
- static bool **m_circular_log_enabled**
- static bool **m_knob_enable_pinplay**
- static bool **m_knob_enable_syscall_emulation**
- static **CacheEfficiencyTracker::Callbacks m_cache_efficiency_callbacks**

6.76.1 Detailed Description

Definition at line 27 of file config.h.

6.76.2 Member Typedef Documentation

6.76.2.1 CommToCoreMap

```
typedef std::unordered_map< UInt32, core_id_t >  Config::CommToCoreMap
```

Definition at line 44 of file config.h.

6.76.3 Member Enumeration Documentation

6.76.3.1 SimulationMode

enum **Config::SimulationMode**

Enumerator

PINTOOL	
STANDALONE	
NUM_SIMULATION_MODES	

Definition at line 30 of file config.h.

6.76.3.2 SimulationROI

enum **Config::SimulationROI**

Enumerator

ROI_FULL	
ROI_MAGIC	
ROI_SCRIPT	

Definition at line 37 of file config.h.

6.76.4 Constructor & Destructor Documentation

6.76.4.1 Config()

```
Config::Config (
            SimulationMode mode )
```

Definition at line 52 of file config.cc.

References computeCoreIDLength(), m_circular_log_enabled, m_core_id_length, m_knob_bbvs, m_knob_clock↩ _skew_minimization_scheme, m_knob_enable_icache_modeling, m_knob_enable_pinplay, m_knob_enable_↩ progress_trace, m_knob_enable_smc_support, m_knob_enable_spinloopdetection, m_knob_enable_sync, m_↩ knob_enable_sync_report, m_knob_enable_syscall_emulation, m_knob_hpi_global, m_knob_hpi_percore, m_↩ knob_issue_memops_at_functional, m_knob_num_host_cores, m_knob_osemu_clock_replace, m_knob_osemu↩ _nprocs, m_knob_osemu_pthread_replace, m_knob_osemu_time_start, m_knob_output_directory, m_knob_roi, m_knob_total_cores, m_simulation_mode, m_singleton, m_suppress_stderr, m_suppress_stdout, m_total_cores, ClockSkewMinimizationObject::parseScheme(), ROI_FULL, ROI_MAGIC, and ROI_SCRIPT.

6.76.4.2 ~Config()

```
Config::~Config ( )
```

Definition at line 129 of file config.cc.

6.76.5 Member Function Documentation

6.76.5.1 computeCoreIDLength()

```
UInt32 Config::computeCoreIDLength (
            UInt32 core_count )  [static], [private]
```

Definition at line 143 of file config.cc.

References ceilLog2().

Referenced by Config().

6.76.5.2 forceEnableSMCSupport()

```
void Config::forceEnableSMCSupport ( )  [inline]
```

Definition at line 73 of file config.h.

References m_knob_enable_smc_support.

6.76.5.3 formatOutputFileName()

```
String Config::formatOutputFileName (
            String filename ) const
```

Definition at line 173 of file config.cc.

References m_knob_output_directory.

Referenced by Simulator::start().

6.76.5.4 getApplicationCores()

```
UInt32 Config::getApplicationCores ( )
```

Definition at line 138 of file config.cc.

References getTotalCores().

Referenced by NetworkModelBus::accountPacket(), NetworkModelEMeshHopByHop::computeMeshDimensions(), DvfsManager::DvfsManager(), and MemoryManagerBase::getCoreListWithMemoryControllers().

6.76.5.5 getBBVsEnabled()

```
bool Config::getBBVsEnabled ( ) const  [inline]
```

Definition at line 93 of file config.h.

References m_knob_bbvs.

6.76.5.6 getCacheEfficiencyCallbacks()

```
const CacheEfficiencyTracker::Callbacks& Config::getCacheEfficiencyCallbacks ( ) const  [inline]
```

Definition at line 96 of file config.h.

References m_cache_efficiency_callbacks.

6.76.5.7 getCircularLogEnabled()

```
bool Config::getCircularLogEnabled ( ) const  [inline]
```

Definition at line 104 of file config.h.

References m_circular_log_enabled.

6.76.5.8 getClockSkewMinimizationScheme()

```
ClockSkewMinimizationObject::Scheme Config::getClockSkewMinimizationScheme ( ) const  [inline]
```

Definition at line 84 of file config.h.

References m_knob_clock_skew_minimization_scheme.

6.76.5.9 getCoreFromCommId()

```
UInt32 Config::getCoreFromCommId (
            UInt32 comm_id )
```

Definition at line 183 of file config.cc.

References INVALID_CORE_ID, and m_comm_to_core_map.

6.76.5.10 getCoreIDLength()

```
UInt32 Config::getCoreIDLength ( )  [inline]
```

Definition at line 64 of file config.h.

References m_core_id_length.

Referenced by Network::getModeledLength().

6.76.5.11 getEnableICacheModeling()

```
bool Config::getEnableICacheModeling ( ) const  [inline]
```

Definition at line 75 of file config.h.

References m_knob_enable_icache_modeling.

6.76.5.12 getEnablePinPlay()

```
bool Config::getEnablePinPlay ( ) const  [inline]
```

Definition at line 90 of file config.h.

References m_knob_enable_pinplay.

6.76.5.13 getEnableProgressTrace()

```
bool Config::getEnableProgressTrace ( ) const  [inline]
```

Definition at line 77 of file config.h.

References m_knob_enable_progress_trace.

6.76.5.14 getEnableSMCSupport()

```
bool Config::getEnableSMCSupport ( ) const  [inline]
```

Definition at line 72 of file config.h.

References m_knob_enable_smc_support.

6.76.5.15 getEnableSpinLoopDetection()

```
bool Config::getEnableSpinLoopDetection ( ) const  [inline]
```

Definition at line 87 of file config.h.

References m_knob_enable_spinloopdetection.

6.76.5.16 getEnableSync()

```
bool Config::getEnableSync ( ) const  [inline]
```

Definition at line 78 of file config.h.

References m_knob_enable_sync.

6.76.5.17 getEnableSyncReport()

```
bool Config::getEnableSyncReport ( ) const  [inline]
```

Definition at line 79 of file config.h.

References m_knob_enable_sync_report.

6.76.5.18 getEnableSyscallEmulation()

```
bool Config::getEnableSyscallEmulation ( ) const  [inline]
```

Definition at line 91 of file config.h.

References m_knob_enable_syscall_emulation.

6.76.5.19 getHPIInstructionsGlobal()

```
UInt64 Config::getHPIInstructionsGlobal ( ) const  [inline]
```

Definition at line 86 of file config.h.

References m_knob_hpi_global.

6.76.5.20 getHPIInstructionsPerCore()

```
UInt64 Config::getHPIInstructionsPerCore ( ) const  [inline]
```

Definition at line 85 of file config.h.

References m_knob_hpi_percore.

6.76.5.21 getIssueMemopsAtFunctional()

```
bool Config::getIssueMemopsAtFunctional ( ) const  [inline]
```

Definition at line 74 of file config.h.

References m_knob_issue_memops_at_functional.

6.76.5.22 getNearestAcceptableCoreCount()

```
UInt32 Config::getNearestAcceptableCoreCount (
            UInt32 core_count )  [static], [private]
```

Definition at line 203 of file config.cc.

References cfg, NetworkModel::computeCoreCountConstraints(), config::Error(), config::Config::getString(), NUM_STATIC_NETWORKS, NetworkModel::parseNetworkType(), STATIC_NETWORK_MEMORY_1, and STATIC_NETWORK_SYSTEM.

6.76.5.23 getNetworkModels()

```
void Config::getNetworkModels (
            UInt32 * models ) const
```

Definition at line 189 of file config.cc.

References cfg, config::Error(), config::Config::getString(), NetworkModel::parseNetworkType(), STATIC_NETWORK_MEMORY_1, and STATIC_NETWORK_SYSTEM.

Referenced by Network::Network().

6.76.5.24 getNumHostCores()

```
UInt32 Config::getNumHostCores ( ) const  [inline]
```

Definition at line 71 of file config.h.

References m_knob_num_host_cores.

6.76.5.25 getOSEmuClockReplace()

```
bool Config::getOSEmuClockReplace ( ) const  [inline]
```

Definition at line 82 of file config.h.

References m_knob_osemu_clock_replace.

6.76.5.26 getOSEmuNprocs()

```
UInt32 Config::getOSEmuNprocs ( ) const  [inline]
```

Definition at line 81 of file config.h.

References m_knob_osemu_nprocs.

6.76.5.27 getOSEmuPthreadReplace()

```
bool Config::getOSEmuPthreadReplace ( ) const  [inline]
```

Definition at line 80 of file config.h.

References m_knob_osemu_pthread_replace.

6.76.5.28 getOSEmuTimeStart()

```
time_t Config::getOSEmuTimeStart ( ) const  [inline]
```

Definition at line 83 of file config.h.

References m_knob_osemu_time_start.

6.76.5.29 getOutputDirectory()

```
String Config::getOutputDirectory ( ) const
```

Definition at line 168 of file config.cc.

References m_knob_output_directory.

6.76.5.30 getSimulationMode()

```
SimulationMode Config::getSimulationMode ( )  [inline]
```

Definition at line 67 of file config.h.

References m_simulation_mode.

6.76.5.31 getSimulationROI()

```
SimulationROI Config::getSimulationROI ( ) const  [inline]
```

Definition at line 76 of file config.h.

References m_knob_roi.

6.76.5.32 getSingleton()

```
Config * Config::getSingleton ( )  [static]
```

Definition at line 42 of file config.cc.

References m_singleton.

Referenced by NetworkModelBus::accountPacket(), SmTransport::clearNodeForId(), NetworkModelEMeshHop↩ ByHop::computeLatency(), NetworkModelEMeshHopByHop::computeMeshDimensions(), CoreManager::Core↩ Manager(), SmTransport::createNode(), DvfsManager::DvfsManager(), DramPerfModelConstant::getAccess↩ Latency(), DramPerfModelNormal::getAccessLatency(), DramPerfModelReadWrite::getAccessLatency(), Core↩ Manager::getCoreFromID(), MemoryManagerBase::getCoreListWithMemoryControllers(), Network::getModeled↩ Length(), NetworkModelEMeshHopByHop::getNextDest(), SmTransport::getNodeFromId(), CoreManager↩ ::initializeCommId(), Network::Network(), NetworkModelEMeshHopCounter::NetworkModelEMeshHopCounter(), NetworkModelEMeshHopCounter::processReceivedPacket(), NetworkModelMagic::processReceivedPacket(), NetworkModelEMeshHopByHop::processReceivedPacket(), SimThreadManager::quitSimThreads(), Core↩ Manager::registerSimThread(), NetworkModelEMeshHopCounter::routePacket(), NetworkModelMagic::route↩ Packet(), NetworkModelBus::routePacket(), NetworkModelEMeshHopByHop::routePacket(), SmTransport::Sm↩ Transport(), and SimThreadManager::spawnSimThreads().

6.76.5.33 getTotalCores()

```
UInt32 Config::getTotalCores ( )
```

Definition at line 133 of file config.cc.

References m_total_cores.

Referenced by CoreManager::CoreManager(), getApplicationCores(), Network::Network(), NetworkModelEMeshHopCounter::NetworkModelEMeshHopCounter(), SimThreadManager::quitSimThreads(), NetworkModelEMeshHopCounter::routePacket(), NetworkModelMagic::routePacket(), NetworkModelBus::routePacket(), NetworkModelEMeshHopByHop::routePacket(), SmTransport::SmTransport(), and SimThreadManager::spawnSimThreads().

6.76.5.34 hasCacheEfficiencyCallbacks()

```
bool Config::hasCacheEfficiencyCallbacks ( ) const  [inline]
```

Definition at line 97 of file config.h.

References m_cache_efficiency_callbacks, and CacheEfficiencyTracker::Callbacks::notify_evict_func.

6.76.5.35 loadFromCmdLine()

```
void Config::loadFromCmdLine ( )
```

Definition at line 163 of file config.cc.

6.76.5.36 loadFromFile()

```
void Config::loadFromFile (
            char * filename )
```

Definition at line 155 of file config.cc.

6.76.5.37 logCoreMap()

```
void Config::logCoreMap ( )
```

6.76.5.38 parseSimulationMode()

```
static SimulationMode Config::parseSimulationMode (
            String mode ) [static], [private]
```

6.76.5.39 setBBVsEnabled()

```
void Config::setBBVsEnabled (
            bool enable ) [inline]
```

Definition at line 94 of file config.h.

References m_knob_bbvs.

6.76.5.40 setCacheEfficiencyCallbacks()

```
void Config::setCacheEfficiencyCallbacks (
             CacheEfficiencyTracker::CallbackGetOwner get_owner_func,
             CacheEfficiencyTracker::CallbackNotifyAccess notify_access_func,
             CacheEfficiencyTracker::CallbackNotifyEvict notify_evict_func,
             UInt64 user_arg )
```

Definition at line 243 of file config.cc.

References config::Error(), CacheEfficiencyTracker::Callbacks::get_owner_func, m_cache_efficiency_callbacks, CacheEfficiencyTracker::Callbacks::notify_access_func, CacheEfficiencyTracker::Callbacks::notify_evict_func, and CacheEfficiencyTracker::Callbacks::user_arg.

6.76.5.41 suppressStderr()

```
bool Config::suppressStderr ( ) const [inline]
```

Definition at line 89 of file config.h.

References m_suppress_stderr.

6.76.5.42 suppressStdout()

```
bool Config::suppressStdout ( ) const [inline]
```

Definition at line 88 of file config.h.

References m_suppress_stdout.

6.76.5.43 updateCommToCoreMap()

```
void Config::updateCommToCoreMap (
              UInt32 comm_id,
              core_id_t core_id )
```

Definition at line 178 of file config.cc.

References m_comm_to_core_map.

Referenced by CoreManager::initializeCommId().

6.76.6 Member Data Documentation

6.76.6.1 m_cache_efficiency_callbacks

```
CacheEfficiencyTracker::Callbacks Config::m_cache_efficiency_callbacks  [static], [private]
```

Definition at line 144 of file config.h.

Referenced by getCacheEfficiencyCallbacks(), hasCacheEfficiencyCallbacks(), and setCacheEfficiencyCallbacks().

6.76.6.2 m_circular_log_enabled

```
bool Config::m_circular_log_enabled  [static], [private]
```

Definition at line 140 of file config.h.

Referenced by Config(), and getCircularLogEnabled().

6.76.6.3 m_comm_to_core_map

```
CommToCoreMap Config::m_comm_to_core_map  [private]
```

Definition at line 112 of file config.h.

Referenced by getCoreFromCommId(), and updateCommToCoreMap().

6.76.6.4 m_core_id_length

UInt32 Config::m_core_id_length [private]

Definition at line 110 of file config.h.

Referenced by Config(), and getCoreIDLength().

6.76.6.5 m_knob_bbvs

bool Config::m_knob_bbvs [static], [private]

Definition at line 133 of file config.h.

Referenced by Config(), getBBVsEnabled(), and setBBVsEnabled().

6.76.6.6 m_knob_clock_skew_minimization_scheme

ClockSkewMinimizationObject::Scheme Config::m_knob_clock_skew_minimization_scheme [static], [private]

Definition at line 134 of file config.h.

Referenced by Config(), and getClockSkewMinimizationScheme().

6.76.6.7 m_knob_enable_icache_modeling

bool Config::m_knob_enable_icache_modeling [static], [private]

Definition at line 124 of file config.h.

Referenced by Config(), and getEnableICacheModeling().

6.76.6.8 m_knob_enable_pinplay

bool Config::m_knob_enable_pinplay [static], [private]

Definition at line 141 of file config.h.

Referenced by Config(), and getEnablePinPlay().

6.76.6.9 m_knob_enable_progress_trace

```
bool Config::m_knob_enable_progress_trace  [static], [private]
```

Definition at line 126 of file config.h.

Referenced by Config(), and getEnableProgressTrace().

6.76.6.10 m_knob_enable_smc_support

```
bool Config::m_knob_enable_smc_support  [static], [private]
```

Definition at line 122 of file config.h.

Referenced by Config(), forceEnableSMCSupport(), and getEnableSMCSupport().

6.76.6.11 m_knob_enable_spinloopdetection

```
bool Config::m_knob_enable_spinloopdetection  [static], [private]
```

Definition at line 137 of file config.h.

Referenced by Config(), and getEnableSpinLoopDetection().

6.76.6.12 m_knob_enable_sync

```
bool Config::m_knob_enable_sync  [static], [private]
```

Definition at line 127 of file config.h.

Referenced by Config(), and getEnableSync().

6.76.6.13 m_knob_enable_sync_report

```
bool Config::m_knob_enable_sync_report  [static], [private]
```

Definition at line 128 of file config.h.

Referenced by Config(), and getEnableSyncReport().

6.76.6.14 m_knob_enable_syscall_emulation

```
bool Config::m_knob_enable_syscall_emulation  [static], [private]
```

Definition at line 142 of file config.h.

Referenced by Config(), and getEnableSyscallEmulation().

6.76.6.15 m_knob_hpi_global

```
UInt64 Config::m_knob_hpi_global  [static], [private]
```

Definition at line 136 of file config.h.

Referenced by Config(), and getHPIInstructionsGlobal().

6.76.6.16 m_knob_hpi_percore

```
UInt64 Config::m_knob_hpi_percore  [static], [private]
```

Definition at line 135 of file config.h.

Referenced by Config(), and getHPIInstructionsPerCore().

6.76.6.17 m_knob_issue_memops_at_functional

```
bool Config::m_knob_issue_memops_at_functional  [static], [private]
```

Definition at line 123 of file config.h.

Referenced by Config(), and getIssueMemopsAtFunctional().

6.76.6.18 m_knob_num_host_cores

```
UInt32 Config::m_knob_num_host_cores  [static], [private]
```

Definition at line 121 of file config.h.

Referenced by Config(), and getNumHostCores().

6.76.6.19 m_knob_osemu_clock_replace

```
bool Config::m_knob_osemu_clock_replace  [static], [private]
```

Definition at line 131 of file config.h.

Referenced by Config(), and getOSEmuClockReplace().

6.76.6.20 m_knob_osemu_nprocs

```
UInt32 Config::m_knob_osemu_nprocs  [static], [private]
```

Definition at line 130 of file config.h.

Referenced by Config(), and getOSEmuNprocs().

6.76.6.21 m_knob_osemu_pthread_replace

```
bool Config::m_knob_osemu_pthread_replace  [static], [private]
```

Definition at line 129 of file config.h.

Referenced by Config(), and getOSEmuPthreadReplace().

6.76.6.22 m_knob_osemu_time_start

```
time_t Config::m_knob_osemu_time_start  [static], [private]
```

Definition at line 132 of file config.h.

Referenced by Config(), and getOSEmuTimeStart().

6.76.6.23 m_knob_output_directory

```
String Config::m_knob_output_directory  [static], [private]
```

Definition at line 119 of file config.h.

Referenced by Config(), formatOutputFileName(), and getOutputDirectory().

6.76.6.24 m_knob_roi

Config::SimulationROI Config::m_knob_roi [static], [private]

Definition at line 125 of file config.h.

Referenced by Config(), and getSimulationROI().

6.76.6.25 m_knob_total_cores

UInt32 Config::m_knob_total_cores [static], [private]

Definition at line 120 of file config.h.

Referenced by Config().

6.76.6.26 m_simulation_mode

SimulationMode Config::m_simulation_mode [private]

Definition at line 115 of file config.h.

Referenced by Config(), and getSimulationMode().

6.76.6.27 m_singleton

Config * Config::m_singleton [static], [private]

Definition at line 117 of file config.h.

Referenced by Config(), and getSingleton().

6.76.6.28 m_suppress_stderr

bool Config::m_suppress_stderr [static], [private]

Definition at line 139 of file config.h.

Referenced by Config(), and suppressStderr().

6.76.6.29 m_suppress_stdout

```
bool Config::m_suppress_stdout  [static], [private]
```

Definition at line 138 of file config.h.

Referenced by Config(), and suppressStdout().

6.76.6.30 m_total_cores

```
UInt32 Config::m_total_cores  [private]
```

Definition at line 109 of file config.h.

Referenced by Config(), and getTotalCores().

The documentation for this class was generated from the following files:

- common/misc/ **config.h**
- common/misc/ **config.cc**

6.77 config::config_parser Struct Reference

```
#include <config_file_grammar.hpp>
```

Inheritance diagram for config::config_parser:

Classes

- struct **definition**
- struct **Name**
- struct **NodeValue**

Public Types

- typedef node_iter_data_factory< **NodeValue** > **factory_t**

Public Member Functions

- void **show_match** (const char *begin, const char *end)

Public Attributes

- const typedef char * **iterator_t**

6.77.1 Detailed Description

Definition at line 46 of file config_file_grammar.hpp.

6.77.2 Member Typedef Documentation

6.77.2.1 factory_t

```
typedef node_iter_data_factory< NodeValue>  config::config_parser::factory_t
```

Definition at line 75 of file config_file_grammar.hpp.

6.77.3 Member Function Documentation

6.77.3.1 show_match()

```
void config::config_parser::show_match (
            const char * begin,
            const char * end )  [inline]
```

Definition at line 70 of file config_file_grammar.hpp.

6.77.4 Member Data Documentation

6.77.4.1 iterator_t

```
const typedef char* config::config_parser::iterator_t
```

Definition at line 76 of file config_file_grammar.hpp.

The documentation for this struct was generated from the following file:

- common/config/ **config_file_grammar.hpp**

6.78 config::ConfigFile Class Reference

ConfigFile (p. 362): A flat-file interface for the **Config** (p. 328) Class. This file contains the class that is used to interface a flat file for config input / output. It uses boost::spirit for the config grammar.

`#include <config_file.hpp>`

Inheritance diagram for config::ConfigFile:

Public Member Functions

- **ConfigFile** (bool case_sensitive=false)
- **ConfigFile** (const **Section** &root, bool case_sensitive=false)
- ~**ConfigFile** ()
- void **saveAs** (const String &path)

  A function for saving the entire configuration tree to the specified path. This function is responsible for walking through the entire configuration tree (starting at the root) and outputing an appropriate text representation than can then be re-read by an appropriate configuration class. Note: In the case of a write-through situation (as is the case with the windows registry), this function is unnecessary.
- void **loadConfigFromString** (const String & **cfg**)

Private Types

- typedef tree_parse_info< **config_parser::iterator_t**, **config_parser::factory_t** > **parse_info_t**
- typedef tree_match< **config_parser::iterator_t**, **config_parser::factory_t** > **tree_match_t**
- typedef tree_match_t::node_t **node_t**
- typedef tree_match_t::const_tree_iterator **tree_iter_t**

Private Member Functions

- void **SaveTreeAs** (std::ofstream &out, const **Section** &current)
- void **loadConfig** ()
- void **loadFileToString** (String &s, const String &filename)

  loadFileToString() *(p. 366) Function that reads a given filename into a String*
- void **parse** (const String &source, **Section** &current)

  parse() *(p. 367) The function that is responsible for recursively building the tree*
- void **evalTree** (**Section** &current, **tree_iter_t** const &node, int depth=0)
- void **showParseTree** (**tree_iter_t** const &node, int depth=0)
- void **unEscapeText** (const String &source, String &dest)
- void **escapeText** (const String &source, String &dest)
- String **getNodeValue** (**tree_iter_t** const &node)
- **RuleID** **getNodeID** (**tree_iter_t** const &node)
- void **createStringKey** (**Section** &current, const String key_name, const String &value)
- void **createIntKey** (**Section** &current, const String key_name, **SInt64** value)
- void **createFloatKey** (**Section** &current, const String key_name, double value)

Additional Inherited Members

6.78.1 Detailed Description

ConfigFile (p. 362): A flat-file interface for the **Config** (p. 328) Class. This file contains the class that is used to interface a flat file for config input / output. It uses boost::spirit for the config grammar.

The grammar is as follows:

```
config       =  key* >> section* >> end
section      =  section_name >> key*
key          =  key_name >> '=' >> value
key_name     =  string!
value        =  real | int | string
section_name = '[' >> *lex_escape_ch_p >> ']'
string       = '"' >> *lex_escape_ch_p >> '"' | +(alnum | punct) >> *(alnum | punct )
```

Definition at line 53 of file config_file.hpp.

6.78.2 Member Typedef Documentation

6.78.2.1 node_t

typedef tree_match_t::node_t **config::ConfigFile::node_t** [private]

Definition at line 59 of file config_file.hpp.

6.78.2.2 parse_info_t

typedef tree_parse_info< **config_parser::iterator_t**, **config_parser::factory_t**> **config::ConfigFile::parse_info_t** [private]

Definition at line 57 of file config_file.hpp.

6.78.2.3 tree_iter_t

typedef tree_match_t::const_tree_iterator **config::ConfigFile::tree_iter_t** [private]

Definition at line 60 of file config_file.hpp.

6.78.2.4 tree_match_t

```
typedef tree_match< config_parser::iterator_t, config_parser::factory_t> config::Config↩
File::tree_match_t [private]
```

Definition at line 58 of file config_file.hpp.

6.78.3 Constructor & Destructor Documentation

6.78.3.1 ConfigFile() [1/2]

```
config::ConfigFile::ConfigFile (
            bool case_sensitive = false )
```

Definition at line 23 of file config_file.cpp.

6.78.3.2 ConfigFile() [2/2]

```
config::ConfigFile::ConfigFile (
            const  Section & root,
            bool case_sensitive = false )
```

Definition at line 29 of file config_file.cpp.

6.78.3.3 ~ConfigFile()

```
config::ConfigFile::~ConfigFile ( ) [inline]
```

Definition at line 66 of file config_file.hpp.

6.78.4 Member Function Documentation

6.78.4.1 createFloatKey()

```
void config::ConfigFile::createFloatKey (
             Section & current,
            const String key_name,
            double value ) [private]
```

6.78.4.2 createIntKey()

```
void config::ConfigFile::createIntKey (
             Section & current,
            const String key_name,
             SInt64 value )  [private]
```

6.78.4.3 createStringKey()

```
void config::ConfigFile::createStringKey (
             Section & current,
            const String key_name,
            const String & value )  [private]
```

6.78.4.4 escapeText()

```
void config::ConfigFile::escapeText (
            const String & source,
            String & dest )  [private]
```

Definition at line 193 of file config_file.cpp.

Referenced by SaveTreeAs().

6.78.4.5 evalTree()

```
void config::ConfigFile::evalTree (
             Section & current,
             tree_iter_t const & node,
            int depth = 0 )  [private]
```

Definition at line 208 of file config_file.cpp.

References config::Section::addKey(), getNodeID(), getNodeValue(), config::Config::getSection_unsafe(), config::Section::isRoot(), config::keyID, config::keyNameID, config::keySeparatorID, config::keyValueArrayID, config::keyValueID, config::keyValueSpanID, config::sectionNameID, and unEscapeText().

Referenced by parse().

6.78.4.6 getNodeID()

```
RuleID config::ConfigFile::getNodeID (
            tree_iter_t const & node )  [private]
```

Definition at line 187 of file config_file.cpp.

Referenced by evalTree().

6.78.4.7 getNodeValue()

```
String config::ConfigFile::getNodeValue (
            tree_iter_t const & node )  [private]
```

Definition at line 180 of file config_file.cpp.

Referenced by evalTree().

6.78.4.8 loadConfig()

```
void config::ConfigFile::loadConfig ( )  [private], [virtual]
```

loadConfig() (p. 366) Function which loads the **Config** (p. 328) tree with the entries extracted from the file located at the m_path member variable.

Implements **config::Config** (p. 338).

Definition at line 52 of file config_file.cpp.

References loadConfigFromString(), loadFileToString(), and config::Config::m_path.

6.78.4.9 loadConfigFromString()

```
void config::ConfigFile::loadConfigFromString (
            const String & cfg )
```

Definition at line 69 of file config_file.cpp.

References cfg, config::Config::m_root, and parse().

Referenced by handle_generic_arg(), and loadConfig().

6.78.4.10 loadFileToString()

```
void config::ConfigFile::loadFileToString (
            String & s,
            const String & filename )  [private]
```

loadFileToString() (p. 366) Function that reads a given filename into a String

Definition at line 36 of file config_file.cpp.

Referenced by loadConfig().

6.78.4.11 parse()

```
void config::ConfigFile::parse (
            const String & source,
             Section & current )  [private]
```

parse() (p. 367) The function that is responsible for recursively building the tree

Definition at line 76 of file config_file.cpp.

References evalTree(), and config::Config::m_root.

Referenced by loadConfigFromString().

6.78.4.12 saveAs()

```
void config::ConfigFile::saveAs (
            const String & path )  [virtual]
```

A function for saving the entire configuration tree to the specified path. This function is responsible for walking through the entire configuration tree (starting at the root) and outputing an appropriate text representation than can then be re-read by an appropriate configuration class. Note: In the case of a write-through situation (as is the case with the windows registry), this function is unnecessary.

Reimplemented from **config::Config** (p. 338).

Definition at line 315 of file config_file.cpp.

References itostr(), config::Config::m_root, and SaveTreeAs().

6.78.4.13 SaveTreeAs()

```
void config::ConfigFile::SaveTreeAs (
            std::ofstream & out,
            const  Section & current )  [private]
```

Definition at line 345 of file config_file.cpp.

References escapeText(), config::Section::getArrayKeys(), config::Section::getFullPath(), config::Section::getKeys(), config::Section::getSubsections(), and config::Section::isRoot().

Referenced by saveAs().

6.78.4.14 showParseTree()

```
void config::ConfigFile::showParseTree (
             tree_iter_t const & node,
            int depth = 0 )  [private]
```

Definition at line 95 of file config_file.cpp.

References config::configID, config::keyID, config::keyNameID, config::keyValueArrayID, config::keyValueID, config::keyValueSpanID, config::sectionID, and config::sectionNameID.

6.78.4.15 unEscapeText()

```
void config::ConfigFile::unEscapeText (
            const String & source,
            String & dest )  [private]
```

Definition at line 139 of file config_file.cpp.

Referenced by evalTree().

The documentation for this class was generated from the following files:

- common/config/ **config_file.hpp**
- common/config/ **config_file.cpp**

6.79 ConstantTimeDistribution Class Reference

```
#include <distribution.h>
```

Inheritance diagram for ConstantTimeDistribution:

Public Member Functions

- **ConstantTimeDistribution** (**SubsecondTime** constant)
- **SubsecondTime next** ()

Private Attributes

- **SubsecondTime value**

6.79.1 Detailed Description

Definition at line 42 of file distribution.h.

6.79.2 Constructor & Destructor Documentation

6.79.2.1 ConstantTimeDistribution()

```
ConstantTimeDistribution::ConstantTimeDistribution (
            SubsecondTime constant )  [inline]
```

Definition at line 45 of file distribution.h.

6.79.3 Member Function Documentation

6.79.3.1 next()

```
SubsecondTime ConstantTimeDistribution::next ( )  [inline], [virtual]
```

Implements **TimeDistribution** (p. 1408).

Definition at line 47 of file distribution.h.

References value.

6.79.4 Member Data Documentation

6.79.4.1 value

```
SubsecondTime ConstantTimeDistribution::value  [private]
```

Definition at line 52 of file distribution.h.

Referenced by next().

The documentation for this class was generated from the following file:

- common/misc/ **distribution.h**

6.80 ContentionModel Class Reference

`#include <contention_model.h>`

Public Member Functions

- **ContentionModel** ()
- **ContentionModel** (String name, **core_id_t** core_id, **UInt32** num_outstanding=1)
- ~**ContentionModel** ()
- uint64_t **getBarrierCompletionTime** (uint64_t **t_start**, uint64_t t_delay, **UInt64** tag=0)
- **SubsecondTime getBarrierCompletionTime** (**SubsecondTime t_start**, **SubsecondTime** t_delay, **UInt64** tag=0)
- uint64_t **getCompletionTime** (uint64_t **t_start**, uint64_t t_delay, **UInt64** tag=0)
- **SubsecondTime getCompletionTime** (**SubsecondTime t_start**, **SubsecondTime** t_delay, **UInt64** tag=0)
- uint64_t **getStartTime** (uint64_t **t_start**)
- **SubsecondTime getStartTime** (**SubsecondTime t_start**)
- **UInt32 getNumUsed** (uint64_t **t_start**)
- **UInt32 getNumUsed** (**SubsecondTime t_start**)
- **SubsecondTime getTagCompletionTime** (**UInt64** tag)
- bool **hasFreeSlot** (**SubsecondTime t_start**, **UInt64** tag=-1)
- bool **hasFreeSlot** (uint64_t **t_start**, **UInt64** tag=-1)
- bool **hasTag** (**UInt64** tag)

Public Attributes

- **UInt64 m_n_requests**
- **UInt64 m_n_barriers**
- **UInt64 m_n_outoforder**
- **UInt64 m_n_simultaneous**
- **UInt64 m_n_hasfreefail**
- **SubsecondTime m_total_delay**
- **SubsecondTime m_total_barrier_delay**

Private Attributes

- **UInt32 m_num_outstanding**
- std::vector< std::pair< **SubsecondTime**, **UInt64** > > **m_time**
- **SubsecondTime m_t_last**
- const **ComponentPeriod** * **m_proc_period**

6.80.1 Detailed Description

Definition at line 8 of file contention_model.h.

6.80.2 Constructor & Destructor Documentation

6.80.2.1 ContentionModel() [1/2]

```
ContentionModel::ContentionModel ( )
```

Definition at line 6 of file contention_model.cc.

6.80.2.2 ContentionModel() [2/2]

```
ContentionModel::ContentionModel (
             String name,
              core_id_t core_id,
              UInt32 num_outstanding = 1 )
```

Definition at line 20 of file contention_model.cc.

References m_n_barriers, m_n_hasfreefail, m_n_outoforder, m_n_requests, m_n_simultaneous, m_num_↩ outstanding, m_total_barrier_delay, m_total_delay, and registerStatsMetric().

6.80.2.3 ~ContentionModel()

```
ContentionModel::~ContentionModel ( )
```

Definition at line 45 of file contention_model.cc.

6.80.3 Member Function Documentation

6.80.3.1 getBarrierCompletionTime() [1/2]

```
SubsecondTime ContentionModel::getBarrierCompletionTime (
            SubsecondTime t_start,
            SubsecondTime t_delay,
            UInt64 tag = 0 )
```

Definition at line 123 of file contention_model.cc.

References m_n_barriers, m_num_outstanding, m_time, m_total_barrier_delay, and t_start.

6.80.3.2 getBarrierCompletionTime() [2/2]

```
uint64_t ContentionModel::getBarrierCompletionTime (
            uint64_t t_start,
            uint64_t t_delay,
            UInt64 tag = 0 )
```

Definition at line 113 of file contention_model.cc.

References SubsecondTimeCycleConverter::cyclesToSubsecondTime(), m_proc_period, SubsecondTimeCycle↩ Converter::subsecondTimeToCycles(), and t_start.

Referenced by IntervalTimer::dispatchInstruction().

6.80.3.3 getCompletionTime() [1/2]

```
SubsecondTime ContentionModel::getCompletionTime (
            SubsecondTime t_start,
            SubsecondTime t_delay,
            UInt64 tag = 0 )
```

Definition at line 159 of file contention_model.cc.

References m_n_outoforder, m_n_requests, m_n_simultaneous, m_num_outstanding, m_t_last, m_time, m_total↩ _delay, t_start, and SubsecondTime::Zero().

6.80.3.4 getCompletionTime() [2/2]

```
uint64_t ContentionModel::getCompletionTime (
            uint64_t t_start,
            uint64_t t_delay,
            UInt64 tag = 0 )
```

Definition at line 148 of file contention_model.cc.

References SubsecondTimeCycleConverter::cyclesToSubsecondTime(), m_proc_period, SubsecondTimeCycle↩ Converter::subsecondTimeToCycles(), and t_start.

Referenced by DramCache::callPrefetcher(), QueueModelContention::computeQueueDelay(), ParametricDram↩ DirectoryMSI::CacheCntlr::copyDataFromNextLevel(), IntervalTimer::dispatchInstruction(), ParametricDram↩ DirectoryMSI::CacheCntlr::insertCacheBlock(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), and ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore().

6.80.3.5 getNumUsed() [1/2]

```
UInt32 ContentionModel::getNumUsed (
            SubsecondTime t_start )
```

Definition at line 56 of file contention_model.cc.

References m_num_outstanding, m_time, and t_start.

6.80.3.6 getNumUsed() [2/2]

```
UInt32 ContentionModel::getNumUsed (
            uint64_t t_start )
```

Definition at line 49 of file contention_model.cc.

References SubsecondTimeCycleConverter::cyclesToSubsecondTime(), m_proc_period, and t_start.

Referenced by RobTimer::printRob(), and RobSmtTimer::printRob().

6.80.3.7 getStartTime() [1/2]

```
SubsecondTime ContentionModel::getStartTime (
            SubsecondTime t_start )
```

Definition at line 247 of file contention_model.cc.

References m_num_outstanding, m_t_last, m_time, and t_start.

6.80.3.8 getStartTime() [2/2]

```
uint64_t ContentionModel::getStartTime (
            uint64_t t_start )
```

Definition at line 239 of file contention_model.cc.

References SubsecondTimeCycleConverter::cyclesToSubsecondTime(), m_proc_period, SubsecondTimeCycle↩Converter::subsecondTimeToCycles(), and t_start.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), and ParametricDramDirectoryMS↩I::CacheCntlr::processMemOpFromCore().

6.80.3.9 getTagCompletionTime()

```
SubsecondTime ContentionModel::getTagCompletionTime (
              UInt64 tag )
```

Definition at line 68 of file contention_model.cc.

References m_num_outstanding, m_time, and SubsecondTime::MaxTime().

Referenced by DramCache::doAccess(), and ParametricDramDirectoryMSI::CacheCntlr::processMemOpFrom↩ Core().

6.80.3.10 hasFreeSlot() [1/2]

```
bool ContentionModel::hasFreeSlot (
              SubsecondTime t_start,
              UInt64 tag = -1 )
```

Definition at line 86 of file contention_model.cc.

References m_n_hasfreefail, m_num_outstanding, m_time, and t_start.

Referenced by RobTimer::doIssue(), hasFreeSlot(), and RobSmtTimer::tryIssue().

6.80.3.11 hasFreeSlot() [2/2]

```
bool ContentionModel::hasFreeSlot (
             uint64_t t_start,
              UInt64 tag = -1 )
```

Definition at line 79 of file contention_model.cc.

References SubsecondTimeCycleConverter::cyclesToSubsecondTime(), hasFreeSlot(), m_proc_period, and t_↩ start.

6.80.3.12 hasTag()

```
bool ContentionModel::hasTag (
              UInt64 tag )
```

Definition at line 102 of file contention_model.cc.

References m_num_outstanding, and m_time.

6.80.4 Member Data Documentation

6.80.4.1 m_n_barriers

UInt64 ContentionModel::m_n_barriers

Definition at line 16 of file contention_model.h.

Referenced by ContentionModel(), and getBarrierCompletionTime().

6.80.4.2 m_n_hasfreefail

UInt64 ContentionModel::m_n_hasfreefail

Definition at line 19 of file contention_model.h.

Referenced by ContentionModel(), and hasFreeSlot().

6.80.4.3 m_n_outoforder

UInt64 ContentionModel::m_n_outoforder

Definition at line 17 of file contention_model.h.

Referenced by ContentionModel(), and getCompletionTime().

6.80.4.4 m_n_requests

UInt64 ContentionModel::m_n_requests

Definition at line 15 of file contention_model.h.

Referenced by ContentionModel(), and getCompletionTime().

6.80.4.5 m_n_simultaneous

UInt64 ContentionModel::m_n_simultaneous

Definition at line 18 of file contention_model.h.

Referenced by ContentionModel(), and getCompletionTime().

6.80.4.6 m_num_outstanding

UInt32 ContentionModel::m_num_outstanding [private]

Definition at line 10 of file contention_model.h.

Referenced by ContentionModel(), getBarrierCompletionTime(), getCompletionTime(), getNumUsed(), getStartTime(), getTagCompletionTime(), hasFreeSlot(), and hasTag().

6.80.4.7 m_proc_period

const **ComponentPeriod*** ContentionModel::m_proc_period [private]

Definition at line 13 of file contention_model.h.

Referenced by getBarrierCompletionTime(), getCompletionTime(), getNumUsed(), getStartTime(), and hasFreeSlot().

6.80.4.8 m_t_last

SubsecondTime ContentionModel::m_t_last [private]

Definition at line 12 of file contention_model.h.

Referenced by getCompletionTime(), and getStartTime().

6.80.4.9 m_time

std::vector<std::pair< **SubsecondTime**, **UInt64**> > ContentionModel::m_time [private]

Definition at line 11 of file contention_model.h.

Referenced by getBarrierCompletionTime(), getCompletionTime(), getNumUsed(), getStartTime(), getTagCompletionTime(), hasFreeSlot(), and hasTag().

6.80.4.10 m_total_barrier_delay

SubsecondTime ContentionModel::m_total_barrier_delay

Definition at line 21 of file contention_model.h.

Referenced by ContentionModel(), and getBarrierCompletionTime().

6.80.4.11 m_total_delay

```
SubsecondTime ContentionModel::m_total_delay
```

Definition at line 20 of file contention_model.h.

Referenced by ContentionModel(), and getCompletionTime().

The documentation for this class was generated from the following files:

- common/performance_model/ **contention_model.h**
- common/performance_model/ **contention_model.cc**

6.81 Core Class Reference

```
#include <core.h>
```

Public Types

- enum **State** {
  RUNNING = 0, **INITIALIZING**, **STALLED**, **SLEEPING**,
  WAKING_UP, **IDLE**, **BROKEN**, **NUM_STATES** }
- enum **lock_signal_t** {
  INVALID_LOCK_SIGNAL = 0, **MIN_LOCK_SIGNAL**, **NONE** = MIN_LOCK_SIGNAL, **LOCK**,
  UNLOCK, **MAX_LOCK_SIGNAL** = UNLOCK, **NUM_LOCK_SIGNAL_TYPES** = MAX_LOCK_SIGNAL - MIN_LOCK_SIGNAL + 1 }
- enum **mem_op_t** {
  INVALID_MEM_OP = 0, **MIN_MEM_OP**, **READ** = MIN_MEM_OP, **READ_EX**,
  WRITE, **MAX_MEM_OP** = WRITE, **NUM_MEM_OP_TYPES** = MAX_MEM_OP - MIN_MEM_OP + 1 }
- enum **MemModeled** {
  MEM_MODELED_NONE, **MEM_MODELED_COUNT**, **MEM_MODELED_COUNT_TLBTIME**, **MEM_M↩ ODELED_TIME**,
  MEM_MODELED_FENCED, **MEM_MODELED_RETURN** }

Public Member Functions

- **Core** (**SInt32** id)
- ∼**Core** ()
- bool **accessBranchPredictor** (**IntPtr** eip, bool taken, **IntPtr** target)
- **MemoryResult** **readInstructionMemory** (**IntPtr** address, **UInt32** instruction_size)
- **MemoryResult** **accessMemory** (**lock_signal_t** lock_signal, **mem_op_t** mem_op_type, **IntPtr** d_addr, char ∗data_buffer, **UInt32** data_size, **MemModeled** modeled= **MEM_MODELED_NONE**, **IntPtr** eip=0, **SubsecondTime** now= **SubsecondTime::MaxTime**(), bool is_fault_mask=false)
- **MemoryResult** **nativeMemOp** (**lock_signal_t** lock_signal, **mem_op_t** mem_op_type, **IntPtr** d_addr, char ∗data_buffer, **UInt32** data_size)
- void **accessMemoryFast** (bool icache, **mem_op_t** mem_op_type, **IntPtr** address)
- void **logMemoryHit** (bool icache, **mem_op_t** mem_op_type, **IntPtr** address, **MemModeled** modeled= **MEM_MODELED_NONE**, **IntPtr** eip=0)
- bool **countInstructions** (**IntPtr** address, **UInt32** count)
- void **emulateCpuid** (**UInt32** eax, **UInt32** ecx, **cpuid_result_t** &res) const
- int **getId** () const

- **Thread** * **getThread** () const
- void **setThread** (**Thread** *thread)
- **Network** * **getNetwork** ()
- **PerformanceModel** * **getPerformanceModel** ()
- **ClockSkewMinimizationClient** * **getClockSkewMinimizationClient** () const
- **MemoryManagerBase** * **getMemoryManager** ()
- const **MemoryManagerBase** * **getMemoryManager** () const
- **ShmemPerfModel** * **getShmemPerfModel** ()
- const **ComponentPeriod** * **getDvfsDomain** () const
- **TopologyInfo** * **getTopologyInfo** ()
- const **TopologyInfo** * **getTopologyInfo** () const
- const **CheetahManager** * **getCheetahManager** () const
- **State** **getState** () const
- void **setState** (**State** core_state)
- **UInt64** **getInstructionCount** ()
- **BbvCount** * **getBbvCount** ()
- **UInt64** **getInstructionsCallback** ()
- bool **isEnabledInstructionsCallback** ()
- void **setInstructionsCallback** (**UInt64** instructions)
- void **disableInstructionsCallback** ()
- void **enablePerformanceModels** ()
- void **disablePerformanceModels** ()
- void **updateSpinCount** (**UInt64** instructions, **SubsecondTime** elapsed_time)

Static Public Member Functions

- static const char * **CoreStateString** (**State** state)

Protected Attributes

- **UInt64** **m_instructions**
- **UInt64** **m_instructions_callback**
- **UInt64** **m_instructions_hpi_callback**
- **UInt64** **m_instructions_hpi_last**

Static Protected Attributes

- static **UInt64** **g_instructions_hpi_global** = 0
- static **UInt64** **g_instructions_hpi_global_callback** = 0

Private Member Functions

- **MemoryResult** **initiateMemoryAccess** (**MemComponent::component_t** mem_component, **lock_signal_t** lock_signal, **mem_op_t** mem_op_type, **IntPtr** address, **Byte** *data_buf, **UInt32** data_size, **MemModeled** modeled, **IntPtr** eip, **SubsecondTime** now)
- void **hookPeriodicInsCheck** ()
- void **hookPeriodicInsCall** ()

Private Attributes

- **core_id_t m_core_id**
- const **ComponentPeriod** * **m_dvfs_domain**
- **MemoryManagerBase** * **m_memory_manager**
- **Thread** * **m_thread**
- **Network** * **m_network**
- **PerformanceModel** * **m_performance_model**
- **ClockSkewMinimizationClient** * **m_clock_skew_minimization_client**
- **Lock m_mem_lock**
- **ShmemPerfModel** * **m_shmem_perf_model**
- **BbvCount m_bbv**
- **TopologyInfo** * **m_topology_info**
- **CheetahManager** * **m_cheetah_manager**
- **State m_core_state**
- **IntPtr m_icache_last_block**
- **UInt64 m_spin_loops**
- **UInt64 m_spin_instructions**
- **SubsecondTime m_spin_elapsed_time**

Static Private Attributes

- static **Lock m_global_core_lock**

Friends

- class **InstructionModeling**

6.81.1 Detailed Description

Definition at line 32 of file core.h.

6.81.2 Member Enumeration Documentation

6.81.2.1 lock_signal_t

enum **Core::lock_signal_t**

Enumerator

INVALID_LOCK_SIGNAL	
MIN_LOCK_SIGNAL	
NONE	
LOCK	
UNLOCK	
MAX_LOCK_SIGNAL	
NUM_LOCK_SIGNAL_TYPES	

Definition at line 48 of file core.h.

6.81.2.2 mem_op_t

enum **Core::mem_op_t**

Enumerator

INVALID_MEM_OP	
MIN_MEM_OP	
READ	
READ_EX	
WRITE	
MAX_MEM_OP	
NUM_MEM_OP_TYPES	

Definition at line 59 of file core.h.

6.81.2.3 MemModeled

enum **Core::MemModeled**

Enumerator

MEM_MODELED_NONE	
MEM_MODELED_COUNT	
MEM_MODELED_COUNT_TLBTIME	
MEM_MODELED_TIME	
MEM_MODELED_FENCED	
MEM_MODELED_RETURN	

Definition at line 71 of file core.h.

6.81.2.4 State

enum **Core::State**

Enumerator

RUNNING	
INITIALIZING	
STALLED	
SLEEPING	

Enumerator

WAKING_UP	
IDLE	
BROKEN	
NUM_STATES	

Definition at line 36 of file core.h.

6.81.3 Constructor & Destructor Documentation

6.81.3.1 Core()

```
Core::Core (
            SInt32 id )
```

Definition at line 67 of file core.cc.

References ClockSkewMinimizationClient::create(), PerformanceModel::create(), MemoryManagerBase::create↩MMU(), LOG_PRINT, m_clock_skew_minimization_client, m_instructions, m_memory_manager, m_network, m↩_performance_model, m_shmem_perf_model, m_spin_elapsed_time, m_spin_instructions, m_spin_loops, and registerStatsMetric().

6.81.3.2 ~Core()

```
Core::~Core ( )
```

Definition at line 107 of file core.cc.

References m_cheetah_manager, m_clock_skew_minimization_client, m_memory_manager, m_network, m_↩performance_model, m_shmem_perf_model, and m_topology_info.

6.81.4 Member Function Documentation

6.81.4.1 accessBranchPredictor()

```
bool Core::accessBranchPredictor (
            IntPtr eip,
           bool taken,
            IntPtr target )
```

Definition at line 194 of file core.cc.

References PerformanceModel::getBranchPredictor(), getPerformanceModel(), BranchPredictor::predict(), and BranchPredictor::update().

Referenced by DynamicInstruction::getBranchCost(), handleBranchWarming(), TraceThread::handleCacheOnly↩Func(), and TraceThread::handleInstructionWarmup().

6.81.4.2 accessMemory()

```
MemoryResult Core::accessMemory (
             lock_signal_t lock_signal,
             mem_op_t mem_op_type,
             IntPtr d_addr,
            char * data_buffer,
             UInt32 data_size,
             MemModeled modeled =  MEM_MODELED_NONE,
             IntPtr eip = 0,
             SubsecondTime now =  SubsecondTime::MaxTime(),
            bool is_fault_mask = false )
```

Definition at line 463 of file core.cc.

References initiateMemoryAccess(), MemComponent::L1_DCACHE, m_core_id, makeMemoryResult(), MEM_M↩ ODELED_NONE, nativeMemOp(), NONE, Config::PINTOOL, Config::STANDALONE, HitWhere::UNKNOWN, and SubsecondTime::Zero().

Referenced by DynamicInstruction::accessMemory(), PthreadEmu::BarrierInit(), PthreadEmu::BarrierWait(), PthreadEmu::CondBroadcast(), PthreadEmu::CondSignal(), PthreadEmu::CondWait(), SyscallServer::futexDo↩ Op(), TraceThread::handleCacheOnlyFunc(), SyscallServer::handleFutexCall(), TraceThread::handleInstruction↩ Warmup(), lite::handleMemoryReadDetailedIssue(), lite::handleMemoryReadFaultinjection(), lite::handleMemory↩ WriteDetailedIssue(), lite::handleMemoryWriteFaultinjection(), RobTimer::issueInstruction(), RobSmtTimer::issue↩ Instruction(), IntervalTimer::issueMemOp(), MagicServer::Magic_unlocked(), PthreadEmu::MutexLock(), Pthread↩ Emu::MutexTrylock(), PthreadEmu::MutexUnlock(), ThreadManager::onThreadExit(), readCstr(), readMemory(), and SyscallMdl::runEnter().

6.81.4.3 accessMemoryFast()

```
void Core::accessMemoryFast (
            bool icache,
             mem_op_t mem_op_type,
             IntPtr address )
```

Definition at line 263 of file core.cc.

References CheetahManager::access(), MemoryManagerBase::coreInitiateMemoryAccessFast(), getMemory↩ Manager(), PerformanceModel::handleMemoryLatency(), m_cheetah_manager, m_performance_model, Hit↩ Where::MISS, and SubsecondTime::Zero().

Referenced by InstructionModeling::accessInstructionCacheWarmup(), lite::handleMemoryRead(), and lite↩ ::handleMemoryWrite().

6.81.4.4 CoreStateString()

```
const char * Core::CoreStateString (
             Core::State state )  [static]
```

Definition at line 56 of file core.cc.

References core_state_names, LOG_ASSERT_ERROR, and NUM_STATES.

Referenced by RoutineTracerOndemand::RtnThread::printStack().

6.81.4.5 countInstructions()

```
bool Core::countInstructions (
              IntPtr address,
              UInt32 count )
```

Definition at line 137 of file core.cc.

References BbvCount::count(), PerformanceModel::countInstructions(), disableInstructionsCallback(), HookType::HOOK_INSTR_COUNT, hookPeriodicInsCheck(), isEnabledInstructionsCallback(), m_bbv, m_core_id, m_instructions, m_instructions_callback, m_performance_model, and BbvCount::sample().

Referenced by InstructionModeling::countInstructions(), TraceThread::handleCacheOnlyFunc(), TraceThread::handleInstructionCountFunc(), and TraceThread::run().

6.81.4.6 disableInstructionsCallback()

```
void Core::disableInstructionsCallback ( )  [inline]
```

Definition at line 124 of file core.h.

References m_instructions_callback.

Referenced by countInstructions(), FastForwardPerformanceManager::disable(), and SamplingManager::disableFastForward().

6.81.4.7 disablePerformanceModels()

```
void Core::disablePerformanceModels ( )
```

Definition at line 128 of file core.cc.

References ShmemPerfModel::disable(), PerformanceModel::disable(), MemoryManagerBase::disableModels(), Network::disableModels(), getMemoryManager(), getNetwork(), getPerformanceModel(), and getShmemPerfModel().

Referenced by FastForwardPerformanceManager::disable().

6.81.4.8 emulateCpuid()

```
void Core::emulateCpuid (
              UInt32 eax,
              UInt32 ecx,
              cpuid_result_t & res ) const
```

Definition at line 525 of file core.cc.

References TopologyInfo::apic_id, TopologyInfo::core_count, cpuid(), cpuid_result_t::eax, cpuid_result_t::ebx, cpuid_result_t::ecx, cpuid_result_t::edx, m_core_id, m_topology_info, TopologyInfo::PACKAGE_SHIFT_BITS, TopologyInfo::smt_count, and TopologyInfo::SMT_SHIFT_BITS.

Referenced by handleCpuid(), and TraceThread::handleEmuFunc().

6.81.4.9 enablePerformanceModels()

```
void Core::enablePerformanceModels ( )
```

Definition at line 120 of file core.cc.

References ShmemPerfModel::enable(), PerformanceModel::enable(), MemoryManagerBase::enableModels(), Network::enableModels(), getMemoryManager(), getNetwork(), getPerformanceModel(), and getShmemPerf↩ Model().

6.81.4.10 getBbvCount()

```
BbvCount* Core::getBbvCount ( )  [inline]
```

Definition at line 120 of file core.h.

References m_bbv.

6.81.4.11 getCheetahManager()

```
const  CheetahManager* Core::getCheetahManager ( ) const  [inline]
```

Definition at line 115 of file core.h.

References m_cheetah_manager.

6.81.4.12 getClockSkewMinimizationClient()

```
ClockSkewMinimizationClient* Core::getClockSkewMinimizationClient ( ) const  [inline]
```

Definition at line 108 of file core.h.

References m_clock_skew_minimization_client.

Referenced by SmtTimer::simulate(), and PerformanceModel::synchronize().

6.81.4.13 getDvfsDomain()

```
const  ComponentPeriod* Core::getDvfsDomain ( ) const  [inline]
```

Definition at line 112 of file core.h.

References m_dvfs_domain.

Referenced by DynamicInstruction::accessMemory(), PeriodicSampling::callbackDetailed(), DynamicInstruction↩ ::getBranchCost(), Instruction::getCost(), TraceThread::handleInstructionCountFunc(), IntervalTimer::issueMem↩ Op(), ParametricDramDirectoryMSI::MemoryManager::MemoryManager(), and LoopTracer::traceInstruction().

6.81.4.14 getId()

```
int Core::getId ( ) const  [inline]
```

Definition at line 103 of file core.h.

References m_core_id.

Referenced by TraceManager::accessMemory(), SmtTimer::barrier(), FastNehalem::Cache< assoc, size_kb >::Cache(), RoutineTracerFunctionStats::ThreadStatCpiMem::callback(), ThreadStatNamedStat::callback(), PerformanceModel::create(), FastNehalem::Dram::Dram(), PrL1PrL2DramDirectoryMSI::DramCntlr::DramCntlr(), lite::emuGetCPU(), FastforwardPerformanceModel::FastforwardPerformanceModel(), SamplingManager::getCoreHistoricCPI(), CoreManager::getCurrentCoreID(), RobSmtPerformanceModel::getRobTimer(), TraceThread::handleEmuFunc(), handleMagic(), handleMagicInstruction(), RobSmtTimer::initializeThread(), InstructionTracerFPStats::InstructionTracerFPStats(), IntervalContentionBoomV1::IntervalContentionBoomV1(), IntervalContentionNehalem::IntervalContentionNehalem(), IntervalTimer::IntervalTimer(), ParametricDramDirectoryMSI::MemoryManager::MemoryManager(), MicroOpPerformanceModel::MicroOpPerformanceModel(), ThreadManager::moveThread(), PthreadEmu::MutexLock(), PthreadEmu::MutexUnlock(), Network::netPullFromTransport(), Network::netRecv(), Network::netSend(), Network::Network(), NetworkModel::NetworkModel(), NetworkModelEMeshHopCounter::NetworkModelEMeshHopCounter(), FastforwardPerformanceModel::notifyElapsedTimeUpdate(), OneIPCPerformanceModel::OneIPCPerformanceModel(), ThreadManager::onThreadExit(), ThreadManager::onThreadStart(), PerformanceModel::PerformanceModel(), PrL1PrL2DramDirectoryMSI::DramCntlr::printDramAccessCount(), RoutineTracerOndemand::RtnThread::printStack(), PthreadEmu::pthreadCount(), CoreManager::registerSimThread(), SamplingManager::resetCoreHistoricCPIs(), RobSmtTimer::RobSmtTimer(), RobTimer::RobTimer(), SyscallMdl::runEnter(), SyscallMdl::runExit(), Thread::setCore(), BarrierSyncClient::synchronize(), InstructionTracerPrint::traceInstruction(), ThreadStatsManager::ThreadStats::update(), Thread::updateCoreTLS(), PthreadEmu::updateState(), TraceThread::va2pa(), and RobSmtPerformanceModel::~RobSmtPerformanceModel().

6.81.4.15 getInstructionCount()

```
UInt64 Core::getInstructionCount ( )  [inline]
```

Definition at line 119 of file core.h.

References m_instructions.

Referenced by SpinLoopDetector::commitBCT().

6.81.4.16 getInstructionsCallback()

```
UInt64 Core::getInstructionsCallback ( )  [inline]
```

Definition at line 121 of file core.h.

References m_instructions_callback.

6.81.4.17 getMemoryManager() [1/2]

```
MemoryManagerBase* Core::getMemoryManager ( )  [inline]
```

Definition at line 109 of file core.h.

References m_memory_manager.

Referenced by accessMemoryFast(), NetworkModelBus::accountPacket(), disablePerformanceModels(), enable↩ PerformanceModels(), Network::getModeledLength(), initiateMemoryAccess(), logMemoryHit(), NetworkModel↩ EMeshHopCounter::processReceivedPacket(), NetworkModelMagic::processReceivedPacket(), NetworkModel↩ EMeshHopByHop::processReceivedPacket(), readInstructionMemory(), and NetworkModelEMeshHopByHop↩ ::routePacket().

6.81.4.18 getMemoryManager() [2/2]

```
const MemoryManagerBase* Core::getMemoryManager ( ) const  [inline]
```

Definition at line 110 of file core.h.

References m_memory_manager.

6.81.4.19 getNetwork()

```
Network* Core::getNetwork ( )  [inline]
```

Definition at line 106 of file core.h.

References m_network.

Referenced by disablePerformanceModels(), enablePerformanceModels(), and Network::netSend().

6.81.4.20 getPerformanceModel()

```
PerformanceModel* Core::getPerformanceModel ( )  [inline]
```

Definition at line 107 of file core.h.

References m_performance_model.

Referenced by SyncClient::__mutexLock(), BarrierSyncServer::abortBarrier(), accessBranchPredictor(), ParametricDramDirectoryMSI::MemoryManager::accessTLB(), BarrierSyncServer::barrierRelease(), SyncClient::barrierWait(), PeriodicSampling::callbackDetailed(), SpinLoopDetector::commitBCT(), SyncClient::condBroadcast(), SyncClient::condSignal(), SyncClient::condWait(), MemoryManagerBase::coreInitiateMemoryAccessFast(), InstructionModeling::countInstructions(), FastForwardPerformanceManager::disable(), SamplingManager::disableFastForward(), disablePerformanceModels(), IntervalTimer::dispatchWindow(), RobTimer::doCommit(), RobSmtTimer::doCommit(), lite::emuClockGettime(), lite::emuGettimeofday(), enablePerformanceModels(), DynamicInstruction::getBranchCost(), SamplingManager::getCoreHistoricCPI(), TraceThread::getCurrentTime(), InstructionModeling::handleBasicBlock(), handleBranchWarming(), TraceThread::handleCacheOnlyFunc(), handleCheckScheduled(), TraceThread::handleEmuFunc(), SyscallMdl::handleFutexCall(), InstructionModeling::handleInstruction(), TraceThread::handleInstructionCountFunc(), TraceThread::handleInstructionWarmup(), handleRdtsc(), initiateMemoryAccess(), ThreadManager::joinThread(), ThreadStatsManager::metricCallback(), SyncClient::mutexUnlock(), Network::netRecv(), Network::netSend(), ThreadManager::onThreadExit(), ThreadManager::onThreadStart(), lite::pthreadAfter(), lite::pthreadBefore(), FastForwardPerformanceManager::recalibrateInstructionsCallback(), SamplingManager::recalibrateInstructionsCallback(), Thread::reschedule(), SamplingManager::resetCoreHistoricCPIs(), TraceThread::run(), SyscallMdl::runEnter(), SyscallMdl::runExit(), ThreadManager::spawnThread(), BarrierSyncClient::synchronize(), BarrierSyncServer::synchronize(), SchedulerPinnedBase::threadSetAffinity(), SchedulerPinnedBase::threadYield(), TraceThread::unblock(), ThreadStatsManager::ThreadStats::update(), and PthreadEmu::updateState().

6.81.4.21 getShmemPerfModel()

```
ShmemPerfModel* Core::getShmemPerfModel ( )  [inline]
```

Definition at line 111 of file core.h.

References m_shmem_perf_model.

Referenced by disablePerformanceModels(), enablePerformanceModels(), and initiateMemoryAccess().

6.81.4.22 getState()

```
State Core::getState ( ) const  [inline]
```

Definition at line 117 of file core.h.

References m_core_state.

Referenced by BarrierSyncServer::isCoreRunning(), and BarrierSyncServer::synchronize().

6.81.4.23 getThread()

```
Thread* Core::getThread ( ) const  [inline]
```

Definition at line 104 of file core.h.

References m_thread.

Referenced by FastForwardPerformanceManager::disable(), BarrierSyncServer::isCoreRunning(), Thread::setCore(), and BarrierSyncServer::synchronize().

6.81.4.24 getTopologyInfo() [1/2]

```
TopologyInfo* Core::getTopologyInfo ( )  [inline]
```

Definition at line 113 of file core.h.

References m_topology_info.

Referenced by ParametricDramDirectoryMSI::MemoryManager::MemoryManager().

6.81.4.25 getTopologyInfo() [2/2]

```
const  TopologyInfo* Core::getTopologyInfo ( ) const  [inline]
```

Definition at line 114 of file core.h.

References m_topology_info.

6.81.4.26 hookPeriodicInsCall()

```
void Core::hookPeriodicInsCall ( )  [private]
```

Definition at line 179 of file core.cc.

References g_instructions_hpi_global, g_instructions_hpi_global_callback, and HookType::HOOK_PERIODIC_INS.

Referenced by hookPeriodicInsCheck().

6.81.4.27 hookPeriodicInsCheck()

```
void Core::hookPeriodicInsCheck ( )  [private]
```

Definition at line 164 of file core.cc.

References g_instructions_hpi_global, g_instructions_hpi_global_callback, hookPeriodicInsCall(), m_instructions, m_instructions_hpi_callback, and m_instructions_hpi_last.

Referenced by countInstructions().

6.81.4.28 initiateMemoryAccess()

```
MemoryResult Core::initiateMemoryAccess (
            MemComponent::component_t mem_component,
            lock_signal_t lock_signal,
            mem_op_t mem_op_type,
            IntPtr address,
            Byte * data_buf,
            UInt32 data_size,
            MemModeled modeled,
            IntPtr eip,
            SubsecondTime now )  [private]
```

Definition at line 275 of file core.cc.

References ShmemPerfModel::_SIM_THREAD, ShmemPerfModel::_USER_THREAD, CheetahManager::access(), TLock< T_LockCreator >::acquire(), MemoryManagerBase::coreInitiateMemoryAccess(), MemoryManagerBase::getCacheBlockSize(), ShmemPerfModel::getElapsedTime(), PerformanceModel::getElapsedTime(), getMemoryManager(), getPerformanceModel(), getShmemPerfModel(), PerformanceModel::handleMemoryLatency(), ShmemPerfModel::incrTotalMemoryAccessLatency(), iolock, PerformanceModel::isEnabled(), itostr(), LOCK, LOG_ASSERT_ERROR, LOG_PRINT, m_cheetah_manager, m_core_id, m_mem_lock, m_performance_model, makeMemoryResult(), SubsecondTime::MaxTime(), MEM_MODELED_COUNT, MEM_MODELED_COUNT_TLBTIME, MEM_MODELED_FENCED, MEM_MODELED_NONE, MEM_MODELED_RETURN, MEM_MODELED_TIME, ModeledString(), MYLOG, Timer::now(), PerformanceModel::queuePseudoInstruction(), Operand::READ, READ, READ_EX, TLock< T_LockCreator >::release(), ShmemPerfModel::setElapsedTime(), HitWhere::UNKNOWN, UNLOCK, Operand::WRITE, WRITE, and SubsecondTime::Zero().

Referenced by accessMemory(), and readInstructionMemory().

6.81.4.29 isEnabledInstructionsCallback()

```
bool Core::isEnabledInstructionsCallback ( )  [inline]
```

Definition at line 122 of file core.h.

References m_instructions_callback.

Referenced by countInstructions().

6.81.4.30 logMemoryHit()

```
void Core::logMemoryHit (
             bool icache,
              mem_op_t mem_op_type,
              IntPtr address,
              MemModeled modeled =  MEM_MODELED_NONE,
              IntPtr eip = 0 )
```

Definition at line 222 of file core.cc.

References MemoryManagerBase::addL1Hits(), and getMemoryManager().

Referenced by TraceThread::handleInstructionWarmup(), lite::handleMemoryWrite(), lite::handleMemoryWrite↩Detailed(), and lite::handleMemoryWriteFaultinjection().

6.81.4.31 nativeMemOp()

```
MemoryResult Core::nativeMemOp (
              lock_signal_t lock_signal,
              mem_op_t mem_op_type,
              IntPtr d_addr,
             char * data_buffer,
              UInt32 data_size )
```

Definition at line 487 of file core.cc.

References TLock< T_LockCreator >::acquire(), applicationMemCopy(), LOCK, m_global_core_lock, make↩MemoryResult(), READ, READ_EX, TLock< T_LockCreator >::release(), HitWhere::UNKNOWN, UNLOCK, WR↩ITE, and SubsecondTime::Zero().

Referenced by accessMemory().

6.81.4.32 readInstructionMemory()

```
MemoryResult Core::readInstructionMemory (
              IntPtr address,
              UInt32 instruction_size )
```

Definition at line 228 of file core.cc.

References MemoryManagerBase::getCacheBlockSize(), getMemoryManager(), initiateMemoryAccess(), Mem↩Component::L1_ICACHE, HitWhere::L1I, LOG_PRINT, m_icache_last_block, makeMemoryResult(), Subsecond↩Time::MaxTime(), MEM_MODELED_COUNT_TLBTIME, NONE, and READ.

Referenced by TraceThread::handleCacheOnlyFunc(), MicroOpPerformanceModel::handleInstruction(), and TraceThread::handleInstructionWarmup().

6.81.4.33 setInstructionsCallback()

```
void Core::setInstructionsCallback (
              UInt64 instructions )  [inline]
```

Definition at line 123 of file core.h.

References m_instructions, and m_instructions_callback.

Referenced by FastForwardPerformanceManager::recalibrateInstructionsCallback(), and SamplingManager::recalibrateInstructionsCallback().

6.81.4.34 setState()

```
void Core::setState (
              State core_state )  [inline]
```

Definition at line 118 of file core.h.

References m_core_state.

Referenced by ThreadManager::createThread_unlocked(), ThreadManager::moveThread(), ThreadManager::onThreadExit(), and ThreadManager::onThreadStart().

6.81.4.35 setThread()

```
void Core::setThread (
              Thread * thread )  [inline]
```

Definition at line 105 of file core.h.

References m_thread.

Referenced by Thread::setCore().

6.81.4.36 updateSpinCount()

```
void Core::updateSpinCount (
              UInt64 instructions,
              SubsecondTime elapsed_time )  [inline]
```

Definition at line 129 of file core.h.

References m_spin_elapsed_time, m_spin_instructions, and m_spin_loops.

Referenced by SpinLoopDetector::commitBCT().

6.81.5 Friends And Related Function Documentation

6.81.5.1 InstructionModeling

```
friend class InstructionModeling [friend]
```

Definition at line 175 of file core.h.

6.81.6 Member Data Documentation

6.81.6.1 g_instructions_hpi_global

```
UInt64 Core::g_instructions_hpi_global = 0 [static], [protected]
```

Definition at line 184 of file core.h.

Referenced by hookPeriodicInsCall(), and hookPeriodicInsCheck().

6.81.6.2 g_instructions_hpi_global_callback

```
UInt64 Core::g_instructions_hpi_global_callback = 0 [static], [protected]
```

Definition at line 185 of file core.h.

Referenced by hookPeriodicInsCall(), and hookPeriodicInsCheck().

6.81.6.3 m_bbv

```
BbvCount Core::m_bbv [private]
```

Definition at line 146 of file core.h.

Referenced by countInstructions(), and getBbvCount().

6.81.6.4 m_cheetah_manager

```
CheetahManager* Core::m_cheetah_manager  [private]
```

Definition at line 148 of file core.h.

Referenced by accessMemoryFast(), getCheetahManager(), initiateMemoryAccess(), and ~Core().

6.81.6.5 m_clock_skew_minimization_client

```
ClockSkewMinimizationClient* Core::m_clock_skew_minimization_client  [private]
```

Definition at line 143 of file core.h.

Referenced by Core(), getClockSkewMinimizationClient(), and ~Core().

6.81.6.6 m_core_id

```
core_id_t Core::m_core_id  [private]
```

Definition at line 137 of file core.h.

Referenced by accessMemory(), countInstructions(), emulateCpuid(), getId(), and initiateMemoryAccess().

6.81.6.7 m_core_state

```
State Core::m_core_state  [private]
```

Definition at line 150 of file core.h.

Referenced by getState(), and setState().

6.81.6.8 m_dvfs_domain

```
const ComponentPeriod* Core::m_dvfs_domain  [private]
```

Definition at line 138 of file core.h.

Referenced by getDvfsDomain().

6.81.6.9 m_global_core_lock

Lock Core::m_global_core_lock [static], [private]

Definition at line 152 of file core.h.

Referenced by nativeMemOp().

6.81.6.10 m_icache_last_block

IntPtr Core::m_icache_last_block [private]

Definition at line 167 of file core.h.

Referenced by readInstructionMemory().

6.81.6.11 m_instructions

UInt64 Core::m_instructions [protected]

Definition at line 179 of file core.h.

Referenced by Core(), countInstructions(), getInstructionCount(), hookPeriodicInsCheck(), and setInstructions↩ Callback().

6.81.6.12 m_instructions_callback

UInt64 Core::m_instructions_callback [protected]

Definition at line 180 of file core.h.

Referenced by countInstructions(), disableInstructionsCallback(), getInstructionsCallback(), isEnabled↩ InstructionsCallback(), and setInstructionsCallback().

6.81.6.13 m_instructions_hpi_callback

UInt64 Core::m_instructions_hpi_callback [protected]

Definition at line 182 of file core.h.

Referenced by hookPeriodicInsCheck().

6.81.6.14 m_instructions_hpi_last

```
UInt64 Core::m_instructions_hpi_last  [protected]
```

Definition at line 183 of file core.h.

Referenced by hookPeriodicInsCheck().

6.81.6.15 m_mem_lock

```
Lock Core::m_mem_lock  [private]
```

Definition at line 144 of file core.h.

Referenced by initiateMemoryAccess().

6.81.6.16 m_memory_manager

```
MemoryManagerBase* Core::m_memory_manager  [private]
```

Definition at line 139 of file core.h.

Referenced by Core(), getMemoryManager(), and ~Core().

6.81.6.17 m_network

```
Network* Core::m_network  [private]
```

Definition at line 141 of file core.h.

Referenced by Core(), getNetwork(), and ~Core().

6.81.6.18 m_performance_model

```
PerformanceModel* Core::m_performance_model  [private]
```

Definition at line 142 of file core.h.

Referenced by accessMemoryFast(), Core(), countInstructions(), getPerformanceModel(), initiateMemoryAccess(), and ~Core().

6.81.6.19 m_shmem_perf_model

```
ShmemPerfModel* Core::m_shmem_perf_model  [private]
```

Definition at line 145 of file core.h.

Referenced by Core(), getShmemPerfModel(), and ~Core().

6.81.6.20 m_spin_elapsed_time

```
SubsecondTime Core::m_spin_elapsed_time  [private]
```

Definition at line 171 of file core.h.

Referenced by Core(), and updateSpinCount().

6.81.6.21 m_spin_instructions

```
UInt64 Core::m_spin_instructions  [private]
```

Definition at line 170 of file core.h.

Referenced by Core(), and updateSpinCount().

6.81.6.22 m_spin_loops

```
UInt64 Core::m_spin_loops  [private]
```

Definition at line 169 of file core.h.

Referenced by Core(), and updateSpinCount().

6.81.6.23 m_thread

```
Thread* Core::m_thread  [private]
```

Definition at line 140 of file core.h.

Referenced by getThread(), and setThread().

6.81.6.24 m_topology_info

```
TopologyInfo* Core::m_topology_info  [private]
```

Definition at line 147 of file core.h.

Referenced by emulateCpuid(), getTopologyInfo(), and ~Core().

The documentation for this class was generated from the following files:

- common/core/ **core.h**
- common/core/ **core.cc**

6.82 CoreManager Class Reference

```
#include <core_manager.h>
```

Public Types

- enum **ThreadType** { **INVALID**, **APP_THREAD**, **CORE_THREAD**, **SIM_THREAD** }

Public Member Functions

- **CoreManager** ()
- ~**CoreManager** ()
- void **initializeCommId** (**SInt32** comm_id)
- void **initializeThread** (**core_id_t** core_id)
- void **terminateThread** ()
- **core_id_t** **registerSimThread** (**ThreadType** type)
- **core_id_t** **getCurrentCoreID** (int threadIndex=-1)
- **Core** * **getCurrentCore** (int threadIndex=-1)
- **Core** * **getCoreFromID** (**core_id_t** core_id)
- bool **amiUserThread** ()
- bool **amiCoreThread** ()
- bool **amiSimThread** ()

Private Attributes

- **UInt32** * **tid_map**
- **TLS** * **m_core_tls**
- **TLS** * **m_thread_type_tls**
- **UInt32** **m_num_registered_sim_threads**
- **UInt32** **m_num_registered_core_threads**
- **Lock** **m_num_registered_threads_lock**
- std::vector< **Core** * > **m_cores**

6.82.1 Detailed Description

Definition at line 17 of file core_manager.h.

6.82.2 Member Enumeration Documentation

6.82.2.1 ThreadType

enum **CoreManager::ThreadType**

Enumerator

INVALID	
APP_THREAD	
CORE_THREAD	
SIM_THREAD	

Definition at line 23 of file core_manager.h.

6.82.3 Constructor & Destructor Documentation

6.82.3.1 CoreManager()

```
CoreManager::CoreManager ( )
```

Definition at line 18 of file core_manager.cc.

References Config::getSingleton(), Config::getTotalCores(), LOG_PRINT, and m_cores.

6.82.3.2 ~CoreManager()

```
CoreManager::~CoreManager ( )
```

Definition at line 34 of file core_manager.cc.

References m_core_tls, m_cores, and m_thread_type_tls.

6.82.4 Member Function Documentation

6.82.4.1 amiCoreThread()

```
bool CoreManager::amiCoreThread ( )
```

Definition at line 120 of file core_manager.cc.

References CORE_THREAD, TLS::getInt(), and m_thread_type_tls.

6.82.4.2 amiSimThread()

```
bool CoreManager::amiSimThread ( )
```

Definition at line 115 of file core_manager.cc.

References TLS::getInt(), m_thread_type_tls, and SIM_THREAD.

Referenced by Log::discoverCore().

6.82.4.3 amiUserThread()

```
bool CoreManager::amiUserThread ( )
```

Definition at line 125 of file core_manager.cc.

References APP_THREAD, TLS::getInt(), and m_thread_type_tls.

6.82.4.4 getCoreFromID()

```
Core * CoreManager::getCoreFromID (
            core_id_t core_id )
```

Definition at line 76 of file core_manager.cc.

References Config::getSingleton(), LOG_ASSERT_ERROR, and m_cores.

6.82.4.5 getCurrentCore()

```
Core* CoreManager::getCurrentCore (
            int threadIndex = -1 )  [inline]
```

Definition at line 43 of file core_manager.h.

References TLS::getPtr(), and m_core_tls.

Referenced by getCurrentCoreID(), initializeCommId(), and registerSimThread().

6.82.4.6 getCurrentCoreID()

```
core_id_t CoreManager::getCurrentCoreID (
            int threadIndex = -1 )  [inline]
```

Definition at line 35 of file core_manager.h.

References getCurrentCore(), Core::getId(), and INVALID_CORE_ID.

Referenced by Log::discoverCore(), and initializeCommId().

6.82.4.7 initializeCommId()

```
void CoreManager::initializeCommId (
              SInt32 comm_id )
```

Definition at line 43 of file core_manager.cc.

References getCurrentCore(), getCurrentCoreID(), Config::getSingleton(), INVALID_CORE_ID, LOG_ASSERT_↩ ERROR, LOG_PRINT, and Config::updateCommToCoreMap().

6.82.4.8 initializeThread()

```
void CoreManager::initializeThread (
              core_id_t core_id )
```

Definition at line 60 of file core_manager.cc.

References APP_THREAD, TLS::get(), LOG_ASSERT_ERROR, LOG_PRINT, m_core_tls, m_cores, m_thread_↩ type_tls, TLS::set(), and TLS::setInt().

6.82.4.9 registerSimThread()

```
core_id_t CoreManager::registerSimThread (
              ThreadType type )
```

Definition at line 82 of file core_manager.cc.

References CORE_THREAD, getCurrentCore(), Core::getId(), Config::getSingleton(), LOG_ASSERT_ERROR, L↩ OG_PRINT_ERROR, m_core_tls, m_cores, m_num_registered_core_threads, m_num_registered_sim_threads, m_num_registered_threads_lock, m_thread_type_tls, TLS::set(), TLS::setInt(), and SIM_THREAD.

6.82.4.10 terminateThread()

```
void CoreManager::terminateThread ( )
```

Definition at line 70 of file core_manager.cc.

References TLS::get(), LOG_ASSERT_WARNING, m_core_tls, and TLS::set().

6.82.5 Member Data Documentation

6.82.5.1 m_core_tls

```
TLS* CoreManager::m_core_tls  [private]
```

Definition at line 56 of file core_manager.h.

Referenced by getCurrentCore(), initializeThread(), registerSimThread(), terminateThread(), and ~CoreManager().

6.82.5.2 m_cores

```
std::vector< Core*> CoreManager::m_cores  [private]
```

Definition at line 63 of file core_manager.h.

Referenced by CoreManager(), getCoreFromID(), initializeThread(), registerSimThread(), and ~CoreManager().

6.82.5.3 m_num_registered_core_threads

```
UInt32 CoreManager::m_num_registered_core_threads  [private]
```

Definition at line 60 of file core_manager.h.

Referenced by registerSimThread().

6.82.5.4 m_num_registered_sim_threads

```
UInt32 CoreManager::m_num_registered_sim_threads  [private]
```

Definition at line 59 of file core_manager.h.

Referenced by registerSimThread().

6.82.5.5 m_num_registered_threads_lock

```
Lock CoreManager::m_num_registered_threads_lock  [private]
```

Definition at line 61 of file core_manager.h.

Referenced by registerSimThread().

6.82.5.6 m_thread_type_tls

```
TLS* CoreManager::m_thread_type_tls  [private]
```

Definition at line 57 of file core_manager.h.

Referenced by amiCoreThread(), amiSimThread(), amiUserThread(), initializeThread(), registerSimThread(), and ~CoreManager().

6.82.5.7 tid_map

```
UInt32* CoreManager::tid_map  [private]
```

Definition at line 55 of file core_manager.h.

The documentation for this class was generated from the following files:

- common/system/ **core_manager.h**
- common/system/ **core_manager.cc**

6.83 CoreModel Class Reference

```
#include <core_model.h>
```

Inheritance diagram for CoreModel:

Public Member Functions

- virtual **IntervalContention** * **createIntervalContentionModel** (const **Core** *core) const =0
- virtual unsigned int **getLongLatencyCutoff** () const =0
- virtual **RobContention** * **createRobContentionModel** (const **Core** *core) const =0
- virtual **Allocator** * **createDMOAllocator** () const =0
- virtual **DynamicMicroOp** * **createDynamicMicroOp** (**Allocator** *alloc, const **MicroOp** *uop, **ComponentPeriod** period) const =0
- virtual unsigned int **getInstructionLatency** (const **MicroOp** *uop) const =0
- virtual unsigned int **getAluLatency** (const **MicroOp** *uop) const =0
- virtual unsigned int **getBypassLatency** (const **DynamicMicroOp** *uop) const =0
- virtual unsigned int **getLongestLatency** () const =0

Static Public Member Functions

- static const **CoreModel** * **getCoreModel** (String type)

Static Private Attributes

- static std::map< String, const **CoreModel** * > **s_core_models**

6.83.1 Detailed Description

Definition at line 17 of file core_model.h.

6.83.2 Member Function Documentation

6.83.2.1 createDMOAllocator()

```
virtual Allocator* CoreModel::createDMOAllocator ( ) const [pure virtual]
```

Implemented in **BaseCoreModel< T >** (p. 102).

6.83.2.2 createDynamicMicroOp()

```
virtual DynamicMicroOp* CoreModel::createDynamicMicroOp (
            Allocator * alloc,
           const MicroOp * uop,
            ComponentPeriod period ) const [pure virtual]
```

Implemented in **BaseCoreModel< T >** (p. 102), **BaseCoreModel< DynamicMicroOpNehalem >** (p. 102), **BaseCoreModel< DynamicMicroOpBoomV1 >** (p. 102), **CoreModelBoomV1** (p. 408), and **CoreModelNehalem** (p. 411).

Referenced by MicroOpPerformanceModel::handleInstruction().

6.83.2.3 createIntervalContentionModel()

```
virtual  IntervalContention* CoreModel::createIntervalContentionModel (
             const  Core * core ) const  [pure virtual]
```

Implemented in **CoreModelBoomV1** (p. 408), and **CoreModelNehalem** (p. 411).

6.83.2.4 createRobContentionModel()

```
virtual  RobContention* CoreModel::createRobContentionModel (
             const  Core * core ) const  [pure virtual]
```

Implemented in **CoreModelBoomV1** (p. 408), and **CoreModelNehalem** (p. 411).

6.83.2.5 getAluLatency()

```
virtual unsigned int CoreModel::getAluLatency (
             const  MicroOp * uop ) const  [pure virtual]
```

Implemented in **CoreModelBoomV1** (p. 408), and **CoreModelNehalem** (p. 412).

Referenced by RobContentionBoomV1::doIssue(), and RobContentionNehalem::doIssue().

6.83.2.6 getBypassLatency()

```
virtual unsigned int CoreModel::getBypassLatency (
             const  DynamicMicroOp * uop ) const  [pure virtual]
```

Implemented in **CoreModelBoomV1** (p. 408), and **CoreModelNehalem** (p. 412).

Referenced by IntervalTimer::dispatchInstruction(), and MicroOpPerformanceModel::handleInstruction().

6.83.2.7 getCoreModel()

```
const  CoreModel * CoreModel::getCoreModel (
             String type )  [static]
```

Definition at line 8 of file core_model.cc.

References LOG_PRINT_ERROR, and s_core_models.

6.83.2.8 getInstructionLatency()

```
virtual unsigned int CoreModel::getInstructionLatency (
            const  MicroOp * uop ) const  [pure virtual]
```

Implemented in **CoreModelBoomV1** (p. 409), and **CoreModelNehalem** (p. 412).

Referenced by DynamicMicroOp::DynamicMicroOp().

6.83.2.9 getLongestLatency()

```
virtual unsigned int CoreModel::getLongestLatency ( ) const  [pure virtual]
```

Implemented in **CoreModelBoomV1** (p. 409), and **CoreModelNehalem** (p. 412).

6.83.2.10 getLongLatencyCutoff()

```
virtual unsigned int CoreModel::getLongLatencyCutoff ( ) const  [pure virtual]
```

Implemented in **CoreModelBoomV1** (p. 409), and **CoreModelNehalem** (p. 413).

Referenced by MicroOpPerformanceModel::handleInstruction(), DynamicMicroOp::isLongLatencyLoad(), and Windows::updateCriticalPathTail().

6.83.3 Member Data Documentation

6.83.3.1 s_core_models

```
std::map< String, const  CoreModel * > CoreModel::s_core_models  [static], [private]
```

Definition at line 20 of file core_model.h.

Referenced by getCoreModel().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/core_model/ **core_model.h**
- common/performance_model/performance_models/core_model/ **core_model.cc**

6.84 CoreModelBoomV1 Class Reference

`#include <core_model_boom_v1.h>`

Inheritance diagram for CoreModelBoomV1:

Public Member Functions

- **CoreModelBoomV1** ()
- virtual **IntervalContention** * **createIntervalContentionModel** (const **Core** *core) const
- virtual **RobContention** * **createRobContentionModel** (const **Core** *core) const
- virtual **DynamicMicroOp** * **createDynamicMicroOp** (**Allocator** *alloc, const **MicroOp** *uop, **ComponentPeriod** period) const
- virtual unsigned int **getInstructionLatency** (const **MicroOp** *uop) const
- virtual unsigned int **getAluLatency** (const **MicroOp** *uop) const
- virtual unsigned int **getBypassLatency** (const **DynamicMicroOp** *uop) const
- virtual unsigned int **getLongestLatency** () const
- virtual unsigned int **getLongLatencyCutoff** () const

Private Attributes

- unsigned int **m_lll_cutoff**

Additional Inherited Members

6.84.1 Detailed Description

Definition at line 7 of file core_model_boom_v1.h.

6.84.2 Constructor & Destructor Documentation

6.84.2.1 CoreModelBoomV1()

`CoreModelBoomV1::CoreModelBoomV1 ( )`

Definition at line 26 of file core_model_boom_v1.cc.

References instrlist, instructionLatencies, m_lll_cutoff, and rv_op_last.

6.84.3 Member Function Documentation

6.84.3.1 createDynamicMicroOp()

```
DynamicMicroOp * CoreModelBoomV1::createDynamicMicroOp (
            Allocator * alloc,
            const  MicroOp * uop,
            ComponentPeriod period ) const  [virtual]
```

Implements **CoreModel** (p. 404).

Definition at line 100 of file core_model_boom_v1.cc.

References DynamicMicroOpBoomV1::getAlu(), DynamicMicroOpBoomV1::getBypassType(), DynamicMicroOp↩BoomV1::getPort(), DynamicMicroOpBoomV1::uop_alu, DynamicMicroOpBoomV1::uop_bypass, and Dynamic↩MicroOpBoomV1::uop_port.

6.84.3.2 createIntervalContentionModel()

```
IntervalContention * CoreModelBoomV1::createIntervalContentionModel (
            const  Core * core ) const  [virtual]
```

Implements **CoreModel** (p. 404).

Definition at line 90 of file core_model_boom_v1.cc.

6.84.3.3 createRobContentionModel()

```
RobContention * CoreModelBoomV1::createRobContentionModel (
            const  Core * core ) const  [virtual]
```

Implements **CoreModel** (p. 405).

Definition at line 95 of file core_model_boom_v1.cc.

6.84.3.4 getAluLatency()

```
unsigned int CoreModelBoomV1::getAluLatency (
            const  MicroOp * uop ) const  [virtual]
```

Implements **CoreModel** (p. 405).

Definition at line 58 of file core_model_boom_v1.cc.

References getInstructionLatency(), MicroOp::getInstructionOpcode(), rv_op_div, rv_op_divd, rv_op_divu, rv_op↩_divud, rv_op_divuw, rv_op_divw, rv_op_fdiv_d, rv_op_fdiv_q, and rv_op_fdiv_s.

6.84.3.5 getBypassLatency()

```
unsigned int CoreModelBoomV1::getBypassLatency (
            const  DynamicMicroOp * uop ) const  [virtual]
```

Implements **CoreModel** (p. 405).

Definition at line 77 of file core_model_boom_v1.cc.

References bypassLatencies, DynamicMicroOpBoomV1::getBypassType(), DynamicMicroOp::getCoreSpecific↩Info(), LOG_ASSERT_ERROR, and DynamicMicroOpBoomV1::UOP_BYPASS_SIZE.

6.84.3.6 getInstructionLatency()

```
unsigned int CoreModelBoomV1::getInstructionLatency (
            const  MicroOp * uop ) const  [virtual]
```

Implements **CoreModel** (p. 405).

Definition at line 51 of file core_model_boom_v1.cc.

References MicroOp::getInstructionOpcode(), instructionLatencies, LOG_ASSERT_ERROR, and rv_op_last.

Referenced by getAluLatency().

6.84.3.7 getLongestLatency()

```
unsigned int CoreModelBoomV1::getLongestLatency ( ) const  [virtual]
```

Implements **CoreModel** (p. 406).

Definition at line 85 of file core_model_boom_v1.cc.

6.84.3.8 getLongLatencyCutoff()

```
virtual unsigned int CoreModelBoomV1::getLongLatencyCutoff ( ) const  [inline], [virtual]
```

Implements **CoreModel** (p. 406).

Definition at line 24 of file core_model_boom_v1.h.

References m_lll_cutoff.

6.84.4 Member Data Documentation

6.84.4.1 m_lll_cutoff

```
unsigned int CoreModelBoomV1::m_lll_cutoff  [private]
```

Definition at line 10 of file core_model_boom_v1.h.

Referenced by CoreModelBoomV1(), and getLongLatencyCutoff().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/core_model/ **core_model_boom_v1.h**
- common/performance_model/performance_models/core_model/ **core_model_boom_v1.cc**

6.85 CoreModelNehalem Class Reference

```
#include <core_model_nehalem.h>
```

Inheritance diagram for CoreModelNehalem:

Public Member Functions

- **CoreModelNehalem** ()
- virtual **IntervalContention** * **createIntervalContentionModel** (const **Core** *core) const
- virtual **RobContention** * **createRobContentionModel** (const **Core** *core) const
- virtual **DynamicMicroOp** * **createDynamicMicroOp** (**Allocator** *alloc, const **MicroOp** *uop, **ComponentPeriod** period) const
- virtual unsigned int **getInstructionLatency** (const **MicroOp** *uop) const
- virtual unsigned int **getAluLatency** (const **MicroOp** *uop) const
- virtual unsigned int **getBypassLatency** (const **DynamicMicroOp** *uop) const
- virtual unsigned int **getLongestLatency** () const
- virtual unsigned int **getLongLatencyCutoff** () const

Private Attributes

- unsigned int **m_lll_cutoff**

Additional Inherited Members

6.85.1 Detailed Description

Definition at line 7 of file core_model_nehalem.h.

6.85.2 Constructor & Destructor Documentation

6.85.2.1 CoreModelNehalem()

```
CoreModelNehalem::CoreModelNehalem ( )
```

Definition at line 15 of file core_model_nehalem.cc.

References bypassLatencies, instructionLatencies, m_lll_cutoff, DynamicMicroOpNehalem::UOP_BYPASS_FP_↩STORE, DynamicMicroOpNehalem::UOP_BYPASS_LOAD_FP, and DynamicMicroOpNehalem::UOP_BYPASS↩_NONE.

6.85.3 Member Function Documentation

6.85.3.1 createDynamicMicroOp()

```
DynamicMicroOp * CoreModelNehalem::createDynamicMicroOp (
            Allocator * alloc,
           const  MicroOp * uop,
            ComponentPeriod period ) const  [virtual]
```

Implements **CoreModel** (p. 404).

Definition at line 185 of file core_model_nehalem.cc.

References DynamicMicroOpNehalem::getAlu(), DynamicMicroOpNehalem::getBypassType(), DynamicMicroOp↩Nehalem::getPort(), DynamicMicroOpNehalem::uop_alu, DynamicMicroOpNehalem::uop_bypass, and Dynamic↩MicroOpNehalem::uop_port.

6.85.3.2 createIntervalContentionModel()

```
IntervalContention * CoreModelNehalem::createIntervalContentionModel (
           const  Core * core ) const  [virtual]
```

Implements **CoreModel** (p. 404).

Definition at line 175 of file core_model_nehalem.cc.

6.85.3.3 createRobContentionModel()

```
RobContention * CoreModelNehalem::createRobContentionModel (
            const  Core * core ) const  [virtual]
```

Implements **CoreModel** (p. 405).

Definition at line 180 of file core_model_nehalem.cc.

6.85.3.4 getAluLatency()

```
unsigned int CoreModelNehalem::getAluLatency (
            const  MicroOp * uop ) const  [virtual]
```

Implements **CoreModel** (p. 405).

Definition at line 147 of file core_model_nehalem.cc.

References getInstructionLatency(), MicroOp::getInstructionOpcode(), and MicroOp::getOperandSize().

6.85.3.5 getBypassLatency()

```
unsigned int CoreModelNehalem::getBypassLatency (
            const  DynamicMicroOp * uop ) const  [virtual]
```

Implements **CoreModel** (p. 405).

Definition at line 162 of file core_model_nehalem.cc.

References bypassLatencies, DynamicMicroOpNehalem::getBypassType(), DynamicMicroOp::getCoreSpecific↩Info(), LOG_ASSERT_ERROR, and DynamicMicroOpNehalem::UOP_BYPASS_SIZE.

6.85.3.6 getInstructionLatency()

```
unsigned int CoreModelNehalem::getInstructionLatency (
            const  MicroOp * uop ) const  [virtual]
```

Implements **CoreModel** (p. 405).

Definition at line 140 of file core_model_nehalem.cc.

References MicroOp::getInstructionOpcode(), instructionLatencies, and LOG_ASSERT_ERROR.

Referenced by getAluLatency().

6.85.3.7 getLongestLatency()

```
unsigned int CoreModelNehalem::getLongestLatency ( ) const  [virtual]
```

Implements **CoreModel** (p. 406).

Definition at line 170 of file core_model_nehalem.cc.

6.85.3.8 getLongLatencyCutoff()

```
virtual unsigned int CoreModelNehalem::getLongLatencyCutoff ( ) const  [inline], [virtual]
```

Implements **CoreModel** (p. 406).

Definition at line 24 of file core_model_nehalem.h.

References m_lll_cutoff.

6.85.4 Member Data Documentation

6.85.4.1 m_lll_cutoff

```
unsigned int CoreModelNehalem::m_lll_cutoff  [private]
```

Definition at line 10 of file core_model_nehalem.h.

Referenced by CoreModelNehalem(), and getLongLatencyCutoff().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/core_model/ **core_model_nehalem.h**
- common/performance_model/performance_models/core_model/ **core_model_nehalem.cc**

6.86 CoreThread Class Reference

```
#include <core_thread.h>
```

Inheritance diagram for CoreThread:

Public Member Functions

- **CoreThread** ()
- ∼**CoreThread** ()
- void **spawn** ()

Private Member Functions

- void **run** ()

Static Private Member Functions

- static void **terminateFunc** (void ∗vp, **NetPacket** pkt)

Private Attributes

- **_Thread** ∗ **m_thread**

Additional Inherited Members

6.86.1 Detailed Description

Definition at line 8 of file core_thread.h.

6.86.2 Constructor & Destructor Documentation

6.86.2.1 CoreThread()

```
CoreThread::CoreThread ( )
```

Definition at line 12 of file core_thread.cc.

6.86.2.2 ∼CoreThread()

```
CoreThread::∼CoreThread ( )
```

Definition at line 17 of file core_thread.cc.

References m_thread.

6.86.3 Member Function Documentation

6.86.3.1 run()

```
void CoreThread::run ( )  [private], [virtual]
```

Implements **Runnable** (p. 1106).

Definition at line 22 of file core_thread.cc.

References CoreManager::CORE_THREAD, CORE_THREAD_TERMINATE_THREADS, PerformanceModel↩ ::iterate(), itostr(), LOG_PRINT, Network::registerCallback(), and terminateFunc().

6.86.3.2 spawn()

```
void CoreThread::spawn ( )
```

Definition at line 53 of file core_thread.cc.

References _Thread::create(), m_thread, and _Thread::run().

Referenced by SimThreadManager::spawnSimThreads().

6.86.3.3 terminateFunc()

```
void CoreThread::terminateFunc (
             void * vp,
              NetPacket pkt )  [static], [private]
```

Definition at line 59 of file core_thread.cc.

Referenced by run().

6.86.4 Member Data Documentation

6.86.4.1 m_thread

```
_Thread* CoreThread::m_thread  [private]
```

Definition at line 21 of file core_thread.h.

Referenced by spawn(), and ~CoreThread().

The documentation for this class was generated from the following files:

- common/system/ **core_thread.h**
- common/system/ **core_thread.cc**

6.87 cpuid_result_t Struct Reference

`#include <cpuid.h>`

Public Attributes

- **UInt32 eax**
- **UInt32 ebx**
- **UInt32 ecx**
- **UInt32 edx**

6.87.1 Detailed Description

Definition at line 4 of file cpuid.h.

6.87.2 Member Data Documentation

6.87.2.1 eax

```
UInt32 cpuid_result_t::eax
```

Definition at line 6 of file cpuid.h.

Referenced by cpuid(), Core::emulateCpuid(), handleCpuid(), and TraceThread::handleEmuFunc().

6.87.2.2 ebx

UInt32 cpuid_result_t::ebx

Definition at line 6 of file cpuid.h.

Referenced by cpuid(), Core::emulateCpuid(), handleCpuid(), and TraceThread::handleEmuFunc().

6.87.2.3 ecx

UInt32 cpuid_result_t::ecx

Definition at line 6 of file cpuid.h.

Referenced by cpuid(), Core::emulateCpuid(), handleCpuid(), and TraceThread::handleEmuFunc().

6.87.2.4 edx

UInt32 cpuid_result_t::edx

Definition at line 6 of file cpuid.h.

Referenced by cpuid(), Core::emulateCpuid(), handleCpuid(), and TraceThread::handleEmuFunc().

The documentation for this struct was generated from the following file:

- common/misc/ **cpuid.h**

6.88 Allocator::DataElement Struct Reference

Public Attributes

- **Allocator** * **allocator**
- char **data** [ ]

6.88.1 Detailed Description

Definition at line 15 of file allocator.h.

6.88.2 Member Data Documentation

6.88.2.1 allocator

```
Allocator* Allocator::DataElement::allocator
```

Definition at line 17 of file allocator.h.

Referenced by TypedAllocator< T, MaxItems >::alloc(), and Allocator::dealloc().

6.88.2.2 data

```
char Allocator::DataElement::data[]
```

Definition at line 18 of file allocator.h.

Referenced by TypedAllocator< T, MaxItems >::alloc().

The documentation for this struct was generated from the following file:

- common/misc/ **allocator.h**

6.89 config::config_parser::definition< ScannerT > Struct Template Reference

```
#include <config_file_grammar.hpp>
```

Public Member Functions

- **definition** (**config_parser** const &self)
- rule< ScannerT > const & **start** () const

Public Attributes

- rule< ScannerT > **r_file**
- rule< ScannerT > **r_config**
- rule< ScannerT > **r_config_node**
- rule< ScannerT > **r_section**
- rule< ScannerT > **r_key**
- rule< ScannerT > **r_key_name**
- rule< ScannerT > **r_key_array**
- rule< ScannerT > **r_value**
- rule< ScannerT > **r_value_long**
- rule< ScannerT > **r_value_single**
- rule< ScannerT > **r_value_single_or_empty**
- rule< ScannerT > **r_value_array**
- rule< ScannerT > **r_value_array2**
- rule< ScannerT > **r_value_span**
- rule< ScannerT > **r_key_node**
- rule< ScannerT > **r_section_node**
- rule< ScannerT > **r_section_name**
- rule< ScannerT > **r_section_name_node**
- rule< ScannerT > **r_section_name_node_node**
- rule< ScannerT > **r_string**

6.89.1 Detailed Description

template<typename ScannerT>
struct config::config_parser::definition< ScannerT >

Definition at line 79 of file config_file_grammar.hpp.

6.89.2 Constructor & Destructor Documentation

6.89.2.1 definition()

```
template<typename ScannerT >
config::config_parser::definition< ScannerT >::  definition (
              config_parser const & self )  [inline]
```

Definition at line 80 of file config_file_grammar.hpp.

References config::keyID, config::keyNameID, config::keySeparatorID, config::keyValueArrayID, config::keyValueID, config::keyValueSpanID, config::long_p, and config::sectionNameID.

6.89.3 Member Function Documentation

6.89.3.1 start()

```
template<typename ScannerT >
rule<ScannerT> const&  config::config_parser::definition< ScannerT >::start ( ) const  [inline]
```

Definition at line 150 of file config_file_grammar.hpp.

6.89.4 Member Data Documentation

6.89.4.1 r_config

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_config
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.2 r_config_node

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_config_node
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.3 r_file

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_file
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.4 r_key

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_key
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.5 r_key_array

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_key_array
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.6 r_key_name

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_key_name
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.7 r_key_node

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_key_node
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.8 r_section

```
template<typename ScannerT >
rule<ScannerT> config::config_parser::definition< ScannerT >::r_section
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.9 r_section_name

```
template<typename ScannerT >
rule<ScannerT> config::config_parser::definition< ScannerT >::r_section_name
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.10 r_section_name_node

```
template<typename ScannerT >
rule<ScannerT> config::config_parser::definition< ScannerT >::r_section_name_node
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.11 r_section_name_node_node

```
template<typename ScannerT >
rule<ScannerT> config::config_parser::definition< ScannerT >::r_section_name_node_node
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.12 r_section_node

```
template<typename ScannerT >
rule<ScannerT> config::config_parser::definition< ScannerT >::r_section_node
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.13 r_string

```
template<typename ScannerT >
rule<ScannerT> config::config_parser::definition< ScannerT >::r_string
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.14 r_value

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_value
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.15 r_value_array

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_value_array
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.16 r_value_array2

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_value_array2
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.17 r_value_long

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_value_long
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.18 r_value_single

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_value_single
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.19 r_value_single_or_empty

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_value_single_or_empty
```

Definition at line 147 of file config_file_grammar.hpp.

6.89.4.20 r_value_span

```
template<typename ScannerT >
rule<ScannerT>  config::config_parser::definition< ScannerT >::r_value_span
```

Definition at line 147 of file config_file_grammar.hpp.

The documentation for this struct was generated from the following file:

- common/config/ **config_file_grammar.hpp**

6.90 DelayInstruction Class Reference

```
#include <instruction.h>
```

Inheritance diagram for DelayInstruction:

Public Types

- enum **delay_type_t** { **DVFS_TRANSITION**, **NUM_TYPES** }

Public Member Functions

- **DelayInstruction** (**SubsecondTime** cost, **delay_type_t** delay_type)
- **delay_type_t** **getDelayType** () const

Private Attributes

- **delay_type_t** **m_delay_type**

Additional Inherited Members

6.90.1 Detailed Description

Definition at line 220 of file instruction.h.

6.90.2 Member Enumeration Documentation

6.90.2.1 delay_type_t

```
enum  DelayInstruction::delay_type_t
```

Enumerator

DVFS_TRANSITION	
NUM_TYPES	

Definition at line 223 of file instruction.h.

6.90.3 Constructor & Destructor Documentation

6.90.3.1 DelayInstruction()

```
DelayInstruction::DelayInstruction (
              SubsecondTime cost,
              delay_type_t delay_type )  [inline]
```

Definition at line 227 of file instruction.h.

6.90.4 Member Function Documentation

6.90.4.1 getDelayType()

```
delay_type_t DelayInstruction::getDelayType ( ) const  [inline]
```

Definition at line 231 of file instruction.h.

References m_delay_type.

Referenced by PerformanceModel::handleIdleInstruction().

6.90.5 Member Data Documentation

6.90.5.1 m_delay_type

```
delay_type_t DelayInstruction::m_delay_type  [private]
```

Definition at line 233 of file instruction.h.

Referenced by getDelayType().

The documentation for this class was generated from the following file:

- common/performance_model/ **instruction.h**

6.91 Directory Class Reference

`#include <directory.h>`

Public Types

- enum **DirectoryType** { **FULL_MAP** = 0, **LIMITED_NO_BROADCAST**, **LIMITLESS**, **NUM_DIRECTOR↩ Y_TYPES** }

Public Member Functions

- **Directory** (**core_id_t** core_id, String directory_type_str, **UInt32** num_entries, **UInt32** max_hw_sharers, **UInt32** max_num_sharers)
- ~**Directory** ()
- **DirectoryEntry** * **getDirectoryEntry** (**UInt32** entry_num)
- void **setDirectoryEntry** (**UInt32** entry_num, **DirectoryEntry** *directory_entry)
- **DirectoryEntry** * **createDirectoryEntry** ()
- template<class DirectorySharers >
  DirectoryEntry * **createDirectoryEntrySized** ()
- **UInt32** **getMaxHwSharers** () const

Static Public Member Functions

- static **DirectoryType** **parseDirectoryType** (String directory_type_str)

Private Attributes

- **DirectoryType** **m_directory_type**
- **UInt32** **m_num_entries**
- **UInt64** **m_num_entries_allocated**
- **UInt32** **m_max_hw_sharers**
- **UInt32** **m_use_max_hw_sharers**
- **UInt32** **m_max_num_sharers**
- **SubsecondTime** **m_limitless_software_trap_penalty**
- **DirectoryEntry** ** **m_directory_entry_list**

6.91.1 Detailed Description

Definition at line 8 of file directory.h.

6.91.2 Member Enumeration Documentation

6.91.2.1 DirectoryType

`enum` **Directory::DirectoryType**

Enumerator

FULL_MAP	
LIMITED_NO_BROADCAST	
LIMITLESS	
NUM_DIRECTORY_TYPES	

Definition at line 11 of file directory.h.

6.91.3 Constructor & Destructor Documentation

6.91.3.1 Directory()

```
Directory::Directory (
             core_id_t core_id,
            String directory_type_str,
             UInt32 num_entries,
             UInt32 max_hw_sharers,
             UInt32 max_num_sharers )
```

Definition at line 10 of file directory.cc.

References LIMITLESS, LOG_PRINT_ERROR, m_directory_entry_list, m_directory_type, m_limitless_software_trap_penalty, m_num_entries, m_num_entries_allocated, SubsecondTime::NS(), parseDirectoryType(), and registerStatsMetric().

6.91.3.2 ~Directory()

```
Directory::~Directory ( )
```

Definition at line 42 of file directory.cc.

References m_directory_entry_list, and m_num_entries.

6.91.4 Member Function Documentation

6.91.4.1 createDirectoryEntry()

```
DirectoryEntry * Directory::createDirectoryEntry ( )
```

Definition at line 88 of file directory.cc.

References m_max_num_sharers.

Referenced by getDirectoryEntry(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCache::replaceDirectoryEntry().

6.91.4.2 createDirectoryEntrySized()

```
template<class DirectorySharers >
DirectoryEntry * Directory::createDirectoryEntrySized
```

Definition at line 106 of file directory.cc.

References FULL_MAP, LIMITED_NO_BROADCAST, LIMITLESS, LOG_PRINT_ERROR, m_directory_type, m_limitless_software_trap_penalty, m_max_hw_sharers, m_max_num_sharers, and m_use_max_hw_sharers.

6.91.4.3 getDirectoryEntry()

```
DirectoryEntry * Directory::getDirectoryEntry (
            UInt32 entry_num )
```

Definition at line 53 of file directory.cc.

References createDirectoryEntry(), LOG_ASSERT_ERROR, m_directory_entry_list, m_num_entries, and m_num_entries_allocated.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getReplacementCandidates(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCache::replaceDirectoryEntry().

6.91.4.4 getMaxHwSharers()

```
UInt32 Directory::getMaxHwSharers ( ) const  [inline]
```

Definition at line 41 of file directory.h.

References m_use_max_hw_sharers.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getMaxHwSharers().

6.91.4.5 parseDirectoryType()

```
Directory::DirectoryType Directory::parseDirectoryType (
            String directory_type_str )  [static]
```

Definition at line 72 of file directory.cc.

References FULL_MAP, LIMITED_NO_BROADCAST, LIMITLESS, and LOG_PRINT_ERROR.

Referenced by Directory().

6.91.4.6 setDirectoryEntry()

```
void Directory::setDirectoryEntry (
              UInt32 entry_num,
              DirectoryEntry * directory_entry )
```

Definition at line 66 of file directory.cc.

References m_directory_entry_list.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCache::replaceDirectoryEntry().

6.91.5 Member Data Documentation

6.91.5.1 m_directory_entry_list

```
DirectoryEntry** Directory::m_directory_entry_list  [private]
```

Definition at line 30 of file directory.h.

Referenced by Directory(), getDirectoryEntry(), setDirectoryEntry(), and ~Directory().

6.91.5.2 m_directory_type

```
DirectoryType Directory::m_directory_type  [private]
```

Definition at line 20 of file directory.h.

Referenced by createDirectoryEntrySized(), and Directory().

6.91.5.3 m_limitless_software_trap_penalty

```
SubsecondTime Directory::m_limitless_software_trap_penalty  [private]
```

Definition at line 28 of file directory.h.

Referenced by createDirectoryEntrySized(), and Directory().

6.91.5.4 m_max_hw_sharers

UInt32 Directory::m_max_hw_sharers [private]

Definition at line 23 of file directory.h.

Referenced by createDirectoryEntrySized().

6.91.5.5 m_max_num_sharers

UInt32 Directory::m_max_num_sharers [private]

Definition at line 25 of file directory.h.

Referenced by createDirectoryEntry(), and createDirectoryEntrySized().

6.91.5.6 m_num_entries

UInt32 Directory::m_num_entries [private]

Definition at line 21 of file directory.h.

Referenced by Directory(), getDirectoryEntry(), and ∼Directory().

6.91.5.7 m_num_entries_allocated

UInt64 Directory::m_num_entries_allocated [private]

Definition at line 22 of file directory.h.

Referenced by Directory(), and getDirectoryEntry().

6.91.5.8 m_use_max_hw_sharers

UInt32 Directory::m_use_max_hw_sharers [private]

Definition at line 24 of file directory.h.

Referenced by createDirectoryEntrySized(), and getMaxHwSharers().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/directory_schemes/ **directory.h**
- common/core/memory_subsystem/directory_schemes/ **directory.cc**

6.92 DirectoryBlockInfo Class Reference

`#include <directory_block_info.h>`

Public Member Functions

- **DirectoryBlockInfo** (**DirectoryState::dstate_t** dstate= **DirectoryState::UNCACHED**)
- ∼**DirectoryBlockInfo** ()
- **DirectoryState::dstate_t getDState** ()
- void **setDState** (**DirectoryState::dstate_t** dstate)

Private Attributes

- **DirectoryState::dstate_t m_dstate**

6.92.1 Detailed Description

Definition at line 6 of file directory_block_info.h.

6.92.2 Constructor & Destructor Documentation

6.92.2.1 DirectoryBlockInfo()

```
DirectoryBlockInfo::DirectoryBlockInfo (
            DirectoryState::dstate_t dstate =  DirectoryState::UNCACHED )  [inline]
```

Definition at line 12 of file directory_block_info.h.

6.92.2.2 ∼DirectoryBlockInfo()

```
DirectoryBlockInfo::~DirectoryBlockInfo ( )  [inline]
```

Definition at line 16 of file directory_block_info.h.

6.92.3 Member Function Documentation

6.92.3.1 getDState()

```
DirectoryState::dstate_t DirectoryBlockInfo::getDState ( )  [inline]
```

Definition at line 18 of file directory_block_info.h.

References m_dstate.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDRAMReply(), PrL1PrL2Dram↩DirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectory↩Cntlr::processFlushRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFrom↩L2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processNullifyReq(), PrL1PrL2DramDirectoryMSI↩::DramDirectoryCntlr::processShReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::process↩UpgradeReqFromL2Cache(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSendToL2↩Cache().

6.92.3.2 setDState()

```
void DirectoryBlockInfo::setDState (
            DirectoryState::dstate_t dstate )  [inline]
```

Definition at line 19 of file directory_block_info.h.

References m_dstate.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2↩DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::Dram↩DirectoryCntlr::processInvRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processShReq↩FromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processUpgradeReqFromL2Cache(), PrL1↩PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbRepFromL2Cache(), and PrL1PrL2DramDirectoryMSI::↩DramDirectoryCntlr::processWbReqFromL2Cache().

6.92.4 Member Data Documentation

6.92.4.1 m_dstate

```
DirectoryState::dstate_t DirectoryBlockInfo::m_dstate  [private]
```

Definition at line 9 of file directory_block_info.h.

Referenced by getDState(), and setDState().

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/directory_schemes/ **directory_block_info.h**

6.93 DirectoryEntry Class Reference

`#include <directory_entry.h>`

Inheritance diagram for DirectoryEntry:

Public Member Functions

- **DirectoryEntry** ()
- virtual ~**DirectoryEntry** ()
- **DirectoryBlockInfo** * **getDirectoryBlockInfo** ()
- virtual bool **hasSharer** (**core_id_t** sharer_id)=0
- virtual bool **addSharer** (**core_id_t** sharer_id, **UInt32** max_hw_sharers)=0
- virtual void **removeSharer** (**core_id_t** sharer_id, bool reply_expected=false)=0
- virtual **UInt32** **getNumSharers** ()=0
- virtual **core_id_t** **getOwner** ()=0
- virtual void **setOwner** (**core_id_t** owner_id)=0
- virtual **core_id_t** **getForwarder** ()
- virtual void **setForwarder** (**core_id_t** forwarder_id)
- **IntPtr** **getAddress** ()
- void **setAddress** (**IntPtr** address)
- virtual **core_id_t** **getOneSharer** ()=0
- virtual std::pair< bool, std::vector< **core_id_t** > > **getSharersList** ()=0
- virtual **SubsecondTime** **getLatency** ()=0

Protected Attributes

- **IntPtr** **m_address**
- **DirectoryBlockInfo** **m_directory_block_info**
- **core_id_t** **m_owner_id**
- **core_id_t** **m_forwarder_id**

6.93.1 Detailed Description

Definition at line 36 of file directory_entry.h.

6.93.2 Constructor & Destructor Documentation

6.93.2.1 DirectoryEntry()

```
DirectoryEntry::DirectoryEntry ( )  [inline]
```

Definition at line 45 of file directory_entry.h.

6.93.2.2 ~DirectoryEntry()

```
virtual DirectoryEntry::~DirectoryEntry ( )  [inline], [virtual]
```

Definition at line 50 of file directory_entry.h.

6.93.3 Member Function Documentation

6.93.3.1 addSharer()

```
virtual bool DirectoryEntry::addSharer (
            core_id_t sharer_id,
            UInt32 max_hw_sharers )  [pure virtual]
```

Implemented in **DirectoryEntryLimitless**< **DirectorySharers** > (p. 443), and **DirectoryEntryLimitedNo**↩ **Broadcast**< **DirectorySharers** > (p. 439).

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2↩ DramDirectoryMSI::DramDirectoryCntlr::processShReqFromL2Cache(), and PrL1PrL2DramDirectoryMSI::Dram↩ DirectoryCntlr::processUpgradeReqFromL2Cache().

6.93.3.2 getAddress()

```
IntPtr DirectoryEntry::getAddress ( )  [inline]
```

Definition at line 73 of file directory_entry.h.

References m_address.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), and PrL1PrL2Dram↩ DirectoryMSI::DramDirectoryCache::replaceDirectoryEntry().

6.93.3.3 getDirectoryBlockInfo()

```
DirectoryBlockInfo* DirectoryEntry::getDirectoryBlockInfo ( ) [inline]
```

Definition at line 53 of file directory_entry.h.

References m_directory_block_info.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDRAMReply(), PrL1PrL2Dram↩DirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectory↩Cntlr::processFlushRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFrom↩L2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processNullifyReq(), PrL1PrL2DramDirectoryMSI↩::DramDirectoryCntlr::processShReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::process↩UpgradeReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache(), and PrL1PrL2DramDirectory↩MSI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache().

6.93.3.4 getForwarder()

```
virtual core_id_t DirectoryEntry::getForwarder ( ) [inline], [virtual]
```

Definition at line 63 of file directory_entry.h.

References m_forwarder_id.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache(), and PrL1PrL2↩DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache().

6.93.3.5 getLatency()

```
virtual SubsecondTime DirectoryEntry::getLatency ( ) [pure virtual]
```

Implemented in **DirectoryEntryLimitless**< **DirectorySharers** > (p. 443), and **DirectoryEntryLimitedNo↩Broadcast**< **DirectorySharers** > (p. 439).

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry().

6.93.3.6 getNumSharers()

```
virtual UInt32 DirectoryEntry::getNumSharers ( ) [pure virtual]
```

Implemented in **DirectoryEntrySized**< **DirectorySharers** > (p. 446).

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), and PrL1Pr↩L2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache().

6.93.3.7 getOneSharer()

```
virtual core_id_t DirectoryEntry::getOneSharer ( ) [pure virtual]
```

Implemented in **DirectoryEntryLimitless**< **DirectorySharers** > (p. 443), and **DirectoryEntryLimitedNo**↩ **Broadcast**< **DirectorySharers** > (p. 439).

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processShReqFromL2Cache().

6.93.3.8 getOwner()

```
virtual core_id_t DirectoryEntry::getOwner ( ) [pure virtual]
```

Implemented in **DirectoryEntryLimitless**< **DirectorySharers** > (p. 443), and **DirectoryEntryLimitedNo**↩ **Broadcast**< **DirectorySharers** > (p. 440).

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2↩ DramDirectoryMSI::DramDirectoryCntlr::processNullifyReq(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr↩ ::processShReqFromL2Cache(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processUpgradeReq↩ FromL2Cache().

6.93.3.9 getSharersList()

```
virtual std::pair<bool, std::vector< core_id_t> > DirectoryEntry::getSharersList ( ) [pure
virtual]
```

Implemented in **DirectoryEntrySized**< **DirectorySharers** > (p. 446).

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2Dram↩ DirectoryMSI::DramDirectoryCntlr::processNullifyReq(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr↩ ::processUpgradeReqFromL2Cache().

6.93.3.10 hasSharer()

```
virtual bool DirectoryEntry::hasSharer (
            core_id_t sharer_id ) [pure virtual]
```

Implemented in **DirectoryEntryLimitless**< **DirectorySharers** > (p. 443), and **DirectoryEntryLimitedNo**↩ **Broadcast**< **DirectorySharers** > (p. 440).

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), PrL1↩ PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::↩ DramDirectoryCntlr::processShReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::process↩ UpgradeReqFromL2Cache(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbRepFromL2↩ Cache().

6.93.3.11 removeSharer()

```
virtual void DirectoryEntry::removeSharer (
            core_id_t sharer_id,
            bool reply_expected = false )  [pure virtual]
```

Implemented in **DirectoryEntryLimitless< DirectorySharers >** (p. 444), and **DirectoryEntryLimitedNo↩ Broadcast< DirectorySharers >** (p. 440).

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), and PrL1Pr↩ L2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache().

6.93.3.12 setAddress()

```
void DirectoryEntry::setAddress (
            IntPtr address )  [inline]
```

Definition at line 74 of file directory_entry.h.

References m_address.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), and PrL1PrL2Dram↩ DirectoryMSI::DramDirectoryCache::replaceDirectoryEntry().

6.93.3.13 setForwarder()

```
virtual void DirectoryEntry::setForwarder (
            core_id_t forwarder_id )  [inline], [virtual]
```

Definition at line 68 of file directory_entry.h.

References m_forwarder_id.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), PrL1PrL2↩ DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache(), and PrL1PrL2DramDirectoryMSI::Dram↩ DirectoryCntlr::retrieveDataAndSendToL2Cache().

6.93.3.14 setOwner()

```
virtual void DirectoryEntry::setOwner (
            core_id_t owner_id )  [pure virtual]
```

Implemented in **DirectoryEntryLimitless< DirectorySharers >** (p. 444), and **DirectoryEntryLimitedNo↩ Broadcast< DirectorySharers >** (p. 440).

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1Pr↩ L2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::↩ DramDirectoryCntlr::processShReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::process↩ UpgradeReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbRepFromL2Cache(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache().

6.93.4 Member Data Documentation

6.93.4.1 m_address

```
IntPtr DirectoryEntry::m_address  [protected]
```

Definition at line 39 of file directory_entry.h.

Referenced by getAddress(), and setAddress().

6.93.4.2 m_directory_block_info

```
DirectoryBlockInfo DirectoryEntry::m_directory_block_info  [protected]
```

Definition at line 40 of file directory_entry.h.

Referenced by getDirectoryBlockInfo().

6.93.4.3 m_forwarder_id

```
core_id_t DirectoryEntry::m_forwarder_id  [protected]
```

Definition at line 42 of file directory_entry.h.

Referenced by getForwarder(), and setForwarder().

6.93.4.4 m_owner_id

```
core_id_t DirectoryEntry::m_owner_id  [protected]
```

Definition at line 41 of file directory_entry.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/directory_schemes/ **directory_entry.h**

6.94 DirectoryEntryLimitedNoBroadcast< DirectorySharers > Class Template Reference

`#include <directory_entry_limited_no_broadcast.h>`

Inheritance diagram for DirectoryEntryLimitedNoBroadcast< DirectorySharers >:

Public Member Functions

- **DirectoryEntryLimitedNoBroadcast** (**UInt32** max_hw_sharers, **UInt32** max_num_sharers)
- ~**DirectoryEntryLimitedNoBroadcast** ()
- bool **hasSharer** (**core_id_t** sharer_id)
- bool **addSharer** (**core_id_t** sharer_id, **UInt32** max_hw_sharers)
- void **removeSharer** (**core_id_t** sharer_id, bool reply_expected)
- **core_id_t** **getOwner** ()
- void **setOwner** (**core_id_t** owner_id)
- **core_id_t** **getOneSharer** ()
- **SubsecondTime** **getLatency** ()

Private Attributes

- **Random** **m_rand_num**

Additional Inherited Members

6.94.1 Detailed Description

template<class DirectorySharers>
class DirectoryEntryLimitedNoBroadcast< DirectorySharers >

Definition at line 9 of file directory_entry_limited_no_broadcast.h.

6.94.2 Constructor & Destructor Documentation

6.94.2.1 DirectoryEntryLimitedNoBroadcast()

```
template<class DirectorySharers >
DirectoryEntryLimitedNoBroadcast< DirectorySharers >::  DirectoryEntryLimitedNoBroadcast (
              UInt32 max_hw_sharers,
              UInt32 max_num_sharers )
```

Definition at line 31 of file directory_entry_limited_no_broadcast.h.

6.94.2.2 ~DirectoryEntryLimitedNoBroadcast()

```
template<class DirectorySharers >
DirectoryEntryLimitedNoBroadcast< DirectorySharers >::~ DirectoryEntryLimitedNoBroadcast
```

Definition at line 38 of file directory_entry_limited_no_broadcast.h.

6.94.3 Member Function Documentation

6.94.3.1 addSharer()

```
template<class DirectorySharers >
bool  DirectoryEntryLimitedNoBroadcast< DirectorySharers >::addSharer (
              core_id_t sharer_id,
              UInt32 max_hw_sharers )  [virtual]
```

Implements **DirectoryEntry** (p. 433).

Definition at line 53 of file directory_entry_limited_no_broadcast.h.

6.94.3.2 getLatency()

```
template<class DirectorySharers >
SubsecondTime  DirectoryEntryLimitedNoBroadcast< DirectorySharers >::getLatency  [virtual]
```

Implements **DirectoryEntry** (p. 434).

Definition at line 105 of file directory_entry_limited_no_broadcast.h.

References SubsecondTime::Zero().

6.94.3.3 getOneSharer()

```
template<class DirectorySharers >
core_id_t DirectoryEntryLimitedNoBroadcast< DirectorySharers >::getOneSharer [virtual]
```

Implements **DirectoryEntry** (p. 434).

Definition at line 93 of file directory_entry_limited_no_broadcast.h.

6.94.3.4 getOwner()

```
template<class DirectorySharers >
core_id_t DirectoryEntryLimitedNoBroadcast< DirectorySharers >::getOwner [virtual]
```

Implements **DirectoryEntry** (p. 435).

Definition at line 77 of file directory_entry_limited_no_broadcast.h.

6.94.3.5 hasSharer()

```
template<class DirectorySharers >
bool DirectoryEntryLimitedNoBroadcast< DirectorySharers >::hasSharer (
            core_id_t sharer_id ) [virtual]
```

Implements **DirectoryEntry** (p. 435).

Definition at line 43 of file directory_entry_limited_no_broadcast.h.

6.94.3.6 removeSharer()

```
template<class DirectorySharers >
void DirectoryEntryLimitedNoBroadcast< DirectorySharers >::removeSharer (
            core_id_t sharer_id,
           bool reply_expected ) [virtual]
```

Implements **DirectoryEntry** (p. 435).

Definition at line 68 of file directory_entry_limited_no_broadcast.h.

6.94.3.7 setOwner()

```
template<class DirectorySharers >
void  DirectoryEntryLimitedNoBroadcast< DirectorySharers >::setOwner (
              core_id_t owner_id )  [virtual]
```

Implements **DirectoryEntry** (p. 436).

Definition at line 84 of file directory_entry_limited_no_broadcast.h.

References INVALID_CORE_ID.

6.94.4 Member Data Documentation

6.94.4.1 m_rand_num

```
template<class DirectorySharers >
Random  DirectoryEntryLimitedNoBroadcast< DirectorySharers >::m_rand_num  [private]
```

Definition at line 27 of file directory_entry_limited_no_broadcast.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/directory_schemes/ **directory_entry_limited_no_broadcast.h**

6.95 DirectoryEntryLimitless< DirectorySharers > Class Template Reference

```
#include <directory_entry_limitless.h>
```

Inheritance diagram for DirectoryEntryLimitless< DirectorySharers >:

Public Member Functions

- **DirectoryEntryLimitless** (**UInt32** max_hw_sharers, **UInt32** max_num_sharers, **SubsecondTime** software_trap_penalty)
- ∼**DirectoryEntryLimitless** ()
- bool **hasSharer** (**core_id_t** sharer_id)
- bool **addSharer** (**core_id_t** sharer_id, **UInt32** max_hw_sharers)
- void **removeSharer** (**core_id_t** sharer_id, bool reply_expected)
- **core_id_t getOwner** ()
- void **setOwner** (**core_id_t** owner_id)
- **core_id_t getOneSharer** ()
- **SubsecondTime getLatency** ()

Private Attributes

- bool **m_software_trap_enabled**
- **SubsecondTime m_software_trap_penalty**

Additional Inherited Members

6.95.1 Detailed Description

template<**class DirectorySharers**>
class DirectoryEntryLimitless< **DirectorySharers** >

Definition at line 8 of file directory_entry_limitless.h.

6.95.2 Constructor & Destructor Documentation

6.95.2.1 DirectoryEntryLimitless()

```
template<class DirectorySharers >
DirectoryEntryLimitless< DirectorySharers >::  DirectoryEntryLimitless (
              UInt32 max_hw_sharers,
              UInt32 max_num_sharers,
              SubsecondTime software_trap_penalty )
```

Definition at line 33 of file directory_entry_limitless.h.

6.95.2.2 ∼DirectoryEntryLimitless()

```
template<class DirectorySharers >
DirectoryEntryLimitless< DirectorySharers >::~ DirectoryEntryLimitless
```

Definition at line 43 of file directory_entry_limitless.h.

6.95.3 Member Function Documentation

6.95.3.1 addSharer()

```
template<class DirectorySharers >
bool  DirectoryEntryLimitless< DirectorySharers >::addSharer (
              core_id_t sharer_id,
              UInt32 max_hw_sharers )  [virtual]
```

Implements **DirectoryEntry** (p. 433).

Definition at line 68 of file directory_entry_limitless.h.

6.95.3.2 getLatency()

```
template<class DirectorySharers >
SubsecondTime  DirectoryEntryLimitless< DirectorySharers >::getLatency  [virtual]
```

Implements **DirectoryEntry** (p. 434).

Definition at line 48 of file directory_entry_limitless.h.

References SubsecondTime::Zero().

6.95.3.3 getOneSharer()

```
template<class DirectorySharers >
core_id_t  DirectoryEntryLimitless< DirectorySharers >::getOneSharer  [virtual]
```

Implements **DirectoryEntry** (p. 434).

Definition at line 110 of file directory_entry_limitless.h.

6.95.3.4 getOwner()

```
template<class DirectorySharers >
core_id_t  DirectoryEntryLimitless< DirectorySharers >::getOwner  [virtual]
```

Implements **DirectoryEntry** (p. 435).

Definition at line 94 of file directory_entry_limitless.h.

6.95.3.5 hasSharer()

```
template<class DirectorySharers >
bool  DirectoryEntryLimitless< DirectorySharers >::hasSharer (
              core_id_t sharer_id )  [virtual]
```

Implements **DirectoryEntry** (p. 435).

Definition at line 58 of file directory_entry_limitless.h.

6.95.3.6 removeSharer()

```
template<class DirectorySharers >
void  DirectoryEntryLimitless< DirectorySharers >::removeSharer (
              core_id_t sharer_id,
             bool reply_expected )  [virtual]
```

Implements **DirectoryEntry** (p. 435).

Definition at line 84 of file directory_entry_limitless.h.

6.95.3.7 setOwner()

```
template<class DirectorySharers >
void  DirectoryEntryLimitless< DirectorySharers >::setOwner (
              core_id_t owner_id )  [virtual]
```

Implements **DirectoryEntry** (p. 436).

Definition at line 101 of file directory_entry_limitless.h.

References INVALID_CORE_ID.

6.95.4 Member Data Documentation

6.95.4.1 m_software_trap_enabled

```
template<class DirectorySharers >
bool  DirectoryEntryLimitless< DirectorySharers >::m_software_trap_enabled  [private]
```

Definition at line 11 of file directory_entry_limitless.h.

6.95.4.2 m_software_trap_penalty

```
template<class DirectorySharers >
SubsecondTime DirectoryEntryLimitless< DirectorySharers >::m_software_trap_penalty [private]
```

Definition at line 12 of file directory_entry_limitless.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/directory_schemes/ **directory_entry_limitless.h**

6.96 DirectoryEntrySized< DirectorySharers > Class Template Reference

```
#include <directory_entry.h>
```

Inheritance diagram for DirectoryEntrySized< DirectorySharers >:

Public Member Functions

- **DirectoryEntrySized** (**UInt32** max_hw_sharers, **UInt32** max_num_sharers)
- virtual **UInt32** **getNumSharers** ()
- virtual std::pair< bool, std::vector< **core_id_t** > > **getSharersList** ()

Protected Attributes

- DirectorySharers **m_sharers**

6.96.1 Detailed Description

template<class DirectorySharers>
class DirectoryEntrySized< DirectorySharers >

Definition at line 83 of file directory_entry.h.

6.96.2 Constructor & Destructor Documentation

6.96.2.1 DirectoryEntrySized()

```
template<class DirectorySharers >
DirectoryEntrySized< DirectorySharers >::  DirectoryEntrySized (
             UInt32 max_hw_sharers,
             UInt32 max_num_sharers )  [inline]
```

Definition at line 89 of file directory_entry.h.

6.96.3 Member Function Documentation

6.96.3.1 getNumSharers()

```
template<class DirectorySharers >
virtual  UInt32  DirectoryEntrySized< DirectorySharers >::getNumSharers ( )  [inline], [virtual]
```

Implements **DirectoryEntry** (p. 434).

Definition at line 95 of file directory_entry.h.

References DirectoryEntrySized< DirectorySharers >::m_sharers.

Referenced by DirectoryEntrySized< DirectorySharers >::getSharersList().

6.96.3.2 getSharersList()

```
template<class DirectorySharers >
virtual std::pair<bool, std::vector< core_id_t> >  DirectoryEntrySized< DirectorySharers >↩
::getSharersList ( )  [inline], [virtual]
```

Implements **DirectoryEntry** (p. 435).

Definition at line 96 of file directory_entry.h.

References DirectoryEntrySized< DirectorySharers >::getNumSharers(), and DirectoryEntrySized< Directory↩ Sharers >::m_sharers.

6.96.4 Member Data Documentation

6.96.4.1 m_sharers

```
template<class DirectorySharers >
DirectorySharers DirectoryEntrySized< DirectorySharers >::m_sharers [protected]
```

Definition at line 86 of file directory_entry.h.

Referenced by DirectoryEntrySized< DirectorySharers >::getNumSharers(), and DirectoryEntrySized< DirectorySharers >::getSharersList().

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/directory_schemes/ **directory_entry.h**

6.97 DirectorySharersBitset< Size > Class Template Reference

```
#include <directory_entry.h>
```

Inheritance diagram for DirectorySharersBitset< Size >:

Public Member Functions

- **DirectorySharersBitset** (**UInt32** max_num_sharers)

6.97.1 Detailed Description

template<long Size>
class DirectorySharersBitset< Size >

Definition at line 13 of file directory_entry.h.

6.97.2 Constructor & Destructor Documentation

6.97.2.1 DirectorySharersBitset()

```
template<long Size>
DirectorySharersBitset< Size >::  DirectorySharersBitset (
              UInt32 max_num_sharers )  [inline]
```

Definition at line 16 of file directory_entry.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/directory_schemes/ **directory_entry.h**

6.98 DirectorySharersVector Class Reference

```
#include <directory_entry.h>
```

Inheritance diagram for DirectorySharersVector:

Public Member Functions

- **DirectorySharersVector** (**UInt32** max_num_sharers)
- **UInt32 count** (void) const

6.98.1 Detailed Description

Definition at line 20 of file directory_entry.h.

6.98.2 Constructor & Destructor Documentation

6.98.2.1 DirectorySharersVector()

```
DirectorySharersVector::DirectorySharersVector (
              UInt32 max_num_sharers )  [inline]
```

Definition at line 23 of file directory_entry.h.

6.98.3 Member Function Documentation

6.98.3.1 count()

```
UInt32 DirectorySharersVector::count (
            void  ) const  [inline]
```

Definition at line 24 of file directory_entry.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/directory_schemes/ **directory_entry.h**

6.99 DirectoryState Class Reference

`#include <directory_state.h>`

Public Types

- enum **dstate_t** {
  UNCACHED = 0, **SHARED**, **OWNED**, **MODIFIED**,
  EXCLUSIVE, **NUM_DIRECTORY_STATES** }

6.99.1 Detailed Description

Definition at line 4 of file directory_state.h.

6.99.2 Member Enumeration Documentation

6.99.2.1 dstate_t

enum **DirectoryState::dstate_t**

Enumerator

UNCACHED	
SHARED	
OWNED	
MODIFIED	
EXCLUSIVE	
NUM_DIRECTORY_STATES	

Definition at line 7 of file directory_state.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/directory_schemes/ **directory_state.h**

6.100 FastNehalem::Dram Class Reference

`#include <fast_cache.h>`

Inheritance diagram for FastNehalem::Dram:

Public Member Functions

- **Dram** (**Core** ∗core, String name, **UInt64** latency)
- **SubsecondTime access** (**Core::mem_op_t** mem_op_type, **IntPtr** tag)

Private Attributes

- const **ComponentLatency m_latency**
- **UInt64 m_reads**
- **UInt64 m_writes**
- **SubsecondTime m_total_latency**

6.100.1 Detailed Description

Definition at line 8 of file fast_cache.h.

6.100.2 Constructor & Destructor Documentation

6.100.2.1 Dram()

```
FastNehalem::Dram::Dram (
              Core * core,
             String name,
              UInt64 latency )  [inline]
```

Definition at line 15 of file fast_cache.h.

References Core::getId(), m_reads, m_total_latency, m_writes, registerStatsMetric(), and SubsecondTime::Zero().

6.100.3 Member Function Documentation

6.100.3.1 access()

```
SubsecondTime FastNehalem::Dram::access (
            Core::mem_op_t mem_op_type,
            IntPtr tag )  [inline], [virtual]
```

Implements **FastNehalem::CacheBase** (p. 150).

Definition at line 24 of file fast_cache.h.

References ComponentLatency::getLatency(), m_latency, m_reads, m_total_latency, m_writes, and Core::WRITE.

6.100.4 Member Data Documentation

6.100.4.1 m_latency

```
const  ComponentLatency FastNehalem::Dram::m_latency  [private]
```

Definition at line 11 of file fast_cache.h.

Referenced by access().

6.100.4.2 m_reads

```
UInt64 FastNehalem::Dram::m_reads  [private]
```

Definition at line 12 of file fast_cache.h.

Referenced by access(), and Dram().

6.100.4.3 m_total_latency

```
SubsecondTime FastNehalem::Dram::m_total_latency  [private]
```

Definition at line 13 of file fast_cache.h.

Referenced by access(), and Dram().

6.100.4.4 m_writes

UInt64 FastNehalem::Dram::m_writes [private]

Definition at line 12 of file fast_cache.h.

Referenced by access(), and Dram().

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/fast_nehalem/ **fast_cache.h**

6.101 DramCache Class Reference

#include <dram_cache.h>

Inheritance diagram for DramCache:

Public Member Functions

- **DramCache** (**MemoryManagerBase** *memory_manager, **ShmemPerfModel** *shmem_perf_model, **AddressHomeLookup** *home_lookup, **UInt32** cache_block_size, **DramCntlrInterface** *dram_cntlr)
- ~**DramCache** ()
- virtual boost::tuple< **SubsecondTime**, **HitWhere::where_t** > **getDataFromDram** (**IntPtr** address, **core_id_t** requester, **Byte** *data_buf, **SubsecondTime** now, **ShmemPerf** *perf)
- virtual boost::tuple< **SubsecondTime**, **HitWhere::where_t** > **putDataToDram** (**IntPtr** address, **core_id_t** requester, **Byte** *data_buf, **SubsecondTime** now)

Private Member Functions

- std::pair< bool, **SubsecondTime** > **doAccess** (**Cache::access_t** access, **IntPtr** address, **core_id_t** requester, **Byte** *data_buf, **SubsecondTime** now, **ShmemPerf** *perf)
- void **insertLine** (**Cache::access_t** access, **IntPtr** address, **core_id_t** requester, **Byte** *data_buf, **SubsecondTime** now)
- **SubsecondTime** **accessDataArray** (**Cache::access_t** access, **core_id_t** requester, **SubsecondTime** **t_start**, **ShmemPerf** *perf)
- void **callPrefetcher** (**IntPtr** address, bool cache_hit, bool prefetch_hit, **SubsecondTime** t_issue)

Private Attributes

- **core_id_t m_core_id**
- **UInt32 m_cache_block_size**
- **SubsecondTime m_data_access_time**
- **SubsecondTime m_tags_access_time**
- **ComponentBandwidth m_data_array_bandwidth**
- **AddressHomeLookup * m_home_lookup**
- **DramCntlrInterface * m_dram_cntlr**
- **Cache * m_cache**
- **QueueModel * m_queue_model**
- **Prefetcher * m_prefetcher**
- bool **m_prefetch_on_prefetch_hit**
- **ContentionModel m_prefetch_mshr**
- **UInt64 m_reads**
- **UInt64 m_writes**
- **UInt64 m_read_misses**
- **UInt64 m_write_misses**
- **UInt64 m_hits_prefetch**
- **UInt64 m_prefetches**
- **SubsecondTime m_prefetch_mshr_delay**

Additional Inherited Members

6.101.1 Detailed Description

Definition at line 12 of file dram_cache.h.

6.101.2 Constructor & Destructor Documentation

6.101.2.1 DramCache()

```
DramCache::DramCache (
                MemoryManagerBase * memory_manager,
                ShmemPerfModel * shmem_perf_model,
                AddressHomeLookup * home_lookup,
                UInt32 cache_block_size,
                DramCntlrInterface * dram_cntlr )
```

Definition at line 12 of file dram_cache.cc.

References QueueModel::create(), Prefetcher::createPrefetcher(), ComponentBandwidth::getRoundedLatency(), k_KILO, LOG_ASSERT_ERROR, m_cache, m_cache_block_size, m_core_id, m_data_array_bandwidth, m_hits↩_prefetch, m_prefetch_mshr_delay, m_prefetch_on_prefetch_hit, m_prefetcher, m_prefetches, m_queue_model, m_read_misses, m_reads, m_write_misses, m_writes, CacheBase::parseAddressHash(), CacheBase::PR_L1_↩CACHE, and registerStatsMetric().

6.101.2.2 ~DramCache()

```
DramCache::~DramCache ( )
```

Definition at line 68 of file dram_cache.cc.

References m_cache, and m_queue_model.

6.101.3 Member Function Documentation

6.101.3.1 accessDataArray()

```
SubsecondTime DramCache::accessDataArray (
            Cache::access_t access,
            core_id_t requester,
            SubsecondTime t_start,
            ShmemPerf * perf ) [private]
```

Definition at line 180 of file dram_cache.cc.

References QueueModel::computeQueueDelay(), ShmemPerf::DRAM_CACHE_BUS, ShmemPerf::DRAM_CACHE_DATA, ShmemPerf::DRAM_CACHE_QUEUE, ComponentBandwidth::getRoundedLatency(), m_cache_block_size, m_data_access_time, m_data_array_bandwidth, m_queue_model, t_start, ShmemPerf::updateTime(), and SubsecondTime::Zero().

Referenced by doAccess(), and insertLine().

6.101.3.2 callPrefetcher()

```
void DramCache::callPrefetcher (
            IntPtr address,
            bool cache_hit,
            bool prefetch_hit,
            SubsecondTime t_issue ) [private]
```

Definition at line 204 of file dram_cache.cc.

References ContentionModel::getCompletionTime(), DramCntlrInterface::getDataFromDram(), Prefetcher::getNextAddress(), insertLine(), INVALID_CORE_ID, CacheBase::LOAD, m_cache, m_cache_block_size, m_core_id, m_dram_cntlr, m_prefetch_mshr, m_prefetch_on_prefetch_hit, m_prefetcher, m_prefetches, Cache::peekSingleLine(), CacheBlockInfo::PREFETCH, and CacheBlockInfo::setOption().

Referenced by doAccess().

6.101.3.3 doAccess()

```
std::pair< bool, SubsecondTime > DramCache::doAccess (
              Cache::access_t access,
              IntPtr address,
              core_id_t requester,
              Byte * data_buf,
              SubsecondTime now,
              ShmemPerf * perf )  [private]
```

Definition at line 101 of file dram_cache.cc.

References accessDataArray(), Cache::accessSingleLine(), callPrefetcher(), CacheBlockInfo::clearOption(), ShmemPerf::DRAM_CACHE_TAGS, DramCntlrInterface::getDataFromDram(), ContentionModel::getTag↩CompletionTime(), CacheBlockInfo::hasOption(), insertLine(), CacheBase::LOAD, m_cache, m_cache_block_size, m_dram_cntlr, m_hits_prefetch, m_prefetch_mshr, m_prefetch_mshr_delay, m_prefetcher, m_tags_access_time, SubsecondTime::MaxTime(), CacheState::MODIFIED, Cache::peekSingleLine(), CacheBlockInfo::PREFETCH, CacheBlockInfo::setCState(), CacheBase::STORE, and ShmemPerf::updateTime().

Referenced by getDataFromDram(), and putDataToDram().

6.101.3.4 getDataFromDram()

```
boost::tuple< SubsecondTime, HitWhere::where_t > DramCache::getDataFromDram (
              IntPtr address,
              core_id_t requester,
              Byte * data_buf,
              SubsecondTime now,
              ShmemPerf * perf )  [virtual]
```

Implements **DramCntlrInterface** (p. 467).

Definition at line 76 of file dram_cache.cc.

References doAccess(), HitWhere::DRAM, HitWhere::DRAM_CACHE, CacheBase::LOAD, m_read_misses, and m_reads.

6.101.3.5 insertLine()

```
void DramCache::insertLine (
              Cache::access_t access,
              IntPtr address,
              core_id_t requester,
              Byte * data_buf,
              SubsecondTime now )  [private]
```

Definition at line 157 of file dram_cache.cc.

References accessDataArray(), CacheBlockInfo::getCState(), Cache::insertSingleLine(), m_cache, m_cache↩_block_size, m_dram_cntlr, CacheState::MODIFIED, Cache::peekSingleLine(), DramCntlrInterface::putDataTo↩Dram(), CacheBlockInfo::setCState(), CacheState::SHARED, and CacheBase::STORE.

Referenced by callPrefetcher(), and doAccess().

6.101.3.6 putDataToDram()

```
boost::tuple< SubsecondTime, HitWhere::where_t > DramCache::putDataToDram (
              IntPtr address,
              core_id_t requester,
              Byte * data_buf,
              SubsecondTime now ) [virtual]
```

Implements **DramCntlrInterface** (p. 468).

Definition at line 88 of file dram_cache.cc.

References doAccess(), HitWhere::DRAM, HitWhere::DRAM_CACHE, m_write_misses, m_writes, and Cache↩ Base::STORE.

6.101.4 Member Data Documentation

6.101.4.1 m_cache

```
Cache* DramCache::m_cache [private]
```

Definition at line 30 of file dram_cache.h.

Referenced by callPrefetcher(), doAccess(), DramCache(), insertLine(), and ∼DramCache().

6.101.4.2 m_cache_block_size

```
UInt32 DramCache::m_cache_block_size [private]
```

Definition at line 23 of file dram_cache.h.

Referenced by accessDataArray(), callPrefetcher(), doAccess(), DramCache(), and insertLine().

6.101.4.3 m_core_id

```
core_id_t DramCache::m_core_id [private]
```

Definition at line 22 of file dram_cache.h.

Referenced by callPrefetcher(), and DramCache().

6.101.4.4 m_data_access_time

SubsecondTime DramCache::m_data_access_time [private]

Definition at line 24 of file dram_cache.h.

Referenced by accessDataArray().

6.101.4.5 m_data_array_bandwidth

ComponentBandwidth DramCache::m_data_array_bandwidth [private]

Definition at line 26 of file dram_cache.h.

Referenced by accessDataArray(), and DramCache().

6.101.4.6 m_dram_cntlr

DramCntlrInterface* DramCache::m_dram_cntlr [private]

Definition at line 29 of file dram_cache.h.

Referenced by callPrefetcher(), doAccess(), and insertLine().

6.101.4.7 m_hits_prefetch

UInt64 DramCache::m_hits_prefetch [private]

Definition at line 38 of file dram_cache.h.

Referenced by doAccess(), and DramCache().

6.101.4.8 m_home_lookup

AddressHomeLookup* DramCache::m_home_lookup [private]

Definition at line 28 of file dram_cache.h.

6.101.4.9 m_prefetch_mshr

ContentionModel DramCache::m_prefetch_mshr [private]

Definition at line 34 of file dram_cache.h.

Referenced by callPrefetcher(), and doAccess().

6.101.4.10 m_prefetch_mshr_delay

SubsecondTime DramCache::m_prefetch_mshr_delay [private]

Definition at line 39 of file dram_cache.h.

Referenced by doAccess(), and DramCache().

6.101.4.11 m_prefetch_on_prefetch_hit

bool DramCache::m_prefetch_on_prefetch_hit [private]

Definition at line 33 of file dram_cache.h.

Referenced by callPrefetcher(), and DramCache().

6.101.4.12 m_prefetcher

Prefetcher* DramCache::m_prefetcher [private]

Definition at line 32 of file dram_cache.h.

Referenced by callPrefetcher(), doAccess(), and DramCache().

6.101.4.13 m_prefetches

UInt64 DramCache::m_prefetches [private]

Definition at line 38 of file dram_cache.h.

Referenced by callPrefetcher(), and DramCache().

6.101.4.14 m_queue_model

```
QueueModel* DramCache::m_queue_model  [private]
```

Definition at line 31 of file dram_cache.h.

Referenced by accessDataArray(), DramCache(), and ~DramCache().

6.101.4.15 m_read_misses

```
UInt64 DramCache::m_read_misses  [private]
```

Definition at line 37 of file dram_cache.h.

Referenced by DramCache(), and getDataFromDram().

6.101.4.16 m_reads

```
UInt64 DramCache::m_reads  [private]
```

Definition at line 36 of file dram_cache.h.

Referenced by DramCache(), and getDataFromDram().

6.101.4.17 m_tags_access_time

```
SubsecondTime DramCache::m_tags_access_time  [private]
```

Definition at line 25 of file dram_cache.h.

Referenced by doAccess().

6.101.4.18 m_write_misses

```
UInt64 DramCache::m_write_misses  [private]
```

Definition at line 37 of file dram_cache.h.

Referenced by DramCache(), and putDataToDram().

6.101.4.19 m_writes

UInt64 DramCache::m_writes [private]

Definition at line 36 of file dram_cache.h.

Referenced by DramCache(), and putDataToDram().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/dram/ **dram_cache.h**
- common/core/memory_subsystem/dram/ **dram_cache.cc**

6.102 PrL1PrL2DramDirectoryMSI::DramCntlr Class Reference

#include <dram_cntlr.h>

Inheritance diagram for PrL1PrL2DramDirectoryMSI::DramCntlr:

Public Member Functions

- **DramCntlr** (**MemoryManagerBase** *memory_manager, **ShmemPerfModel** *shmem_perf_model, **UInt32** cache_block_size)
- ~**DramCntlr** ()
- **DramPerfModel** * **getDramPerfModel** ()
- boost::tuple< **SubsecondTime**, **HitWhere::where_t** > **getDataFromDram** (**IntPtr** address, **core_id_t** requester, **Byte** *data_buf, **SubsecondTime** now, **ShmemPerf** *perf)
- boost::tuple< **SubsecondTime**, **HitWhere::where_t** > **putDataToDram** (**IntPtr** address, **core_id_t** requester, **Byte** *data_buf, **SubsecondTime** now)

Private Types

- typedef std::unordered_map< **IntPtr**, **UInt64** > **AccessCountMap**

Private Member Functions

- **SubsecondTime** **runDramPerfModel** (**core_id_t** requester, **SubsecondTime** time, **IntPtr** address, **DramCntlrInterface::access_t** access_type, **ShmemPerf** *perf)
- void **addToDramAccessCount** (**IntPtr** address, **access_t** access_type)
- void **printDramAccessCount** (void)

Private Attributes

- std::unordered_map< **IntPtr**, **Byte** * > **m_data_map**
- **DramPerfModel** * **m_dram_perf_model**
- **FaultInjector** * **m_fault_injector**
- **AccessCountMap** * **m_dram_access_count**
- **UInt64** **m_reads**
- **UInt64** **m_writes**
- **ShmemPerf** **m_dummy_shmem_perf**

Additional Inherited Members

6.102.1 Detailed Description

Definition at line 20 of file dram_cntlr.h.

6.102.2 Member Typedef Documentation

6.102.2.1 AccessCountMap

```
typedef std::unordered_map< IntPtr, UInt64 >  PrL1PrL2DramDirectoryMSI::DramCntlr::Access↩
CountMap  [private]
```

Definition at line 27 of file dram_cntlr.h.

6.102.3 Constructor & Destructor Documentation

6.102.3.1 DramCntlr()

```
PrL1PrL2DramDirectoryMSI::DramCntlr::DramCntlr (
            MemoryManagerBase * memory_manager,
            ShmemPerfModel * shmem_perf_model,
            UInt32 cache_block_size )
```

Definition at line 24 of file dram_cntlr.cc.

References DramPerfModel::createDramPerfModel(), MemComponent::DRAM, MemoryManagerBase::getCore(), Core::getId(), m_dram_access_count, m_dram_perf_model, m_fault_injector, m_reads, m_writes, DramCntlr↩Interface::NUM_ACCESS_TYPES, and registerStatsMetric().

6.102.3.2 ~DramCntlr()

```
PrL1PrL2DramDirectoryMSI::DramCntlr::~DramCntlr ( )
```

Definition at line 44 of file dram_cntlr.cc.

References m_dram_access_count, m_dram_perf_model, and printDramAccessCount().

6.102.4 Member Function Documentation

6.102.4.1 addToDramAccessCount()

```
void PrL1PrL2DramDirectoryMSI::DramCntlr::addToDramAccessCount (
              IntPtr address,
              DramCntlrInterface::access_t access_type )  [private]
```

Definition at line 119 of file dram_cntlr.cc.

References m_dram_access_count.

Referenced by getDataFromDram(), and putDataToDram().

6.102.4.2 getDataFromDram()

```
boost::tuple< SubsecondTime, HitWhere::where_t > PrL1PrL2DramDirectoryMSI::DramCntlr::get↩
DataFromDram (
              IntPtr address,
              core_id_t requester,
              Byte * data_buf,
              SubsecondTime now,
              ShmemPerf * perf )  [virtual]
```

Implements **DramCntlrInterface** (p. 467).

Definition at line 53 of file dram_cntlr.cc.

References addToDramAccessCount(), HitWhere::DRAM, DramCntlrInterface::getCacheBlockSize(), itostr(), m↩
_data_map, m_fault_injector, m_reads, MYLOG, FaultInjector::preRead(), DramCntlrInterface::READ, and run↩
DramPerfModel().

6.102.4.3 getDramPerfModel()

```
DramPerfModel* PrL1PrL2DramDirectoryMSI::DramCntlr::getDramPerfModel ( )  [inline]
```

Definition at line 45 of file dram_cntlr.h.

References m_dram_perf_model.

Referenced by ParametricDramDirectoryMSI::MemoryManager::disableModels(), and ParametricDramDirectory↩ MSI::MemoryManager::enableModels().

6.102.4.4 printDramAccessCount()

```
void PrL1PrL2DramDirectoryMSI::DramCntlr::printDramAccessCount (
            void  )  [private]
```

Definition at line 125 of file dram_cntlr.cc.

References MemoryManagerBase::getCore(), Core::getId(), LOG_PRINT, m_dram_access_count, DramCntlr↩ Interface::m_memory_manager, DramCntlrInterface::NUM_ACCESS_TYPES, and DramCntlrInterface::READ.

Referenced by ∼DramCntlr().

6.102.4.5 putDataToDram()

```
boost::tuple< SubsecondTime, HitWhere::where_t > PrL1PrL2DramDirectoryMSI::DramCntlr::put↩
DataToDram (
              IntPtr address,
              core_id_t requester,
              Byte * data_buf,
              SubsecondTime now )  [virtual]
```

Implements **DramCntlrInterface** (p. 468).

Definition at line 82 of file dram_cntlr.cc.

References addToDramAccessCount(), HitWhere::DRAM, DramCntlrInterface::getCacheBlockSize(), LOG_PRIN↩ T_ERROR, m_data_map, m_dummy_shmem_perf, m_fault_injector, m_writes, MYLOG, FaultInjector::postWrite(), runDramPerfModel(), and DramCntlrInterface::WRITE.

6.102.4.6 runDramPerfModel()

```
SubsecondTime PrL1PrL2DramDirectoryMSI::DramCntlr::runDramPerfModel (
            core_id_t requester,
            SubsecondTime time,
            IntPtr address,
            DramCntlrInterface::access_t access_type,
            ShmemPerf * perf )  [private]
```

Definition at line 111 of file dram_cntlr.cc.

References DramPerfModel::getAccessLatency(), DramCntlrInterface::getCacheBlockSize(), and m_dram_perf_↩ model.

Referenced by getDataFromDram(), and putDataToDram().

6.102.5 Member Data Documentation

6.102.5.1 m_data_map

```
std::unordered_map< IntPtr,  Byte*> PrL1PrL2DramDirectoryMSI::DramCntlr::m_data_map  [private]
```

Definition at line 23 of file dram_cntlr.h.

Referenced by getDataFromDram(), and putDataToDram().

6.102.5.2 m_dram_access_count

```
AccessCountMap* PrL1PrL2DramDirectoryMSI::DramCntlr::m_dram_access_count  [private]
```

Definition at line 28 of file dram_cntlr.h.

Referenced by addToDramAccessCount(), DramCntlr(), printDramAccessCount(), and ~DramCntlr().

6.102.5.3 m_dram_perf_model

```
DramPerfModel* PrL1PrL2DramDirectoryMSI::DramCntlr::m_dram_perf_model  [private]
```

Definition at line 24 of file dram_cntlr.h.

Referenced by DramCntlr(), getDramPerfModel(), runDramPerfModel(), and ~DramCntlr().

6.102.5.4 m_dummy_shmem_perf

ShmemPerf PrL1PrL2DramDirectoryMSI::DramCntlr::m_dummy_shmem_perf [private]

Definition at line 31 of file dram_cntlr.h.

Referenced by putDataToDram().

6.102.5.5 m_fault_injector

FaultInjector* PrL1PrL2DramDirectoryMSI::DramCntlr::m_fault_injector [private]

Definition at line 25 of file dram_cntlr.h.

Referenced by DramCntlr(), getDataFromDram(), and putDataToDram().

6.102.5.6 m_reads

UInt64 PrL1PrL2DramDirectoryMSI::DramCntlr::m_reads [private]

Definition at line 29 of file dram_cntlr.h.

Referenced by DramCntlr(), and getDataFromDram().

6.102.5.7 m_writes

UInt64 PrL1PrL2DramDirectoryMSI::DramCntlr::m_writes [private]

Definition at line 29 of file dram_cntlr.h.

Referenced by DramCntlr(), and putDataToDram().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **dram_cntlr.h**
- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **dram_cntlr.cc**

6.103 DramCntlrInterface Class Reference

`#include <dram_cntlr_interface.h>`

Inheritance diagram for DramCntlrInterface:

Public Types

- enum **access_t** { **READ** = 0, **WRITE**, **NUM_ACCESS_TYPES** }

Public Member Functions

- **DramCntlrInterface** (**MemoryManagerBase** *memory_manager, **ShmemPerfModel** *shmem_perf_↩ model, **UInt32** cache_block_size)
- virtual ~**DramCntlrInterface** ()
- virtual boost::tuple< **SubsecondTime**, **HitWhere::where_t** > **getDataFromDram** (**IntPtr** address, **core_id_t** requester, **Byte** *data_buf, **SubsecondTime** now, **ShmemPerf** *perf)=0
- virtual boost::tuple< **SubsecondTime**, **HitWhere::where_t** > **putDataToDram** (**IntPtr** address, **core_↩ id_t** requester, **Byte** *data_buf, **SubsecondTime** now)=0
- void **handleMsgFromTagDirectory** (**core_id_t** sender, **PrL1PrL2DramDirectoryMSI::ShmemMsg** *shmem_msg)

Protected Member Functions

- **UInt32 getCacheBlockSize** ()
- **MemoryManagerBase** * **getMemoryManager** ()
- **ShmemPerfModel** * **getShmemPerfModel** ()

Protected Attributes

- **MemoryManagerBase** * **m_memory_manager**
- **ShmemPerfModel** * **m_shmem_perf_model**
- **UInt32 m_cache_block_size**

6.103.1 Detailed Description

Definition at line 15 of file dram_cntlr_interface.h.

6.103.2 Member Enumeration Documentation

6.103.2.1 access_t

enum **DramCntlrInterface::access_t**

Enumerator

READ	
WRITE	
NUM_ACCESS_TYPES	

Definition at line 27 of file dram_cntlr_interface.h.

6.103.3 Constructor & Destructor Documentation

6.103.3.1 DramCntlrInterface()

```
DramCntlrInterface::DramCntlrInterface (
              MemoryManagerBase * memory_manager,
              ShmemPerfModel * shmem_perf_model,
              UInt32 cache_block_size )  [inline]
```

Definition at line 34 of file dram_cntlr_interface.h.

6.103.3.2 ~DramCntlrInterface()

```
virtual DramCntlrInterface::~DramCntlrInterface ( )  [inline], [virtual]
```

Definition at line 39 of file dram_cntlr_interface.h.

6.103.4 Member Function Documentation

6.103.4.1 getCacheBlockSize()

```
UInt32 DramCntlrInterface::getCacheBlockSize ( )  [inline], [protected]
```

Definition at line 22 of file dram_cntlr_interface.h.

References m_cache_block_size.

Referenced by PrL1PrL2DramDirectoryMSI::DramCntlr::getDataFromDram(), handleMsgFromTagDirectory(), PrL1PrL2DramDirectoryMSI::DramCntlr::putDataToDram(), and PrL1PrL2DramDirectoryMSI::DramCntlr::runDramPerfModel().

6.103.4.2 getDataFromDram()

```
virtual boost::tuple< SubsecondTime, HitWhere::where_t> DramCntlrInterface::getDataFromDram (
              IntPtr address,
              core_id_t requester,
              Byte * data_buf,
              SubsecondTime now,
              ShmemPerf * perf )  [pure virtual]
```

Implemented in **PrL1PrL2DramDirectoryMSI::DramCntlr** (p. 462), and **DramCache** (p. 455).

Referenced by ParametricDramDirectoryMSI::CacheCntlr::accessDRAM(), DramCache::callPrefetcher(), DramCache::doAccess(), and handleMsgFromTagDirectory().

6.103.4.3 getMemoryManager()

```
MemoryManagerBase* DramCntlrInterface::getMemoryManager ( )  [inline], [protected]
```

Definition at line 23 of file dram_cntlr_interface.h.

References m_memory_manager.

Referenced by handleMsgFromTagDirectory().

6.103.4.4 getShmemPerfModel()

```
ShmemPerfModel* DramCntlrInterface::getShmemPerfModel ( )  [inline], [protected]
```

Definition at line 24 of file dram_cntlr_interface.h.

References m_shmem_perf_model.

Referenced by handleMsgFromTagDirectory().

6.103.4.5 handleMsgFromTagDirectory()

```
void DramCntlrInterface::handleMsgFromTagDirectory (
              core_id_t sender,
              PrL1PrL2DramDirectoryMSI::ShmemMsg * shmem_msg )
```

Definition at line 7 of file dram_cntlr_interface.cc.

References ShmemPerfModel::_SIM_THREAD, MemComponent::DRAM, ShmemPerf::DRAM, HitWhere::DRAM_CACHE, ShmemPerf::DRAM_CACHE, PrL1PrL2DramDirectoryMSI::ShmemMsg::DRAM_READ_REP, PrL1PrL2DramDirectoryMSI::ShmemMsg::DRAM_READ_REQ, PrL1PrL2DramDirectoryMSI::ShmemMsg::DRAM_WRITE_REQ, PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), getCacheBlockSize(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataBuf(), getDataFromDram(), ShmemPerfModel::getElapsedTime(), getMemoryManager(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), getShmemPerfModel(), ShmemPerfModel::incrElapsedTime(), LOG_PRINT_ERROR, putDataToDram(), MemoryManagerBase::sendMsg(), MemComponent::TAG_DIR, and ShmemPerf::updateTime().

Referenced by ParametricDramDirectoryMSI::CacheCntlr::flushBlock(), and ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork().

6.103.4.6 putDataToDram()

```
virtual boost::tuple< SubsecondTime, HitWhere::where_t> DramCntlrInterface::putDataToDram (
              IntPtr address,
              core_id_t requester,
              Byte * data_buf,
              SubsecondTime now )  [pure virtual]
```

Implemented in **PrL1PrL2DramDirectoryMSI::DramCntlr** (p. 463), and **DramCache** (p. 455).

Referenced by ParametricDramDirectoryMSI::CacheCntlr::accessDRAM(), handleMsgFromTagDirectory(), and DramCache::insertLine().

6.103.5 Member Data Documentation

6.103.5.1 m_cache_block_size

```
UInt32 DramCntlrInterface::m_cache_block_size  [protected]
```

Definition at line 20 of file dram_cntlr_interface.h.

Referenced by getCacheBlockSize().

6.103.5.2 m_memory_manager

```
MemoryManagerBase* DramCntlrInterface::m_memory_manager  [protected]
```

Definition at line 18 of file dram_cntlr_interface.h.

Referenced by getMemoryManager(), and PrL1PrL2DramDirectoryMSI::DramCntlr::printDramAccessCount().

6.103.5.3 m_shmem_perf_model

```
ShmemPerfModel* DramCntlrInterface::m_shmem_perf_model  [protected]
```

Definition at line 19 of file dram_cntlr_interface.h.

Referenced by getShmemPerfModel().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/dram/ **dram_cntlr_interface.h**
- common/core/memory_subsystem/dram/ **dram_cntlr_interface.cc**

6.104 PrL1PrL2DramDirectoryMSI::DramDirectoryCache Class Reference

`#include <dram_directory_cache.h>`

Public Member Functions

- **DramDirectoryCache** (**core_id_t** core_id, String directory_type_str, **UInt32** total_entries, **UInt32** associativity, **UInt32** cache_block_size, **UInt32** max_hw_sharers, **UInt32** max_num_sharers, **ComponentLatency** dram_directory_cache_access_time, **ShmemPerfModel** *shmem_perf_model)
- ~**DramDirectoryCache** ()
- **DirectoryEntry** * **getDirectoryEntry** (**IntPtr** address, bool modeled=false)
- **DirectoryEntry** * **replaceDirectoryEntry** (**IntPtr** replaced_address, **IntPtr** address, bool modeled)
- void **invalidateDirectoryEntry** (**IntPtr** address)
- void **getReplacementCandidates** (**IntPtr** address, std::vector< **DirectoryEntry** * > &replacement_candidate_list)
- **UInt32** **getMaxHwSharers** () const

Private Member Functions

- **ShmemPerfModel** * **getShmemPerfModel** ()
- void **splitAddress** (**IntPtr** address, **IntPtr** &tag, **UInt32** &set_index)
- **UInt32** **getCacheBlockSize** ()
- **UInt32** **getLogCacheBlockSize** ()
- **UInt32** **getNumSets** ()
- **UInt32** **getLogNumSets** ()

Private Attributes

- **Directory** * **m_directory**
- **UInt32** * **m_replacement_ptrs**
- std::vector< **DirectoryEntry** * > **m_replaced_directory_entry_list**
- **UInt32** **m_total_entries**
- **UInt32** **m_associativity**
- **UInt32** **m_num_sets**
- **UInt32** **m_cache_block_size**
- **UInt32** **m_log_num_sets**
- **UInt32** **m_log_cache_block_size**
- **ComponentLatency** **m_dram_directory_cache_access_time**
- **ShmemPerfModel** * **m_shmem_perf_model**

6.104.1 Detailed Description

Definition at line 11 of file dram_directory_cache.h.

6.104.2 Constructor & Destructor Documentation

6.104.2.1 DramDirectoryCache()

```
PrL1PrL2DramDirectoryMSI::DramDirectoryCache::DramDirectoryCache (
              core_id_t core_id,
             String directory_type_str,
              UInt32 total_entries,
              UInt32 associativity,
              UInt32 cache_block_size,
              UInt32 max_hw_sharers,
              UInt32 max_num_sharers,
              ComponentLatency dram_directory_cache_access_time,
              ShmemPerfModel * shmem_perf_model )
```

Definition at line 8 of file dram_directory_cache.cc.

References floorLog2(), m_associativity, m_cache_block_size, m_directory, m_log_cache_block_size, m_log_num_sets, m_num_sets, m_replacement_ptrs, and m_total_entries.

6.104.2.2 ~DramDirectoryCache()

```
PrL1PrL2DramDirectoryMSI::DramDirectoryCache::~DramDirectoryCache ( )
```

Definition at line 35 of file dram_directory_cache.cc.

References m_directory, and m_replacement_ptrs.

6.104.3 Member Function Documentation

6.104.3.1 getCacheBlockSize()

```
UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getCacheBlockSize ( )  [inline], [private]
```

Definition at line 32 of file dram_directory_cache.h.

References m_cache_block_size.

6.104.3.2 getDirectoryEntry()

```
DirectoryEntry * PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry (
            IntPtr address,
           bool modeled = false )
```

Definition at line 42 of file dram_directory_cache.cc.

References ShmemPerfModel::_SIM_THREAD, DirectoryEntry::getAddress(), Directory::getDirectoryEntry(), DirectoryEntry::getLatency(), ComponentLatency::getLatency(), getShmemPerfModel(), ShmemPerfModel::incr↩ ElapsedTime(), INVALID_ADDRESS, m_associativity, m_directory, m_dram_directory_cache_access_time, m_↩ replaced_directory_entry_list, m_shmem_perf_model, DirectoryEntry::setAddress(), and splitAddress().

Referenced by getReplacementCandidates(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::handleMsgFrom↩ L2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDRAMReply(), PrL1PrL2DramDirectory↩ MSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr↩ ::processFlushRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2↩ Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processNullifyReq(), PrL1PrL2DramDirectoryMSI::↩ DramDirectoryCntlr::processShReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::process↩ UpgradeReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache(), and PrL1PrL2DramDirectory↩ MSI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache().

6.104.3.3 getLogCacheBlockSize()

```
UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getLogCacheBlockSize ( )  [inline],
[private]
```

Definition at line 33 of file dram_directory_cache.h.

References m_log_cache_block_size.

Referenced by splitAddress().

6.104.3.4 getLogNumSets()

```
UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getLogNumSets ( )  [inline], [private]
```

Definition at line 35 of file dram_directory_cache.h.

References m_log_num_sets.

Referenced by splitAddress().

6.104.3.5 getMaxHwSharers()

```
UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getMaxHwSharers ( ) const  [inline]
```

Definition at line 55 of file dram_directory_cache.h.

References Directory::getMaxHwSharers(), and m_directory.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2↩DramDirectoryMSI::DramDirectoryCntlr::processShReqFromL2Cache(), and PrL1PrL2DramDirectoryMSI::Dram↩DirectoryCntlr::processUpgradeReqFromL2Cache().

6.104.3.6 getNumSets()

```
UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getNumSets ( )  [inline], [private]
```

Definition at line 34 of file dram_directory_cache.h.

References m_num_sets.

Referenced by splitAddress().

6.104.3.7 getReplacementCandidates()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getReplacementCandidates (
              IntPtr address,
             std::vector<  DirectoryEntry * > & replacement_candidate_list )
```

Definition at line 93 of file dram_directory_cache.cc.

References Directory::getDirectoryEntry(), getDirectoryEntry(), m_associativity, m_directory, m_replacement_ptrs, and splitAddress().

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDirectoryEntryAllocationReq().

6.104.3.8 getShmemPerfModel()

```
ShmemPerfModel* PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getShmemPerfModel ( )  [inline],
[private]
```

Definition at line 29 of file dram_directory_cache.h.

References m_shmem_perf_model.

Referenced by getDirectoryEntry(), and replaceDirectoryEntry().

6.104.3.9 invalidateDirectoryEntry()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCache::invalidateDirectoryEntry (
              IntPtr address )
```

Definition at line 138 of file dram_directory_cache.cc.

References LOG_PRINT_ERROR, and m_replaced_directory_entry_list.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processNullifyReq().

6.104.3.10 replaceDirectoryEntry()

```
DirectoryEntry * PrL1PrL2DramDirectoryMSI::DramDirectoryCache::replaceDirectoryEntry (
              IntPtr replaced_address,
              IntPtr address,
             bool modeled )
```

Definition at line 109 of file dram_directory_cache.cc.

References ShmemPerfModel::_SIM_THREAD, Directory::createDirectoryEntry(), DirectoryEntry::getAddress(), Directory::getDirectoryEntry(), ComponentLatency::getLatency(), getShmemPerfModel(), ShmemPerfModel::incr↩ ElapsedTime(), LOG_PRINT_ERROR, m_associativity, m_directory, m_dram_directory_cache_access_time, m_↩ replaced_directory_entry_list, m_shmem_perf_model, DirectoryEntry::setAddress(), Directory::setDirectoryEntry(), and splitAddress().

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDirectoryEntryAllocationReq().

6.104.3.11 splitAddress()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCache::splitAddress (
              IntPtr address,
              IntPtr & tag,
              UInt32 & set_index )  [private]
```

Definition at line 157 of file dram_directory_cache.cc.

References getLogCacheBlockSize(), getLogNumSets(), and getNumSets().

Referenced by getDirectoryEntry(), getReplacementCandidates(), and replaceDirectoryEntry().

6.104.4 Member Data Documentation

6.104.4.1 m_associativity

UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCache::m_associativity [private]

Definition at line 19 of file dram_directory_cache.h.

Referenced by DramDirectoryCache(), getDirectoryEntry(), getReplacementCandidates(), and replaceDirectory↩ Entry().

6.104.4.2 m_cache_block_size

UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCache::m_cache_block_size [private]

Definition at line 22 of file dram_directory_cache.h.

Referenced by DramDirectoryCache(), and getCacheBlockSize().

6.104.4.3 m_directory

Directory* PrL1PrL2DramDirectoryMSI::DramDirectoryCache::m_directory [private]

Definition at line 14 of file dram_directory_cache.h.

Referenced by DramDirectoryCache(), getDirectoryEntry(), getMaxHwSharers(), getReplacementCandidates(), replaceDirectoryEntry(), and ∼DramDirectoryCache().

6.104.4.4 m_dram_directory_cache_access_time

ComponentLatency PrL1PrL2DramDirectoryMSI::DramDirectoryCache::m_dram_directory_cache_↩ access_time [private]

Definition at line 26 of file dram_directory_cache.h.

Referenced by getDirectoryEntry(), and replaceDirectoryEntry().

6.104.4.5 m_log_cache_block_size

UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCache::m_log_cache_block_size [private]

Definition at line 24 of file dram_directory_cache.h.

Referenced by DramDirectoryCache(), and getLogCacheBlockSize().

6.104.4.6 m_log_num_sets

UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCache::m_log_num_sets [private]

Definition at line 23 of file dram_directory_cache.h.

Referenced by DramDirectoryCache(), and getLogNumSets().

6.104.4.7 m_num_sets

UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCache::m_num_sets [private]

Definition at line 21 of file dram_directory_cache.h.

Referenced by DramDirectoryCache(), and getNumSets().

6.104.4.8 m_replaced_directory_entry_list

std::vector< **DirectoryEntry*** > PrL1PrL2DramDirectoryMSI::DramDirectoryCache::m_replaced_↩
directory_entry_list [private]

Definition at line 16 of file dram_directory_cache.h.

Referenced by getDirectoryEntry(), invalidateDirectoryEntry(), and replaceDirectoryEntry().

6.104.4.9 m_replacement_ptrs

UInt32* PrL1PrL2DramDirectoryMSI::DramDirectoryCache::m_replacement_ptrs [private]

Definition at line 15 of file dram_directory_cache.h.

Referenced by DramDirectoryCache(), getReplacementCandidates(), and ∼DramDirectoryCache().

6.104.4.10 m_shmem_perf_model

ShmemPerfModel* PrL1PrL2DramDirectoryMSI::DramDirectoryCache::m_shmem_perf_model [private]

Definition at line 27 of file dram_directory_cache.h.

Referenced by getDirectoryEntry(), getShmemPerfModel(), and replaceDirectoryEntry().

6.104.4.11 m_total_entries

UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCache::m_total_entries [private]

Definition at line 18 of file dram_directory_cache.h.

Referenced by DramDirectoryCache().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **dram_directory_cache.h**
- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **dram_directory_cache.cc**

6.105 PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr Class Reference

#include <dram_directory_cntlr.h>

Public Member Functions

- **DramDirectoryCntlr** (**core_id_t** core_id, **MemoryManagerBase** *memory_manager, **AddressHome↩ Lookup** *dram_controller_home_lookup, **NucaCache** *nuca_cache, **UInt32** dram_directory_total_entries, **UInt32** dram_directory_associativity, **UInt32** cache_block_size, **UInt32** dram_directory_max_num_sharers, **UInt32** dram_directory_max_hw_sharers, String dram_directory_type_str, **ComponentLatency** dram_↩ directory_cache_access_time, **ShmemPerfModel** *shmem_perf_model)
- ~**DramDirectoryCntlr** ()
- void **handleMsgFromL2Cache** (**core_id_t** sender, **ShmemMsg** *shmem_msg)
- void **handleMsgFromDRAM** (**core_id_t** sender, **ShmemMsg** *shmem_msg)
- **DramDirectoryCache** * **getDramDirectoryCache** ()

Private Member Functions

- **UInt32** **getCacheBlockSize** ()
- **MemoryManagerBase** * **getMemoryManager** ()
- **ShmemPerfModel** * **getShmemPerfModel** ()
- **DirectoryEntry** * **processDirectoryEntryAllocationReq** (**ShmemReq** *shmem_req)
- void **processNullifyReq** (**ShmemReq** *shmem_req)
- void **processNextReqFromL2Cache** (**IntPtr** address)
- void **processExReqFromL2Cache** (**ShmemReq** *shmem_req, **Byte** *cached_data_buf=NULL)
- void **processShReqFromL2Cache** (**ShmemReq** *shmem_req, **Byte** *cached_data_buf=NULL)
- void **processWbReqFromL2Cache** (**ShmemReq** *shmem_req, **Byte** *cached_data_buf=NULL)
- void **retrieveDataAndSendToL2Cache** (**ShmemMsg::msg_t** reply_msg_type, **core_id_t** receiver, **IntPtr** address, **Byte** *cached_data_buf, **ShmemMsg** *orig_shmem_msg)
- void **processDRAMReply** (**core_id_t** sender, **ShmemMsg** *shmem_msg)
- void **processUpgradeReqFromL2Cache** (**ShmemReq** *shmem_req, **Byte** *cached_data_buf=NULL)
- void **processInvRepFromL2Cache** (**core_id_t** sender, **ShmemMsg** *shmem_msg)
- void **processFlushRepFromL2Cache** (**core_id_t** sender, **ShmemMsg** *shmem_msg)
- void **processWbRepFromL2Cache** (**core_id_t** sender, **ShmemMsg** *shmem_msg)
- void **sendDataToNUCA** (**IntPtr** address, **core_id_t** requester, **Byte** *data_buf, **SubsecondTime** now, bool count)
- void **sendDataToDram** (**IntPtr** address, **core_id_t** requester, **Byte** *data_buf, **SubsecondTime** now)
- void **updateShmemPerf** (**ShmemReq** *shmem_req, **ShmemPerf::shmem_times_type_t** reason= **ShmemPerf::UNKNOWN**)
- void **updateShmemPerf** (**ShmemMsg** *shmem_msg, **ShmemPerf::shmem_times_type_t** reason= **ShmemPerf::UNKNOWN**)

Private Attributes

- **MemoryManagerBase** * **m_memory_manager**
- **AddressHomeLookup** * **m_dram_controller_home_lookup**
- **DramDirectoryCache** * **m_dram_directory_cache**
- **ReqQueueList** * **m_dram_directory_req_queue_list**
- **NucaCache** * **m_nuca_cache**
- **core_id_t** **m_core_id**
- **UInt32** **m_cache_block_size**
- **ShmemPerfModel** * **m_shmem_perf_model**
- **ShmemPerf** **m_dummy_shmem_perf**
- **CoherencyProtocol::type_t** **m_protocol**
- **UInt64** **evict** [**DirectoryState::NUM_DIRECTORY_STATES**]
- **UInt64** **forward**
- **UInt64** **forward_failed**

6.105.1 Detailed Description

Definition at line 18 of file dram_directory_cntlr.h.

6.105.2 Constructor & Destructor Documentation

6.105.2.1 DramDirectoryCntlr()

```
PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::DramDirectoryCntlr (
              core_id_t core_id,
              MemoryManagerBase * memory_manager,
              AddressHomeLookup * dram_controller_home_lookup,
              NucaCache * nuca_cache,
              UInt32 dram_directory_total_entries,
              UInt32 dram_directory_associativity,
              UInt32 cache_block_size,
              UInt32 dram_directory_max_num_sharers,
              UInt32 dram_directory_max_hw_sharers,
             String dram_directory_type_str,
              ComponentLatency dram_directory_cache_access_time,
              ShmemPerfModel * shmem_perf_model )
```

Definition at line 34 of file dram_directory_cntlr.cc.

References PrL1PrL2DramDirectoryMSI::DStateString(), evict, forward, forward_failed, LOG_PRINT_ERROR, m_dram_directory_cache, m_dram_directory_req_queue_list, m_protocol, m_shmem_perf_model, CoherencyProtocol::MESI, CoherencyProtocol::MESIF, CoherencyProtocol::MSI, DirectoryState::NUM_DIRECTORY_STATES, registerStatsMetric(), and DirectoryState::UNCACHED.

6.105.2.2 ~DramDirectoryCntlr()

```
PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::~DramDirectoryCntlr ( )
```

Definition at line 96 of file dram_directory_cntlr.cc.

References m_dram_directory_cache, and m_dram_directory_req_queue_list.

6.105.3 Member Function Documentation

6.105.3.1 getCacheBlockSize()

```
UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::getCacheBlockSize ( ) [inline], [private]
```

Definition at line 40 of file dram_directory_cntlr.h.

References m_cache_block_size.

Referenced by processDRAMReply(), retrieveDataAndSendToL2Cache(), sendDataToDram(), and sendDataToNUCA().

6.105.3.2 getDramDirectoryCache()

```
DramDirectoryCache* PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::getDramDirectoryCache ( )
[inline]
```

Definition at line 90 of file dram_directory_cntlr.h.

References m_dram_directory_cache.

Referenced by ParametricDramDirectoryMSI::MemoryManager::getDramDirectoryCache().

6.105.3.3 getMemoryManager()

```
MemoryManagerBase* PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::getMemoryManager ( ) [inline],
[private]
```

Definition at line 41 of file dram_directory_cntlr.h.

References m_memory_manager.

Referenced by processDRAMReply(), processExReqFromL2Cache(), processNullifyReq(), processShReqFromL2Cache(), processUpgradeReqFromL2Cache(), retrieveDataAndSendToL2Cache(), sendDataToDram(), and sendDataToNUCA().

6.105.3.4 getShmemPerfModel()

```
ShmemPerfModel* PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::getShmemPerfModel ( ) [inline],
[private]
```

Definition at line 42 of file dram_directory_cntlr.h.

References m_shmem_perf_model.

Referenced by handleMsgFromL2Cache(), processDirectoryEntryAllocationReq(), processDRAMReply(), processFlushRepFromL2Cache(), processInvRepFromL2Cache(), processNextReqFromL2Cache(), processWbRepFromL2Cache(), processWbReqFromL2Cache(), retrieveDataAndSendToL2Cache(), sendDataToNUCA(), and updateShmemPerf().

6.105.3.5 handleMsgFromDRAM()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::handleMsgFromDRAM (
            core_id_t sender,
            ShmemMsg * shmem_msg )
```

Definition at line 223 of file dram_directory_cntlr.cc.

References PrL1PrL2DramDirectoryMSI::ShmemMsg::DRAM_READ_REP, PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), LOG_PRINT_ERROR, MYLOG, and processDRAMReply().

Referenced by ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork().

6.105.3.6 handleMsgFromL2Cache()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::handleMsgFromL2Cache (
            core_id_t sender,
            ShmemMsg * shmem_msg )
```

Definition at line 103 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, ReqQueueListTemplate< T_Req >::enqueue(), PrL1PrL2DramDirectoryMSI::ShmemMsg::EX_REQ, PrL1PrL2DramDirectoryMSI::ShmemMsg::FLUSH_REP, PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), ShmemPerfModel::getElapsedTime(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), getShmemPerfModel(), PrL1PrL2DramDirectoryMSI::ShmemMsg::INV_REP, LOG_PRINT_ERROR, m_dram_directory_cache, m_dram_directory_req_queue_list, MYLOG, processExReqFromL2Cache(), processFlushRepFromL2Cache(), processInvRepFromL2Cache(), processShReqFromL2Cache(), processUpgradeReqFromL2Cache(), processWbRepFromL2Cache(), processWbReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::ShmemMsg::SH_REQ, ReqQueueListTemplate< T_Req >::size(), ShmemPerf::TD_ACCESS, updateShmemPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::UPGRADE_REQ, PrL1PrL2DramDirectoryMSI::ShmemMsg::WB_REP, and PrL1PrL2DramDirectoryMSI::ShmemMsg::WB_REQ.

Referenced by ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork().

6.105.3.7 processDirectoryEntryAllocationReq()

```
DirectoryEntry * PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDirectoryEntryAllocationReq (
            ShmemReq * shmem_req ) [private]
```

Definition at line 282 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, ReqQueueListTemplate< T_Req >::enqueue(), evict, PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), ShmemPerfModel::getElapsedTime(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getReplacementCandidates(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), getShmemPerfModel(), LOG_ASSERT_ERROR, m_dram_directory_cache, m_dram_directory_req_queue_list, m_dummy_shmem_perf, MYLOG, PrL1PrL2DramDirectoryMSI::ShmemMsg::NULLIFY_REQ, processNullifyReq(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::replaceDirectoryEntry(), ReqQueueListTemplate< T_Req >::size(), and MemComponent::TAG_DIR.

Referenced by processExReqFromL2Cache(), processShReqFromL2Cache(), and processUpgradeReqFromL2Cache().

6.105.3.8 processDRAMReply()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDRAMReply (
            core_id_t sender,
            ShmemMsg * shmem_msg ) [private]
```

Definition at line 790 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, HitWhere::DRAM, HitWhere::DRAM_LOCAL, HitWhere::DRAM_REMOTE, PrL1PrL2DramDirectoryMSI::ShmemMsg::EX_REP, PrL1PrL2DramDirectoryMSI::ShmemMsg::EX_REQ, DirectoryState::EXCLUSIVE, ReqQueueListTemplate< T_Req >::front(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), getCacheBlockSize(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataBuf(), DirectoryEntry::getDirectoryBlockInfo(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), DirectoryBlockInfo::getDState(), getMemoryManager(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), getShmemPerfModel(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getWhere(), MemComponent::L2_CACHE, LOG_PRINT_ERROR, m_dram_directory_cache, m_dram_directory_req_queue_list, DirectoryState::MODIFIED, MYLOG, processNextReqFromL2Cache(), sendDataToNUCA(), MemoryManagerBase::sendMsg(), PrL1PrL2DramDirectoryMSI::ShmemMsg::SH_REP, PrL1PrL2DramDirectoryMSI::ShmemMsg::SH_REQ, DirectoryState::SHARED, ReqQueueListTemplate< T_Req >::size(), MemComponent::TAG_DIR, ShmemPerf::TD_ACCESS, updateShmemPerf(), and PrL1PrL2DramDirectoryMSI::ShmemMsg::UPGRADE_REQ.

Referenced by handleMsgFromDRAM().

6.105.3.9 processExReqFromL2Cache()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache (
              ShmemReq * shmem_req,
              Byte * cached_data_buf = NULL )  [private]
```

Definition at line 419 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, DirectoryEntry::addSharer(), MemoryManagerBase::broadcastMsg(), PrL1PrL2DramDirectoryMSI::ShmemMsg::EX_REP, DirectoryState::EXCLUSIVE, PrL1PrL2DramDirectoryMSI::ShmemMsg::FLUSH_REQ, PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), DirectoryEntry::getDirectoryBlockInfo(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), DirectoryBlockInfo::getDState(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getMaxHwSharers(), getMemoryManager(), DirectoryEntry::getOwner(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), DirectoryEntry::getSharersList(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), PrL1PrL2DramDirectoryMSI::ShmemMsg::INV_REQ, MemComponent::L2_CACHE, LOG_PRINT_ERROR, m_dram_directory_cache, m_dummy_shmem_perf, DirectoryState::MODIFIED, MYLOG, processDirectoryEntryAllocationReq(), retrieveDataAndSendToL2Cache(), MemoryManagerBase::sendMsg(), DirectoryBlockInfo::setDState(), DirectoryEntry::setOwner(), DirectoryState::SHARED, MemComponent::TAG_DIR, ShmemPerf::TD_ACCESS, DirectoryState::UNCACHED, HitWhere::UNKNOWN, and updateShmemPerf().

Referenced by handleMsgFromL2Cache(), processFlushRepFromL2Cache(), processInvRepFromL2Cache(), and processNextReqFromL2Cache().

6.105.3.10 processFlushRepFromL2Cache()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache (
              core_id_t sender,
              ShmemMsg * shmem_msg )  [private]
```

Definition at line 1141 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, PrL1PrL2DramDirectoryMSI::ShmemMsg::EX_REQ, ReqQueueListTemplate< T_Req >::front(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataBuf(), DirectoryEntry::getDirectoryBlockInfo(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), DirectoryBlockInfo::getDState(), ShmemPerfModel::getElapsedTime(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), DirectoryEntry::getNumSharers(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), getShmemPerfModel(), PrL1PrL2DramDirectoryMSI::ShmemReq::getTime(), DirectoryEntry::hasSharer(), INVALID_CORE_ID, m_dram_directory_cache, m_dram_directory_req_queue_list, MYLOG, processExReqFromL2Cache(), processNullifyReq(), processShReqFromL2Cache(), processUpgradeReqFromL2Cache(), DirectoryEntry::removeSharer(), sendDataToDram(), PrL1PrL2DramDirectoryMSI::ShmemMsg::setDataBuf(), DirectoryBlockInfo::setDState(), DirectoryEntry::setForwarder(), DirectoryEntry::setOwner(), PrL1PrL2DramDirectoryMSI::ShmemMsg::SH_REQ, DirectoryState::SHARED, ReqQueueListTemplate< T_Req >::size(), ShmemPerf::TD_ACCESS, DirectoryState::UNCACHED, ShmemPerfModel::updateElapsedTime(), updateShmemPerf(), PrL1PrL2DramDirectoryMSI::ShmemReq::updateTime(), ShmemPerf::updateTime(), and PrL1PrL2DramDirectoryMSI::ShmemMsg::UPGRADE_REQ.

Referenced by handleMsgFromL2Cache().

6.105.3.11 processInvRepFromL2Cache()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache (
              core_id_t sender,
              ShmemMsg * shmem_msg )  [private]
```

Definition at line 868 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, PrL1PrL2DramDirectoryMSI::ShmemMsg::EX_REQ, DirectoryState::EXCLUSIVE, ReqQueueListTemplate< T_Req >::front(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), DirectoryEntry::getDirectoryBlockInfo(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), DirectoryBlockInfo::getDState(), DirectoryEntry::getForwarder(), PrL1PrL2DramDirectoryMSI::ShmemReq::getForwardingFrom(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), DirectoryEntry::getNumSharers(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), getShmemPerfModel(), PrL1PrL2DramDirectoryMSI::ShmemReq::getTime(), PrL1PrL2DramDirectoryMSI::ShmemReq::getWaitForData(), DirectoryEntry::hasSharer(), ShmemPerf::INV_IMBALANCE, INVALID_CORE_ID, PrL1PrL2DramDirectoryMSI::ShmemReq::isForwarding(), LOG_ASSERT_ERROR, m_dram_directory_cache, m_dram_directory_req_queue_list, MYLOG, processExReqFromL2Cache(), processNullifyReq(), processShReqFromL2Cache(), processUpgradeReqFromL2Cache(), DirectoryEntry::removeSharer(), DirectoryBlockInfo::setDState(), DirectoryEntry::setForwarder(), PrL1PrL2DramDirectoryMSI::ShmemMsg::SH_REQ, DirectoryState::SHARED, ReqQueueListTemplate< T_Req >::size(), DirectoryState::UNCACHED, ShmemPerfModel::updateElapsedTime(), updateShmemPerf(), PrL1PrL2DramDirectoryMSI::ShmemReq::updateTime(), and PrL1PrL2DramDirectoryMSI::ShmemMsg::UPGRADE_REQ.

Referenced by handleMsgFromL2Cache().

6.105.3.12 processNextReqFromL2Cache()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processNextReqFromL2Cache (
              IntPtr address )  [private]
```

Definition at line 242 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, ReqQueueListTemplate< T_Req >::dequeue(), ReqQueueListTemplate< T_Req >::empty(), PrL1PrL2DramDirectoryMSI::ShmemMsg::EX_REQ, ReqQueueListTemplate< T_Req >::front(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), getShmemPerfModel(), PrL1PrL2DramDirectoryMSI::ShmemReq::getTime(), LOG_PRINT_ERROR, m_dram_directory_req_queue_list, MYLOG, processExReqFromL2Cache(), processShReqFromL2Cache(), processUpgradeReqFromL2Cache(), ShmemPerfModel::setElapsedTime(), PrL1PrL2DramDirectoryMSI::ShmemMsg::SH_REQ, ReqQueueListTemplate< T_Req >::size(), and PrL1PrL2DramDirectoryMSI::ShmemMsg::UPGRADE_REQ.

Referenced by processDRAMReply(), processNullifyReq(), processUpgradeReqFromL2Cache(), and retrieveDataAndSendToL2Cache().

6.105.3.13 processNullifyReq()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processNullifyReq (
              ShmemReq * shmem_req )  [private]
```

Definition at line 335 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, MemoryManagerBase::broadcastMsg(), DirectoryState::EXCLUSIVE, PrL1PrL2DramDirectoryMSI::ShmemMsg::FLUSH_REQ, PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), DirectoryEntry::getDirectoryBlockInfo(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), DirectoryBlockInfo::getDState(), getMemoryManager(), DirectoryEntry::getOwner(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), DirectoryEntry::getSharersList(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), PrL1PrL2DramDirectoryMSI::ShmemMsg::INV_REQ, PrL1PrL2DramDirectoryMSI::DramDirectoryCache::invalidateDirectoryEntry(), MemComponent::L2_CACHE, LOG_PRINT_ERROR, m_dram_directory_cache, m_dummy_shmem_perf, DirectoryState::MODIFIED, MYLOG, processNextReqFromL2Cache(), MemoryManagerBase::sendMsg(), DirectoryState::SHARED, MemComponent::TAG_DIR, DirectoryState::UNCACHED, and HitWhere::UNKNOWN.

Referenced by processDirectoryEntryAllocationReq(), processFlushRepFromL2Cache(), and processInvRepFromL2Cache().

6.105.3.14 processShReqFromL2Cache()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processShReqFromL2Cache (
              ShmemReq * shmem_req,
              Byte * cached_data_buf = NULL )  [private]
```

Definition at line 511 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, DirectoryEntry::addSharer(), PrL1PrL2DramDirectoryMSI::ShmemMsg::EX_REP, DirectoryState::EXCLUSIVE, forward, forward_failed, PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), DirectoryEntry::getDirectoryBlockInfo(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), DirectoryBlockInfo::getDState(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getMaxHwSharers(), getMemoryManager(), DirectoryEntry::getOneSharer(), DirectoryEntry::getOwner(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), DirectoryEntry::hasSharer(), PrL1PrL2DramDirectoryMSI::ShmemMsg::INV_REQ, MemComponent::L2_CACHE, LOG_PRINT_ERROR, m_dram_directory_cache, m_protocol, DirectoryState::MODIFIED, CoherencyProtocol::MSI, MYLOG, processDirectoryEntryAllocationReq(), retrieveDataAndSendToL2Cache(), MemoryManagerBase::sendMsg(), DirectoryBlockInfo::setDState(), DirectoryEntry::setOwner(), PrL1PrL2DramDirectoryMSI::ShmemMsg::SH_REP, DirectoryState::SHARED, MemComponent::TAG_DIR, ShmemPerf::TD_ACCESS, DirectoryState::UNCACHED, HitWhere::UNKNOWN, updateShmemPerf(), and PrL1PrL2DramDirectoryMSI::ShmemMsg::WB_REQ.

Referenced by handleMsgFromL2Cache(), processFlushRepFromL2Cache(), processInvRepFromL2Cache(), processNextReqFromL2Cache(), processWbRepFromL2Cache(), and processWbReqFromL2Cache().

6.105.3.15 processUpgradeReqFromL2Cache()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processUpgradeReqFromL2Cache (
              ShmemReq * shmem_req,
              Byte * cached_data_buf = NULL )  [private]
```

Definition at line 972 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, DirectoryEntry::addSharer(), MemoryManagerBase::broadcastMsg(), PrL1PrL2DramDirectoryMSI::ShmemMsg::EX_REP, DirectoryState::EXCLUSIVE, PrL1PrL2DramDirectoryMSI::ShmemMsg::FLUSH_REQ, PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataBuf(), DirectoryEntry::getDirectoryBlockInfo(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), DirectoryBlockInfo::getDState(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getMaxHwSharers(), getMemoryManager(), DirectoryEntry::getOwner(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), DirectoryEntry::getSharersList(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), DirectoryEntry::hasSharer(), PrL1PrL2DramDirectoryMSI::ShmemMsg::INV_REQ, MemComponent::L2_CACHE, LOG_PRINT_ERROR, m_dram_directory_cache, m_dummy_shmem_perf, DirectoryState::MODIFIED, MYLOG, processDirectoryEntryAllocationReq(), processNextReqFromL2Cache(), retrieveDataAndSendToL2Cache(), MemoryManagerBase::sendMsg(), DirectoryBlockInfo::setDState(), DirectoryEntry::setOwner(), DirectoryState::SHARED, MemComponent::TAG_DIR, ShmemPerf::TD_ACCESS, DirectoryState::UNCACHED, HitWhere::UNKNOWN, updateShmemPerf(), and PrL1PrL2DramDirectoryMSI::ShmemMsg::UPGRADE_REP.

Referenced by handleMsgFromL2Cache(), processFlushRepFromL2Cache(), processInvRepFromL2Cache(), and processNextReqFromL2Cache().

6.105.3.16 processWbRepFromL2Cache()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbRepFromL2Cache (
              core_id_t sender,
              ShmemMsg * shmem_msg )  [private]
```

Definition at line 1229 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, ReqQueueListTemplate< T_Req >::front(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataBuf(), DirectoryEntry::getDirectoryBlockInfo(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), ShmemPerfModel::getElapsedTime(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), getShmemPerfModel(), PrL1PrL2DramDirectoryMSI::ShmemReq::getTime(), DirectoryEntry::hasSharer(), INVALID_CORE_ID, LOG_ASSERT_ERROR, LOG_PRINT_ERROR, m_dram_directory_cache, m_dram_directory_req_queue_list, MYLOG, processShReqFromL2Cache(), DirectoryBlockInfo::setDState(), DirectoryEntry::setOwner(), PrL1PrL2DramDirectoryMSI::ShmemMsg::SH_REQ, DirectoryState::SHARED, ReqQueueListTemplate< T_Req >::size(), ShmemPerf::TD_ACCESS, ShmemPerfModel::updateElapsedTime(), updateShmemPerf(), PrL1PrL2DramDirectoryMSI::ShmemReq::updateTime(), and ShmemPerf::updateTime().

Referenced by handleMsgFromL2Cache().

6.105.3.17 processWbReqFromL2Cache()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache (
                ShmemReq * shmem_req,
                Byte * cached_data_buf = NULL )  [private]
```

Definition at line 629 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, ReqQueueListTemplate< T_Req >::front(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataBuf(), DirectoryEntry::getDirectoryBlockInfo(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), ShmemPerfModel::getElapsedTime(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), getShmemPerfModel(), PrL1PrL2DramDirectoryMSI::ShmemReq::getTime(), INVALID_CORE_ID, LOG_ASSERT_ERROR, LOG_PRINT_ERROR, m_dram_directory_cache, m_dram_directory_req_queue_list, MYLOG, processShReqFromL2Cache(), DirectoryBlockInfo::setDState(), DirectoryEntry::setOwner(), PrL1PrL2DramDirectoryMSI::ShmemMsg::SH_REQ, DirectoryState::SHARED, ReqQueueListTemplate< T_Req >::size(), ShmemPerf::TD_ACCESS, ShmemPerfModel::updateElapsedTime(), updateShmemPerf(), PrL1PrL2DramDirectoryMSI::ShmemReq::updateTime(), and ShmemPerf::updateTime().

Referenced by handleMsgFromL2Cache().

6.105.3.18 retrieveDataAndSendToL2Cache()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache (
                ShmemMsg::msg_t reply_msg_type,
                core_id_t receiver,
                IntPtr address,
                Byte * cached_data_buf,
                ShmemMsg * orig_shmem_msg )  [private]
```

Definition at line 679 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, HitWhere::CACHE_REMOTE, MemComponent::DRAM, PrL1PrL2DramDirectoryMSI::ShmemMsg::DRAM_READ_REQ, ReqQueueListTemplate< T_Req >::front(), getCacheBlockSize(), DirectoryEntry::getDirectoryBlockInfo(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::getDirectoryEntry(), DirectoryBlockInfo::getDState(), DirectoryEntry::getForwarder(), PrL1PrL2DramDirectoryMSI::ShmemReq::getForwardingFrom(), AddressHomeLookup::getHome(), getMemoryManager(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg(), getShmemPerfModel(), ShmemPerfModel::incrElapsedTime(), INVALID_CORE_ID, MemComponent::L2_CACHE, m_dram_controller_home_lookup, m_dram_directory_cache, m_dram_directory_req_queue_list, m_nuca_cache, m_protocol, CoherencyProtocol::MESIF, HitWhere::MISS, MYLOG, HitWhere::NUCA_CACHE, processNextReqFromL2Cache(), NucaCache::read(), MemoryManagerBase::sendMsg(), DirectoryEntry::setForwarder(), PrL1PrL2DramDirectoryMSI::ShmemReq::setForwardingFrom(), PrL1PrL2DramDirectoryMSI::ShmemReq::setWaitForData(), DirectoryState::SHARED, ReqQueueListTemplate< T_Req >::size(), MemComponent::TAG_DIR, HitWhere::UNKNOWN, PrL1PrL2DramDirectoryMSI::ShmemMsg::UPGRADE_REQ, and PrL1PrL2DramDirectoryMSI::ShmemMsg::WB_REQ.

Referenced by processExReqFromL2Cache(), processShReqFromL2Cache(), and processUpgradeReqFromL2Cache().

6.105.3.19 sendDataToDram()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::sendDataToDram (
              IntPtr address,
              core_id_t requester,
              Byte * data_buf,
              SubsecondTime now )  [private]
```

Definition at line 1308 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, MemComponent::DRAM, PrL1PrL2DramDirectoryMSI::ShmemMsg::DRAM_WRITE_REQ, getCacheBlockSize(), AddressHomeLookup::getHome(), getMemoryManager(), m_dram_controller_home_lookup, m_dummy_shmem_perf, m_nuca_cache, MYLOG, sendDataToNUCA(), MemoryManagerBase::sendMsg(), MemComponent::TAG_DIR, and HitWhere::UNKNOWN.

Referenced by processFlushRepFromL2Cache().

6.105.3.20 sendDataToNUCA()

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::sendDataToNUCA (
              IntPtr address,
              core_id_t requester,
              Byte * data_buf,
              SubsecondTime now,
             bool count )  [private]
```

Definition at line 1273 of file dram_directory_cntlr.cc.

References ShmemPerfModel::_SIM_THREAD, MemComponent::DRAM, PrL1PrL2DramDirectoryMSI::ShmemMsg::DRAM_WRITE_REQ, getCacheBlockSize(), AddressHomeLookup::getHome(), getMemoryManager(), getShmemPerfModel(), m_core_id, m_dram_controller_home_lookup, m_dummy_shmem_perf, m_nuca_cache, MYLOG, MemoryManagerBase::sendMsg(), MemComponent::TAG_DIR, HitWhere::UNKNOWN, and NucaCache::write().

Referenced by processDRAMReply(), and sendDataToDram().

6.105.3.21 updateShmemPerf() [1/2]

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::updateShmemPerf (
              ShmemMsg * shmem_msg,
              ShmemPerf::shmem_times_type_t reason =  ShmemPerf::UNKNOWN )  [inline], [private]
```

Definition at line 67 of file dram_directory_cntlr.h.

References ShmemPerfModel::_SIM_THREAD, PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), getShmemPerfModel(), and ShmemPerf::updateTime().

6.105.3.22 updateShmemPerf() [2/2]

```
void PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::updateShmemPerf (
              ShmemReq * shmem_req,
              ShmemPerf::shmem_times_type_t reason =  ShmemPerf::UNKNOWN )  [inline], [private]
```

Definition at line 63 of file dram_directory_cntlr.h.

References PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg().

Referenced by handleMsgFromL2Cache(), processDRAMReply(), processExReqFromL2Cache(), processFlushRepFromL2Cache(), processInvRepFromL2Cache(), processShReqFromL2Cache(), processUpgradeReqFromL2Cache(), processWbRepFromL2Cache(), and processWbReqFromL2Cache().

6.105.4 Member Data Documentation

6.105.4.1 evict

```
UInt64 PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::evict[ DirectoryState::NUM_DIRECTORY_STATES]  [private]
```

Definition at line 37 of file dram_directory_cntlr.h.

Referenced by DramDirectoryCntlr(), and processDirectoryEntryAllocationReq().

6.105.4.2 forward

```
UInt64 PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::forward  [private]
```

Definition at line 38 of file dram_directory_cntlr.h.

Referenced by DramDirectoryCntlr(), and processShReqFromL2Cache().

6.105.4.3 forward_failed

```
UInt64 PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::forward_failed  [private]
```

Definition at line 38 of file dram_directory_cntlr.h.

Referenced by DramDirectoryCntlr(), and processShReqFromL2Cache().

6.105.4.4 m_cache_block_size

UInt32 PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::m_cache_block_size [private]

Definition at line 30 of file dram_directory_cntlr.h.

Referenced by getCacheBlockSize().

6.105.4.5 m_core_id

core_id_t PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::m_core_id [private]

Definition at line 29 of file dram_directory_cntlr.h.

Referenced by sendDataToNUCA().

6.105.4.6 m_dram_controller_home_lookup

AddressHomeLookup* PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::m_dram_controller_home_↩
lookup [private]

Definition at line 23 of file dram_directory_cntlr.h.

Referenced by retrieveDataAndSendToL2Cache(), sendDataToDram(), and sendDataToNUCA().

6.105.4.7 m_dram_directory_cache

DramDirectoryCache* PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::m_dram_directory_cache
[private]

Definition at line 24 of file dram_directory_cntlr.h.

Referenced by DramDirectoryCntlr(), getDramDirectoryCache(), handleMsgFromL2Cache(), processDirectory↩
EntryAllocationReq(), processDRAMReply(), processExReqFromL2Cache(), processFlushRepFromL2Cache(), processInvRepFromL2Cache(), processNullifyReq(), processShReqFromL2Cache(), processUpgradeReqFrom↩
L2Cache(), processWbRepFromL2Cache(), processWbReqFromL2Cache(), retrieveDataAndSendToL2Cache(), and ∼DramDirectoryCntlr().

6.105.4.8 m_dram_directory_req_queue_list

ReqQueueList* PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::m_dram_directory_req_queue_list [private]

Definition at line 25 of file dram_directory_cntlr.h.

Referenced by DramDirectoryCntlr(), handleMsgFromL2Cache(), processDirectoryEntryAllocationReq(), processDRAMReply(), processFlushRepFromL2Cache(), processInvRepFromL2Cache(), processNextReqFromL2Cache(), processWbRepFromL2Cache(), processWbReqFromL2Cache(), retrieveDataAndSendToL2Cache(), and ~DramDirectoryCntlr().

6.105.4.9 m_dummy_shmem_perf

ShmemPerf PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::m_dummy_shmem_perf [private]

Definition at line 33 of file dram_directory_cntlr.h.

Referenced by processDirectoryEntryAllocationReq(), processExReqFromL2Cache(), processNullifyReq(), processUpgradeReqFromL2Cache(), sendDataToDram(), and sendDataToNUCA().

6.105.4.10 m_memory_manager

MemoryManagerBase* PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::m_memory_manager [private]

Definition at line 22 of file dram_directory_cntlr.h.

Referenced by getMemoryManager().

6.105.4.11 m_nuca_cache

NucaCache* PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::m_nuca_cache [private]

Definition at line 27 of file dram_directory_cntlr.h.

Referenced by retrieveDataAndSendToL2Cache(), sendDataToDram(), and sendDataToNUCA().

6.105.4.12 m_protocol

CoherencyProtocol::type_t PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::m_protocol [private]

Definition at line 35 of file dram_directory_cntlr.h.

Referenced by DramDirectoryCntlr(), processShReqFromL2Cache(), and retrieveDataAndSendToL2Cache().

6.105.4.13 m_shmem_perf_model

```
ShmemPerfModel* PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::m_shmem_perf_model  [private]
```

Definition at line 32 of file dram_directory_cntlr.h.

Referenced by DramDirectoryCntlr(), and getShmemPerfModel().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **dram_directory_cntlr.h**
- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **dram_directory_cntlr.cc**

6.106 DramDirectoryPerfModel Class Reference

```
#include <dram_directory_perf_model.h>
```

Inheritance diagram for DramDirectoryPerfModel:

Public Member Functions

- **DramDirectoryPerfModel** ()
- ~**DramDirectoryPerfModel** ()
- **UInt32 getLatency** (**DramDirActions_t** action)

Additional Inherited Members

6.106.1 Detailed Description

Definition at line 6 of file dram_directory_perf_model.h.

6.106.2 Constructor & Destructor Documentation

6.106.2.1 DramDirectoryPerfModel()

```
DramDirectoryPerfModel::DramDirectoryPerfModel ( )  [inline]
```

Definition at line 11 of file dram_directory_perf_model.h.

6.106.2.2 ~DramDirectoryPerfModel()

```
DramDirectoryPerfModel::~DramDirectoryPerfModel ( )  [inline]
```

Definition at line 12 of file dram_directory_perf_model.h.

6.106.3 Member Function Documentation

6.106.3.1 getLatency()

```
UInt32 DramDirectoryPerfModel::getLatency (
            DramDirActions_t action )  [inline], [virtual]
```

Reimplemented from **DramDirectoryPerfModelBase** (p. 494).

Definition at line 14 of file dram_directory_perf_model.h.

References DramDirectoryPerfModelBase::ACCESS_DIR_CACHE, DramDirectoryPerfModelBase::access_dir_cache_delay, DramDirectoryPerfModelBase::DEQUEUE_REQUEST, DramDirectoryPerfModelBase::dequeue_request_delay, DramDirectoryPerfModelBase::ENQUEUE_REQUEST, DramDirectoryPerfModelBase::enqueue_request_delay, LOG_ASSERT_ERROR, DramDirectoryPerfModelBase::PROCESS_ACK, DramDirectoryPerfModelBase::process_ack_delay, DramDirectoryPerfModelBase::PROCESS_REQUEST, DramDirectoryPerfModelBase::process_request_delay, DramDirectoryPerfModelBase::RECEIVE_MESSAGE, DramDirectoryPerfModelBase::SEND_MESSAGE, DramDirectoryPerfModelBase::shmem_receive_message_delay, and DramDirectoryPerfModelBase::shmem_send_message_delay.

The documentation for this class was generated from the following file:

- common/performance_model/ **dram_directory_perf_model.h**

6.107 DramDirectoryPerfModelBase Class Reference

```
#include <dram_directory_perf_model_base.h>
```

Inheritance diagram for DramDirectoryPerfModelBase:

Public Types

- enum **DramDirActions_t** { **ACCESS_DIR_CACHE** = 0, **ENQUEUE_REQUEST**, **DEQUEUE_REQUEST**, **PROCESS_REQUEST**, **PROCESS_ACK**, **SEND_MESSAGE**, **RECEIVE_MESSAGE**, **NUM_DRAM_DIR_ACTIONS** }
- enum **DramDirectoryPerfModel_t** { **DRAM_DIRECTORY_PERF_MODEL** = 0, **NUM_DRAM_DIRECTORY_PERF_MODELS** }

Public Member Functions

- **DramDirectoryPerfModelBase** ()
- virtual ~**DramDirectoryPerfModelBase** ()
- virtual **UInt32 getLatency** (**DramDirActions_t** action)

Static Public Member Functions

- static **DramDirectoryPerfModelBase** * **createModel** (**UInt32** type)

Protected Attributes

- **UInt32 access_dir_cache_delay**
- **UInt32 enqueue_request_delay**
- **UInt32 dequeue_request_delay**
- **UInt32 process_request_delay**
- **UInt32 process_ack_delay**
- **UInt32 shmem_send_message_delay**
- **UInt32 shmem_receive_message_delay**

6.107.1 Detailed Description

Definition at line 9 of file dram_directory_perf_model_base.h.

6.107.2 Member Enumeration Documentation

6.107.2.1 DramDirActions_t

enum **DramDirectoryPerfModelBase::DramDirActions_t**

Enumerator

ACCESS_DIR_CACHE	
ENQUEUE_REQUEST	
DEQUEUE_REQUEST	
PROCESS_REQUEST	
PROCESS_ACK	
SEND_MESSAGE	
RECEIVE_MESSAGE	
NUM_DRAM_DIR_ACTIONS	

Definition at line 37 of file dram_directory_perf_model_base.h.

6.107.2.2 DramDirectoryPerfModel_t

```
enum DramDirectoryPerfModelBase::DramDirectoryPerfModel_t
```

Enumerator

DRAM_DIRECTORY_PERF_MODEL	
NUM_DRAM_DIRECTORY_PERF_MODELS	

Definition at line 49 of file dram_directory_perf_model_base.h.

6.107.3 Constructor & Destructor Documentation

6.107.3.1 DramDirectoryPerfModelBase()

```
DramDirectoryPerfModelBase::DramDirectoryPerfModelBase ( ) [inline]
```

Definition at line 22 of file dram_directory_perf_model_base.h.

References access_dir_cache_delay.

6.107.3.2 ~DramDirectoryPerfModelBase()

```
virtual DramDirectoryPerfModelBase::~DramDirectoryPerfModelBase ( ) [inline], [virtual]
```

Definition at line 33 of file dram_directory_perf_model_base.h.

6.107.4 Member Function Documentation

6.107.4.1 createModel()

```
DramDirectoryPerfModelBase * DramDirectoryPerfModelBase::createModel (
            UInt32 type ) [static]
```

Definition at line 5 of file dram_directory_perf_model_base.cc.

References DRAM_DIRECTORY_PERF_MODEL, and LOG_ASSERT_ERROR.

6.107.4.2 getLatency()

```
virtual  UInt32 DramDirectoryPerfModelBase::getLatency (
             DramDirActions_t action )  [inline], [virtual]
```

Reimplemented in **DramDirectoryPerfModel** (p. 492).

Definition at line 55 of file dram_directory_perf_model_base.h.

6.107.5 Member Data Documentation

6.107.5.1 access_dir_cache_delay

```
UInt32 DramDirectoryPerfModelBase::access_dir_cache_delay  [protected]
```

Definition at line 12 of file dram_directory_perf_model_base.h.

Referenced by DramDirectoryPerfModelBase(), and DramDirectoryPerfModel::getLatency().

6.107.5.2 dequeue_request_delay

```
UInt32 DramDirectoryPerfModelBase::dequeue_request_delay  [protected]
```

Definition at line 14 of file dram_directory_perf_model_base.h.

Referenced by DramDirectoryPerfModel::getLatency().

6.107.5.3 enqueue_request_delay

```
UInt32 DramDirectoryPerfModelBase::enqueue_request_delay  [protected]
```

Definition at line 13 of file dram_directory_perf_model_base.h.

Referenced by DramDirectoryPerfModel::getLatency().

6.107.5.4 process_ack_delay

```
UInt32 DramDirectoryPerfModelBase::process_ack_delay  [protected]
```

Definition at line 16 of file dram_directory_perf_model_base.h.

Referenced by DramDirectoryPerfModel::getLatency().

6.107.5.5 process_request_delay

```
UInt32 DramDirectoryPerfModelBase::process_request_delay  [protected]
```

Definition at line 15 of file dram_directory_perf_model_base.h.

Referenced by DramDirectoryPerfModel::getLatency().

6.107.5.6 shmem_receive_message_delay

```
UInt32 DramDirectoryPerfModelBase::shmem_receive_message_delay  [protected]
```

Definition at line 18 of file dram_directory_perf_model_base.h.

Referenced by DramDirectoryPerfModel::getLatency().

6.107.5.7 shmem_send_message_delay

```
UInt32 DramDirectoryPerfModelBase::shmem_send_message_delay  [protected]
```

Definition at line 17 of file dram_directory_perf_model_base.h.

Referenced by DramDirectoryPerfModel::getLatency().

The documentation for this class was generated from the following files:

- common/performance_model/ **dram_directory_perf_model_base.h**
- common/performance_model/ **dram_directory_perf_model_base.cc**

6.108 DramPerfModel Class Reference

```
#include <dram_perf_model.h>
```

Inheritance diagram for DramPerfModel:

Public Member Functions

- **DramPerfModel** (**core_id_t** core_id, **UInt64** cache_block_size)
- virtual ~**DramPerfModel** ()
- virtual **SubsecondTime** **getAccessLatency** (**SubsecondTime** pkt_time, **UInt64** pkt_size, **core_id_t** requester, **IntPtr** address, **DramCntlrInterface::access_t** access_type, **ShmemPerf** *perf)=0
- void **enable** ()
- void **disable** ()
- **UInt64** **getTotalAccesses** ()

Static Public Member Functions

- static **DramPerfModel** * **createDramPerfModel** (**core_id_t** core_id, **UInt32** cache_block_size)

Protected Member Functions

- void **dramAccessed** (**SubsecondTime** pkt_time, **UInt64** pkt_size, **core_id_t** requester, **IntPtr** address, **DramCntlrInterface::access_t** access_type, **ShmemPerf** *perf)

Protected Attributes

- bool **m_enabled**
- **UInt64** **m_num_accesses**
- FILE * **dram_log_file**

6.108.1 Detailed Description

Definition at line 22 of file dram_perf_model.h.

6.108.2 Constructor & Destructor Documentation

6.108.2.1 DramPerfModel()

```
DramPerfModel::DramPerfModel (
              core_id_t core_id,
              UInt64 cache_block_size )  [inline]
```

Definition at line 47 of file dram_perf_model.h.

References dram_log_file.

6.108.2.2 ~DramPerfModel()

```
virtual DramPerfModel::~DramPerfModel ( )  [inline], [virtual]
```

Definition at line 53 of file dram_perf_model.h.

References dram_log_file.

6.108.3 Member Function Documentation

6.108.3.1 createDramPerfModel()

```
DramPerfModel * DramPerfModel::createDramPerfModel (
            core_id_t core_id,
            UInt32 cache_block_size )  [static]
```

Definition at line 8 of file dram_perf_model.cc.

References LOG_PRINT_ERROR.

Referenced by PrL1PrL2DramDirectoryMSI::DramCntlr::DramCntlr().

6.108.3.2 disable()

```
void DramPerfModel::disable ( )  [inline]
```

Definition at line 58 of file dram_perf_model.h.

References m_enabled.

Referenced by ParametricDramDirectoryMSI::MemoryManager::disableModels().

6.108.3.3 dramAccessed()

```
void DramPerfModel::dramAccessed (
              SubsecondTime pkt_time,
              UInt64 pkt_size,
              core_id_t requester,
              IntPtr address,
              DramCntlrInterface::access_t access_type,
              ShmemPerf * perf )  [protected]
```

Register DRAM access.

- pkt_time
- pkt_size
- requester
- address
- access_type
- perf

Implemented by Kleber Kruger

Definition at line 30 of file dram_perf_model.cc.

References dram_log_file, SubsecondTime::getNS(), and DramCntlrInterface::WRITE.

Referenced by DramPerfModelConstant::getAccessLatency().

6.108.3.4 enable()

```
void DramPerfModel::enable ( )  [inline]
```

Definition at line 57 of file dram_perf_model.h.

References m_enabled.

Referenced by ParametricDramDirectoryMSI::MemoryManager::enableModels().

6.108.3.5 getAccessLatency()

```
virtual  SubsecondTime DramPerfModel::getAccessLatency (
              SubsecondTime pkt_time,
              UInt64 pkt_size,
              core_id_t requester,
              IntPtr address,
              DramCntlrInterface::access_t access_type,
              ShmemPerf * perf )  [pure virtual]
```

Implemented in **DramPerfModelReadWrite** (p. 507), **DramPerfModelNormal** (p. 505), and **DramPerfModelConstant** (p. 502).

Referenced by PrL1PrL2DramDirectoryMSI::DramCntlr::runDramPerfModel().

6.108.3.6 getTotalAccesses()

```
UInt64 DramPerfModel::getTotalAccesses ( )  [inline]
```

Definition at line 60 of file dram_perf_model.h.

References m_num_accesses.

6.108.4 Member Data Documentation

6.108.4.1 dram_log_file

```
FILE* DramPerfModel::dram_log_file  [protected]
```

Definition at line 28 of file dram_perf_model.h.

Referenced by dramAccessed(), DramPerfModel(), and ~DramPerfModel().

6.108.4.2 m_enabled

```
bool DramPerfModel::m_enabled  [protected]
```

Definition at line 25 of file dram_perf_model.h.

Referenced by disable(), enable(), DramPerfModelConstant::getAccessLatency(), DramPerfModelNormal::getAccessLatency(), and DramPerfModelReadWrite::getAccessLatency().

6.108.4.3 m_num_accesses

```
UInt64 DramPerfModel::m_num_accesses  [protected]
```

Definition at line 26 of file dram_perf_model.h.

Referenced by DramPerfModelConstant::getAccessLatency(), DramPerfModelNormal::getAccessLatency(), DramPerfModelReadWrite::getAccessLatency(), and getTotalAccesses().

The documentation for this class was generated from the following files:

- common/performance_model/ **dram_perf_model.h**
- common/performance_model/ **dram_perf_model.cc**

6.109 DramPerfModelConstant Class Reference

`#include <dram_perf_model_constant.h>`

Inheritance diagram for DramPerfModelConstant:

Public Member Functions

- **DramPerfModelConstant** (**core_id_t** core_id, **UInt32** cache_block_size)
- ~**DramPerfModelConstant** ()
- **SubsecondTime getAccessLatency** (**SubsecondTime** pkt_time, **UInt64** pkt_size, **core_id_t** requester, **IntPtr** address, **DramCntlrInterface::access_t** access_type, **ShmemPerf** *perf)

Private Attributes

- **QueueModel** * **m_queue_model**
- **SubsecondTime m_dram_access_cost**
- **ComponentBandwidth m_dram_bandwidth**
- **SubsecondTime m_total_queueing_delay**
- **SubsecondTime m_total_access_latency**

Additional Inherited Members

6.109.1 Detailed Description

Definition at line 10 of file dram_perf_model_constant.h.

6.109.2 Constructor & Destructor Documentation

6.109.2.1 DramPerfModelConstant()

```
DramPerfModelConstant::DramPerfModelConstant (
            core_id_t core_id,
            UInt32 cache_block_size )
```

Definition at line 8 of file dram_perf_model_constant.cc.

References QueueModel::create(), SubsecondTime::FS(), ComponentBandwidth::getRoundedLatency(), m_↩ dram_access_cost, m_dram_bandwidth, m_queue_model, m_total_access_latency, m_total_queueing_delay, TimeConverter< T >::NStoFS(), and registerStatsMetric().

6.109.2.2 ~DramPerfModelConstant()

```
DramPerfModelConstant::~DramPerfModelConstant ( )
```

Definition at line 28 of file dram_perf_model_constant.cc.

References m_queue_model.

6.109.3 Member Function Documentation

6.109.3.1 getAccessLatency()

```
SubsecondTime DramPerfModelConstant::getAccessLatency (
            SubsecondTime pkt_time,
            UInt64 pkt_size,
            core_id_t requester,
            IntPtr address,
            DramCntlrInterface::access_t access_type,
            ShmemPerf * perf )  [virtual]
```

Implements **DramPerfModel** (p. 499).

Definition at line 38 of file dram_perf_model_constant.cc.

References QueueModel::computeQueueDelay(), ShmemPerf::DRAM_BUS, ShmemPerf::DRAM_DEVICE, ShmemPerf::DRAM_QUEUE, DramPerfModel::dramAccessed(), ComponentBandwidth::getRoundedLatency(), Config::getSingleton(), m_dram_access_cost, m_dram_bandwidth, DramPerfModel::m_enabled, DramPerfModel::m_num_accesses, m_queue_model, m_total_access_latency, m_total_queueing_delay, ShmemPerf::updateTime(), and SubsecondTime::Zero().

6.109.4 Member Data Documentation

6.109.4.1 m_dram_access_cost

```
SubsecondTime DramPerfModelConstant::m_dram_access_cost  [private]
```

Definition at line 14 of file dram_perf_model_constant.h.

Referenced by DramPerfModelConstant(), and getAccessLatency().

6.109.4.2 m_dram_bandwidth

```
ComponentBandwidth DramPerfModelConstant::m_dram_bandwidth [private]
```

Definition at line 15 of file dram_perf_model_constant.h.

Referenced by DramPerfModelConstant(), and getAccessLatency().

6.109.4.3 m_queue_model

```
QueueModel* DramPerfModelConstant::m_queue_model [private]
```

Definition at line 13 of file dram_perf_model_constant.h.

Referenced by DramPerfModelConstant(), getAccessLatency(), and ~DramPerfModelConstant().

6.109.4.4 m_total_access_latency

```
SubsecondTime DramPerfModelConstant::m_total_access_latency [private]
```

Definition at line 18 of file dram_perf_model_constant.h.

Referenced by DramPerfModelConstant(), and getAccessLatency().

6.109.4.5 m_total_queueing_delay

```
SubsecondTime DramPerfModelConstant::m_total_queueing_delay [private]
```

Definition at line 17 of file dram_perf_model_constant.h.

Referenced by DramPerfModelConstant(), and getAccessLatency().

The documentation for this class was generated from the following files:

- common/performance_model/ **dram_perf_model_constant.h**
- common/performance_model/ **dram_perf_model_constant.cc**

6.110 DramPerfModelNormal Class Reference

```
#include <dram_perf_model_normal.h>
```

Inheritance diagram for DramPerfModelNormal:

Public Member Functions

- **DramPerfModelNormal** (**core_id_t** core_id, **UInt32** cache_block_size)
- ∼**DramPerfModelNormal** ()
- **SubsecondTime getAccessLatency** (**SubsecondTime** pkt_time, **UInt64** pkt_size, **core_id_t** requester, **IntPtr** address, **DramCntlrInterface::access_t** access_type, **ShmemPerf** ∗perf)

Private Attributes

- **QueueModel** ∗ **m_queue_model**
- **TimeDistribution** ∗ **m_dram_access_cost**
- **ComponentBandwidth m_dram_bandwidth**
- **SubsecondTime m_total_queueing_delay**
- **SubsecondTime m_total_access_latency**

Additional Inherited Members

6.110.1 Detailed Description

Definition at line 11 of file dram_perf_model_normal.h.

6.110.2 Constructor & Destructor Documentation

6.110.2.1 DramPerfModelNormal()

```
DramPerfModelNormal::DramPerfModelNormal (
              core_id_t core_id,
              UInt32 cache_block_size )
```

Definition at line 8 of file dram_perf_model_normal.cc.

References QueueModel::create(), SubsecondTime::FS(), ComponentBandwidth::getRoundedLatency(), m_↩ dram_access_cost, m_dram_bandwidth, m_queue_model, m_total_access_latency, m_total_queueing_delay, TimeConverter< T >::NStoFS(), and registerStatsMetric().

6.110.2.2 ∼DramPerfModelNormal()

```
DramPerfModelNormal::∼DramPerfModelNormal ( )
```

Definition at line 31 of file dram_perf_model_normal.cc.

References m_dram_access_cost, and m_queue_model.

6.110.3 Member Function Documentation

6.110.3.1 getAccessLatency()

```
SubsecondTime DramPerfModelNormal::getAccessLatency (
            SubsecondTime pkt_time,
            UInt64 pkt_size,
            core_id_t requester,
            IntPtr address,
            DramCntlrInterface::access_t access_type,
            ShmemPerf * perf )  [virtual]
```

Implements **DramPerfModel** (p. 499).

Definition at line 42 of file dram_perf_model_normal.cc.

References QueueModel::computeQueueDelay(), ShmemPerf::DRAM_BUS, ShmemPerf::DRAM_DEVICE, ShmemPerf::DRAM_QUEUE, ComponentBandwidth::getRoundedLatency(), Config::getSingleton(), m_dram_↩access_cost, m_dram_bandwidth, DramPerfModel::m_enabled, DramPerfModel::m_num_accesses, m_queue_↩model, m_total_access_latency, m_total_queueing_delay, TimeDistribution::next(), ShmemPerf::updateTime(), and SubsecondTime::Zero().

6.110.4 Member Data Documentation

6.110.4.1 m_dram_access_cost

```
TimeDistribution* DramPerfModelNormal::m_dram_access_cost  [private]
```

Definition at line 15 of file dram_perf_model_normal.h.

Referenced by DramPerfModelNormal(), getAccessLatency(), and ~DramPerfModelNormal().

6.110.4.2 m_dram_bandwidth

```
ComponentBandwidth DramPerfModelNormal::m_dram_bandwidth  [private]
```

Definition at line 16 of file dram_perf_model_normal.h.

Referenced by DramPerfModelNormal(), and getAccessLatency().

6.110.4.3 m_queue_model

```
QueueModel* DramPerfModelNormal::m_queue_model  [private]
```

Definition at line 14 of file dram_perf_model_normal.h.

Referenced by DramPerfModelNormal(), getAccessLatency(), and ~DramPerfModelNormal().

6.110.4.4 m_total_access_latency

```
SubsecondTime DramPerfModelNormal::m_total_access_latency  [private]
```

Definition at line 19 of file dram_perf_model_normal.h.

Referenced by DramPerfModelNormal(), and getAccessLatency().

6.110.4.5 m_total_queueing_delay

```
SubsecondTime DramPerfModelNormal::m_total_queueing_delay  [private]
```

Definition at line 18 of file dram_perf_model_normal.h.

Referenced by DramPerfModelNormal(), and getAccessLatency().

The documentation for this class was generated from the following files:

- common/performance_model/ **dram_perf_model_normal.h**
- common/performance_model/ **dram_perf_model_normal.cc**

6.111 DramPerfModelReadWrite Class Reference

`#include <dram_perf_model_readwrite.h>`

Inheritance diagram for DramPerfModelReadWrite:

Public Member Functions

- **DramPerfModelReadWrite** (**core_id_t** core_id, **UInt32** cache_block_size)
- ~**DramPerfModelReadWrite** ()
- **SubsecondTime getAccessLatency** (**SubsecondTime** pkt_time, **UInt64** pkt_size, **core_id_t** requester, **IntPtr** address, **DramCntlrInterface::access_t** access_type, **ShmemPerf** *perf)

Private Attributes

- **QueueModel** * **m_queue_model_read**
- **QueueModel** * **m_queue_model_write**
- **SubsecondTime** **m_dram_access_cost**
- **ComponentBandwidth** **m_dram_bandwidth**
- bool **m_shared_readwrite**
- **SubsecondTime** **m_total_read_queueing_delay**
- **SubsecondTime** **m_total_write_queueing_delay**
- **SubsecondTime** **m_total_access_latency**

Additional Inherited Members

6.111.1 Detailed Description

Definition at line 10 of file dram_perf_model_readwrite.h.

6.111.2 Constructor & Destructor Documentation

6.111.2.1 DramPerfModelReadWrite()

```
DramPerfModelReadWrite::DramPerfModelReadWrite (
            core_id_t core_id,
            UInt32 cache_block_size )
```

Definition at line 8 of file dram_perf_model_readwrite.cc.

References QueueModel::create(), SubsecondTime::FS(), ComponentBandwidth::getRoundedLatency(), m_↩ dram_access_cost, m_dram_bandwidth, m_queue_model_read, m_queue_model_write, m_total_access_latency, m_total_read_queueing_delay, m_total_write_queueing_delay, TimeConverter< T >::NStoFS(), and register↩ StatsMetric().

6.111.2.2 ~DramPerfModelReadWrite()

```
DramPerfModelReadWrite::~DramPerfModelReadWrite ( )
```

Definition at line 34 of file dram_perf_model_readwrite.cc.

References m_queue_model_read, and m_queue_model_write.

6.111.3 Member Function Documentation

6.111.3.1 getAccessLatency()

```
SubsecondTime DramPerfModelReadWrite::getAccessLatency (
            SubsecondTime pkt_time,
            UInt64 pkt_size,
            core_id_t requester,
            IntPtr address,
            DramCntlrInterface::access_t access_type,
            ShmemPerf * perf )  [virtual]
```

Implements **DramPerfModel** (p. 499).

Definition at line 46 of file dram_perf_model_readwrite.cc.

References QueueModel::computeQueueDelay(), ShmemPerf::DRAM_BUS, ShmemPerf::DRAM_DEVICE, ShmemPerf::DRAM_QUEUE, ComponentBandwidth::getRoundedLatency(), Config::getSingleton(), m_dram_↩ access_cost, m_dram_bandwidth, DramPerfModel::m_enabled, DramPerfModel::m_num_accesses, m_queue↩ _model_read, m_queue_model_write, m_shared_readwrite, m_total_access_latency, m_total_read_queueing_↩ delay, m_total_write_queueing_delay, DramCntlrInterface::READ, ShmemPerf::updateTime(), and Subsecond↩ Time::Zero().

6.111.4 Member Data Documentation

6.111.4.1 m_dram_access_cost

```
SubsecondTime DramPerfModelReadWrite::m_dram_access_cost  [private]
```

Definition at line 15 of file dram_perf_model_readwrite.h.

Referenced by DramPerfModelReadWrite(), and getAccessLatency().

6.111.4.2 m_dram_bandwidth

```
ComponentBandwidth DramPerfModelReadWrite::m_dram_bandwidth  [private]
```

Definition at line 16 of file dram_perf_model_readwrite.h.

Referenced by DramPerfModelReadWrite(), and getAccessLatency().

6.111.4.3 m_queue_model_read

```
QueueModel* DramPerfModelReadWrite::m_queue_model_read  [private]
```

Definition at line 13 of file dram_perf_model_readwrite.h.

Referenced by DramPerfModelReadWrite(), getAccessLatency(), and ~DramPerfModelReadWrite().

6.111.4.4 m_queue_model_write

```
QueueModel* DramPerfModelReadWrite::m_queue_model_write [private]
```

Definition at line 14 of file dram_perf_model_readwrite.h.

Referenced by DramPerfModelReadWrite(), getAccessLatency(), and ~DramPerfModelReadWrite().

6.111.4.5 m_shared_readwrite

```
bool DramPerfModelReadWrite::m_shared_readwrite [private]
```

Definition at line 17 of file dram_perf_model_readwrite.h.

Referenced by getAccessLatency().

6.111.4.6 m_total_access_latency

```
SubsecondTime DramPerfModelReadWrite::m_total_access_latency [private]
```

Definition at line 21 of file dram_perf_model_readwrite.h.

Referenced by DramPerfModelReadWrite(), and getAccessLatency().

6.111.4.7 m_total_read_queueing_delay

```
SubsecondTime DramPerfModelReadWrite::m_total_read_queueing_delay [private]
```

Definition at line 19 of file dram_perf_model_readwrite.h.

Referenced by DramPerfModelReadWrite(), and getAccessLatency().

6.111.4.8 m_total_write_queueing_delay

```
SubsecondTime DramPerfModelReadWrite::m_total_write_queueing_delay [private]
```

Definition at line 20 of file dram_perf_model_readwrite.h.

Referenced by DramPerfModelReadWrite(), and getAccessLatency().

The documentation for this class was generated from the following files:

- common/performance_model/ **dram_perf_model_readwrite.h**
- common/performance_model/ **dram_perf_model_readwrite.cc**

6.112 DvfsManager Class Reference

`#include <dvfs_manager.h>`

Public Types

- enum **DvfsGlobalDomain** { **DOMAIN_GLOBAL_DEFAULT**, **DOMAIN_GLOBAL_MAX** }

Public Member Functions

- **DvfsManager** ()
- **UInt32** **getCoreDomainId** (**UInt32** core_id)
- const **ComponentPeriod** * **getCoreDomain** (**UInt32** core_id)
- const **ComponentPeriod** * **getGlobalDomain** (**DvfsGlobalDomain** domain_id= **DOMAIN_GLOBAL_DEFAULT**)

Protected Member Functions

- void **setCoreDomain** (**UInt32** core_id, **ComponentPeriod** new_freq)

Private Attributes

- **UInt32** **m_cores_per_socket**
- **SubsecondTime** **m_transition_latency**
- **UInt32** **m_num_proc_domains**
- **UInt32** **m_num_app_cores**
- std::vector< **ComponentPeriod** > **app_proc_domains**
- std::vector< **ComponentPeriod** > **global_domains**

Friends

- class **MagicServer**

6.112.1 Detailed Description

Definition at line 11 of file dvfs_manager.h.

6.112.2 Member Enumeration Documentation

6.112.2.1 DvfsGlobalDomain

enum **DvfsManager::DvfsGlobalDomain**

Enumerator

DOMAIN_GLOBAL_DEFAULT	
DOMAIN_GLOBAL_MAX	

Definition at line 14 of file dvfs_manager.h.

6.112.3 Constructor & Destructor Documentation

6.112.3.1 DvfsManager()

```
DvfsManager::DvfsManager ( )
```

Definition at line 13 of file dvfs_manager.cc.

References app_proc_domains, DOMAIN_GLOBAL_MAX, ComponentPeriod::fromFreqHz(), Config::getApplicationCores(), getCoreDomainId(), Config::getSingleton(), global_domains, LOG_ASSERT_ERROR, m_cores_per_socket, m_num_app_cores, m_num_proc_domains, m_transition_latency, and SubsecondTime::NS().

6.112.4 Member Function Documentation

6.112.4.1 getCoreDomain()

```
const ComponentPeriod * DvfsManager::getCoreDomain (
            UInt32 core_id )
```

Definition at line 59 of file dvfs_manager.cc.

References app_proc_domains, DOMAIN_GLOBAL_DEFAULT, getCoreDomainId(), global_domains, and m_num_app_cores.

6.112.4.2 getCoreDomainId()

```
UInt32 DvfsManager::getCoreDomainId (
            UInt32 core_id )
```

Definition at line 51 of file dvfs_manager.cc.

References LOG_ASSERT_ERROR, m_cores_per_socket, and m_num_app_cores.

Referenced by DvfsManager(), getCoreDomain(), and setCoreDomain().

6.112.4.3 getGlobalDomain()

```
const ComponentPeriod * DvfsManager::getGlobalDomain (
            DvfsGlobalDomain domain_id = DOMAIN_GLOBAL_DEFAULT )
```

Definition at line 72 of file dvfs_manager.cc.

References global_domains, and LOG_ASSERT_ERROR.

6.112.4.4 setCoreDomain()

```
void DvfsManager::setCoreDomain (
            UInt32 core_id,
            ComponentPeriod new_freq ) [protected]
```

Definition at line 80 of file dvfs_manager.cc.

References app_proc_domains, DelayInstruction::DVFS_TRANSITION, getCoreDomainId(), ComponentPeriod↩ ::getPeriod(), LOG_PRINT_ERROR, m_num_app_cores, and m_transition_latency.

6.112.5 Friends And Related Function Documentation

6.112.5.1 MagicServer

```
friend class MagicServer [friend]
```

Definition at line 26 of file dvfs_manager.h.

6.112.6 Member Data Documentation

6.112.6.1 app_proc_domains

```
std::vector< ComponentPeriod> DvfsManager::app_proc_domains [private]
```

Definition at line 32 of file dvfs_manager.h.

Referenced by DvfsManager(), getCoreDomain(), and setCoreDomain().

6.112.6.2 global_domains

```
std::vector< ComponentPeriod> DvfsManager::global_domains  [private]
```

Definition at line 33 of file dvfs_manager.h.

Referenced by DvfsManager(), getCoreDomain(), and getGlobalDomain().

6.112.6.3 m_cores_per_socket

```
UInt32 DvfsManager::m_cores_per_socket  [private]
```

Definition at line 28 of file dvfs_manager.h.

Referenced by DvfsManager(), and getCoreDomainId().

6.112.6.4 m_num_app_cores

```
UInt32 DvfsManager::m_num_app_cores  [private]
```

Definition at line 31 of file dvfs_manager.h.

Referenced by DvfsManager(), getCoreDomain(), getCoreDomainId(), and setCoreDomain().

6.112.6.5 m_num_proc_domains

```
UInt32 DvfsManager::m_num_proc_domains  [private]
```

Definition at line 30 of file dvfs_manager.h.

Referenced by DvfsManager().

6.112.6.6 m_transition_latency

```
SubsecondTime DvfsManager::m_transition_latency  [private]
```

Definition at line 29 of file dvfs_manager.h.

Referenced by DvfsManager(), and setCoreDomain().

The documentation for this class was generated from the following files:

- common/system/ **dvfs_manager.h**
- common/system/ **dvfs_manager.cc**

6.113 DynamicInstruction Class Reference

`#include <dynamic_instruction.h>`

Classes

- struct **BranchInfo**
- struct **MemoryInfo**

Public Member Functions

- ~**DynamicInstruction** ()
- **SubsecondTime** **getCost** (**Core** *core)
- bool **isBranch** () const
- bool **isMemory** () const
- void **addMemory** (bool e, **SubsecondTime** l, **IntPtr** a, **UInt32** s, **Operand::Direction** dir, **UInt32** num_misses, **HitWhere::where_t** hit_where)
- void **addBranch** (bool taken, **IntPtr** target)
- **SubsecondTime** **getBranchCost** (**Core** *core, bool *p_is_mispredict=NULL)
- void **accessMemory** (**Core** *core)

Static Public Member Functions

- static **Allocator** * **createAllocator** ()
- static **DynamicInstruction** * **alloc** (**Allocator** *alloc, **Instruction** *ins, **IntPtr** **eip**)
- static void **operator delete** (void *ptr)

Public Attributes

- **Instruction** * **instruction**
- **IntPtr** **eip**
- **BranchInfo** **branch_info**
- **UInt8** **num_memory**
- **MemoryInfo** **memory_info** [**MAX_MEMORY**]

Static Public Attributes

- static const **UInt8** **MAX_MEMORY** = 2

Private Member Functions

- **DynamicInstruction** (**Instruction** *ins, **IntPtr** _eip)

6.113.1 Detailed Description

Definition at line 12 of file dynamic_instruction.h.

6.113.2 Constructor & Destructor Documentation

6.113.2.1 DynamicInstruction()

```
DynamicInstruction::DynamicInstruction (
              Instruction * ins,
              IntPtr _eip )  [inline], [private]
```

Definition at line 16 of file dynamic_instruction.h.

References branch_info, eip, instruction, DynamicInstruction::BranchInfo::is_branch, and num_memory.

Referenced by alloc().

6.113.2.2 ~DynamicInstruction()

```
DynamicInstruction::~DynamicInstruction ( )
```

Definition at line 13 of file dynamic_instruction.cc.

References instruction, and Instruction::isPseudo().

6.113.3 Member Function Documentation

6.113.3.1 accessMemory()

```
void DynamicInstruction::accessMemory (
              Core * core )
```

Definition at line 42 of file dynamic_instruction.cc.

References Core::accessMemory(), DynamicInstruction::MemoryInfo::addr, Instruction::getAddress(), Core::getDvfsDomain(), ComponentPeriod::getPeriod(), MemoryResult::hit_where, DynamicInstruction::MemoryInfo::hit_where, instruction, Instruction::isAtomic(), MemoryResult::latency, DynamicInstruction::MemoryInfo::latency, Core::MEM_MODELED_RETURN, memory_info, Core::NONE, num_memory, HitWhere::PREDICATE_FALSE, Operand::READ, Core::READ, Core::READ_EX, DynamicInstruction::MemoryInfo::size, HitWhere::UNKNOWN, and Core::WRITE.

Referenced by OneIPCPerformanceModel::handleInstruction(), and MicroOpPerformanceModel::handleInstruction().

6.113.3.2 addBranch()

```
void DynamicInstruction::addBranch (
             bool taken,
              IntPtr target )  [inline]
```

Definition at line 77 of file dynamic_instruction.h.

References branch_info, DynamicInstruction::BranchInfo::is_branch, DynamicInstruction::BranchInfo::taken, and DynamicInstruction::BranchInfo::target.

Referenced by TraceThread::handleInstructionDetailed().

6.113.3.3 addMemory()

```
void DynamicInstruction::addMemory (
             bool e,
              SubsecondTime l,
              IntPtr a,
              UInt32 s,
              Operand::Direction dir,
              UInt32 num_misses,
              HitWhere::where_t hit_where )  [inline]
```

Definition at line 64 of file dynamic_instruction.h.

References DynamicInstruction::MemoryInfo::addr, DynamicInstruction::MemoryInfo::dir, DynamicInstruction↩ ::MemoryInfo::executed, DynamicInstruction::MemoryInfo::hit_where, DynamicInstruction::MemoryInfo::latency, LOG_ASSERT_ERROR, MAX_MEMORY, memory_info, num_memory, DynamicInstruction::MemoryInfo::num_↩ misses, and DynamicInstruction::MemoryInfo::size.

Referenced by TraceThread::addDetailedMemoryInfo().

6.113.3.4 alloc()

```
static  DynamicInstruction* DynamicInstruction::alloc (
              Allocator * alloc,
              Instruction * ins,
              IntPtr eip )  [inline], [static]
```

Definition at line 51 of file dynamic_instruction.h.

References alloc(), DynamicInstruction(), and eip.

Referenced by alloc(), and PerformanceModel::createDynamicInstruction().

6.113.3.5 createAllocator()

```
Allocator * DynamicInstruction::createAllocator ( )  [static]
```

Definition at line 8 of file dynamic_instruction.cc.

6.113.3.6 getBranchCost()

```
SubsecondTime DynamicInstruction::getBranchCost (
            Core * core,
           bool * p_is_mispredict = NULL )
```

Definition at line 27 of file dynamic_instruction.cc.

References Core::accessBranchPredictor(), branch_info, eip, PerformanceModel::getBranchPredictor(), Core::getDvfsDomain(), BranchPredictor::getMispredictPenalty(), Core::getPerformanceModel(), DynamicInstruction::BranchInfo::taken, and DynamicInstruction::BranchInfo::target.

Referenced by getCost(), and MicroOpPerformanceModel::handleInstruction().

6.113.3.7 getCost()

```
SubsecondTime DynamicInstruction::getCost (
            Core * core )
```

Definition at line 19 of file dynamic_instruction.cc.

References getBranchCost(), Instruction::getCost(), instruction, and isBranch().

Referenced by MicroOpPerformanceModel::handleInstruction().

6.113.3.8 isBranch()

```
bool DynamicInstruction::isBranch ( ) const  [inline]
```

Definition at line 61 of file dynamic_instruction.h.

References branch_info, and DynamicInstruction::BranchInfo::is_branch.

Referenced by getCost().

6.113.3.9 isMemory()

```
bool DynamicInstruction::isMemory ( ) const  [inline]
```

Definition at line 62 of file dynamic_instruction.h.

References num_memory.

6.113.3.10 operator delete()

```
static void DynamicInstruction::operator delete (
            void * ptr )  [inline], [static]
```

Definition at line 57 of file dynamic_instruction.h.

References Allocator::dealloc().

6.113.4 Member Data Documentation

6.113.4.1 branch_info

```
BranchInfo DynamicInstruction::branch_info
```

Definition at line 43 of file dynamic_instruction.h.

Referenced by addBranch(), DynamicInstruction(), getBranchCost(), MicroOpPerformanceModel::handle↩Instruction(), and isBranch().

6.113.4.2 eip

```
IntPtr DynamicInstruction::eip
```

Definition at line 42 of file dynamic_instruction.h.

Referenced by alloc(), DynamicInstruction(), getBranchCost(), and MicroOpPerformanceModel::handleInstruction().

6.113.4.3 instruction

```
Instruction* DynamicInstruction::instruction
```

Definition at line 41 of file dynamic_instruction.h.

Referenced by accessMemory(), DynamicInstruction(), getCost(), OneIPCPerformanceModel::handleInstruction(), MicroOpPerformanceModel::handleInstruction(), PerformanceModel::iterate(), and ~DynamicInstruction().

6.113.4.4 MAX_MEMORY

```
const UInt8 DynamicInstruction::MAX_MEMORY = 2 [static]
```

Definition at line 39 of file dynamic_instruction.h.

Referenced by addMemory().

6.113.4.5 memory_info

```
MemoryInfo DynamicInstruction::memory_info[ MAX_MEMORY]
```

Definition at line 45 of file dynamic_instruction.h.

Referenced by accessMemory(), addMemory(), OneIPCPerformanceModel::handleInstruction(), and MicroOp↩ PerformanceModel::handleInstruction().

6.113.4.6 num_memory

```
UInt8 DynamicInstruction::num_memory
```

Definition at line 44 of file dynamic_instruction.h.

Referenced by accessMemory(), addMemory(), DynamicInstruction(), OneIPCPerformanceModel::handle↩ Instruction(), MicroOpPerformanceModel::handleInstruction(), and isMemory().

The documentation for this class was generated from the following files:

- common/performance_model/ **dynamic_instruction.h**
- common/performance_model/ **dynamic_instruction.cc**

6.114 DynamicMicroOp Class Reference

`#include <dynamic_micro_op.h>`

Inheritance diagram for DynamicMicroOp:

Public Member Functions

- **DynamicMicroOp** (const **MicroOp** *uop, const **CoreModel** *core_model, **ComponentPeriod** period)
- virtual ~**DynamicMicroOp** ()
- const **MicroOp** * **getMicroOp** () const
- template<typename T >
  const T * **getCoreSpecificInfo** () const
- void **squash** (std::vector< **DynamicMicroOp** * > *array=NULL)
- bool **isSquashed** ()
- uint32_t **getDependenciesLength** () const
- uint64_t **getDependency** (uint32_t index) const
- void **addDependency** (uint64_t **sequenceNumber**)
- void **removeDependency** (uint64_t **sequenceNumber**)
- uint32_t **getIntraInstrDependenciesLength** () const
- void **setIntraInstrDependenciesLength** (uint32_t deps)
- uint32_t **getMicroOpTypeOffset** () const
- void **setMicroOpTypeOffset** (uint32_t offset)
- uint32_t **getSquashedCount** () const
- void **setSquashedCount** (uint32_t count)
- void **setFirst** (bool _first)
- bool **isFirst** () const
- void **setLast** (bool _last)
- bool **isLast** () const
- bool **isBranchTaken** () const
- void **setBranchTaken** (bool _branch_taken)
- bool **isBranchMispredicted** () const
- void **setBranchMispredicted** (bool mispredicted)
- **IntPtr** **getBranchTarget** () const
- void **setBranchTarget** (**IntPtr** **address**)
- const **Memory::Access** & **getLoadAccess** () const
- bool **isLongLatencyLoad** () const
- const **Memory::Access** & **getStoreAccess** () const
- uint32_t **getExecLatency** () const
- void **setExecLatency** (uint32_t latency)
- uint64_t **getSequenceNumber** () const
- void **setSequenceNumber** (uint64_t number)
- **HitWhere::where_t** **getDCacheHitWhere** () const
- void **setDCacheHitWhere** (**HitWhere::where_t** _hitWhere)
- **HitWhere::where_t** **getICacheHitWhere** () const
- void **setICacheHitWhere** (**HitWhere::where_t** _hitWhere)
- uint32_t **getICacheLatency** () const

- void **setICacheLatency** (uint32_t _latency)
- void **setAddress** (const **Memory::Access** &loadAccess)
- const **Memory::Access** & **getAddress** (void) const
- void **setForceLongLatencyLoad** (bool forceLLL)
- **SubsecondTime** **getPeriod** () const
- virtual const char * **getType** () const =0

Static Public Member Functions

- template<typename T >
  static T * **alloc** (**Allocator** *alloc, const **MicroOp** *uop, const **CoreModel** *core_model, **Component**↩ **Period** period)
- static void **operator delete** (void *ptr)

Private Attributes

- const **MicroOp** * **m_uop**
- const **CoreModel** * **m_core_model**
- const **SubsecondTime** **m_period**
- uint64_t **sequenceNumber**
- **Memory::Access** **address**
- bool **squashed**
- uint32_t **intraInstructionDependencies**
- uint32_t **microOpTypeOffset**
- uint32_t **squashedCount**
- uint32_t **dependenciesLength**
- uint64_t **dependencies** [**MAXIMUM_NUMBER_OF_DEPENDENCIES**]
- uint32_t **execLatency**
- bool **branchTaken**
- bool **branchMispredicted**
- **IntPtr** **branchTargetAddress**
- **HitWhere::where_t** **dCacheHitWhere**
- **HitWhere::where_t** **iCacheHitWhere**
- uint32_t **iCacheLatency**
- bool **m_forceLongLatencyLoad**
- bool **first**
- bool **last**

6.114.1 Detailed Description

Definition at line 14 of file dynamic_micro_op.h.

6.114.2 Constructor & Destructor Documentation

6.114.2.1 DynamicMicroOp()

```
DynamicMicroOp::DynamicMicroOp (
            const  MicroOp * uop,
            const  CoreModel * core_model,
             ComponentPeriod period )
```

Definition at line 5 of file dynamic_micro_op.cc.

References branchMispredicted, branchTaken, dCacheHitWhere, dependencies, dependenciesLength, exec↩ Latency, first, CoreModel::getInstructionLatency(), iCacheHitWhere, iCacheLatency, intraInstructionDependencies, MicroOp::intraInstructionDependencies, INVALID_SEQNR, MicroOp::isFirst(), MicroOp::isLast(), HitWhere::L1I, last, LOG_ASSERT_ERROR, m_core_model, m_forceLongLatencyLoad, m_uop, MAXIMUM_NUMBER_OF_D↩ EPENDENCIES, microOpTypeOffset, MicroOp::microOpTypeOffset, sequenceNumber, squashed, squashedCount, HitWhere::UNKNOWN, and SubsecondTime::Zero().

6.114.2.2 ~DynamicMicroOp()

```
DynamicMicroOp::~DynamicMicroOp ( )  [virtual]
```

Definition at line 41 of file dynamic_micro_op.cc.

6.114.3 Member Function Documentation

6.114.3.1 addDependency()

```
void DynamicMicroOp::addDependency (
            uint64_t sequenceNumber )
```

Definition at line 77 of file dynamic_micro_op.cc.

References Tools::contains(), dependencies, dependenciesLength, MAXIMUM_NUMBER_OF_DEPENDENCIES, and sequenceNumber.

Referenced by RegisterDependencies::setDependencies(), MemoryDependencies::setDependencies(), RobSmt↩ Timer::setDependencies(), and RobTimer::simulate().

6.114.3.2 alloc()

```
template<typename T >
static T* DynamicMicroOp::alloc (
             Allocator * alloc,
            const  MicroOp * uop,
            const  CoreModel * core_model,
             ComponentPeriod period )  [inline], [static]
```

Definition at line 74 of file dynamic_micro_op.h.

6.114.3.3 getAddress()

```
const Memory::Access& DynamicMicroOp::getAddress (
            void ) const [inline]
```

Definition at line 142 of file dynamic_micro_op.h.

References address.

Referenced by RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), IntervalTimer::issueMemOp(), RobTimer::printRob(), and RobSmtTimer::printRob().

6.114.3.4 getBranchTarget()

```
IntPtr DynamicMicroOp::getBranchTarget ( ) const [inline]
```

Definition at line 118 of file dynamic_micro_op.h.

References branchTargetAddress, MicroOp::isBranch(), LOG_ASSERT_ERROR, and m_uop.

6.114.3.5 getCoreSpecificInfo()

```
template<typename T >
const T* DynamicMicroOp::getCoreSpecificInfo ( ) const [inline]
```

Definition at line 84 of file dynamic_micro_op.h.

References getType(), and LOG_ASSERT_ERROR.

Referenced by IntervalContentionBoomV1::addFunctionalUnitStats(), IntervalContentionNehalem::addFunctionalUnitStats(), RobContentionBoomV1::doIssue(), RobContentionNehalem::doIssue(), CoreModelNehalem::getBypassLatency(), CoreModelBoomV1::getBypassLatency(), IntervalContentionBoomV1::removeFunctionalUnitStats(), IntervalContentionNehalem::removeFunctionalUnitStats(), RobContentionBoomV1::tryIssue(), and RobContentionNehalem::tryIssue().

6.114.3.6 getDCacheHitWhere()

```
HitWhere::where_t DynamicMicroOp::getDCacheHitWhere ( ) const [inline]
```

Definition at line 131 of file dynamic_micro_op.h.

References dCacheHitWhere.

Referenced by RobTimer::countOutstandingMemop(), RobSmtTimer::countOutstandingMemop(), IntervalTimer::dispatchInstruction(), RobTimer::doIssue(), RobTimer::findCpiComponent(), RobSmtTimer::findCpiComponent(), getCpContrType(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), IntervalTimer::issueMemOp(), and RobSmtTimer::tryIssue().

6.114.3.7 getDependenciesLength()

```
uint32_t DynamicMicroOp::getDependenciesLength ( ) const  [inline]
```

Definition at line 94 of file dynamic_micro_op.h.

References dependenciesLength.

Referenced by RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), RobTimer::printRob(), RobSmtTimer::printRob(), RobSmtTimer::setDependencies(), and RobTimer::simulate().

6.114.3.8 getDependency()

```
uint64_t DynamicMicroOp::getDependency (
            uint32_t index ) const
```

Definition at line 67 of file dynamic_micro_op.cc.

References dependencies, dependenciesLength, intraInstructionDependencies, microOpTypeOffset, and sequenceNumber.

Referenced by IntervalTimer::blockWindow(), Windows::calculateBranchResolutionLatency(), IntervalTimer::getMaxProducerExecTime(), RobTimer::printRob(), RobSmtTimer::printRob(), RobSmtTimer::setDependencies(), and RobTimer::simulate().

6.114.3.9 getExecLatency()

```
uint32_t DynamicMicroOp::getExecLatency ( ) const  [inline]
```

Definition at line 125 of file dynamic_micro_op.h.

References execLatency.

Referenced by IntervalTimer::blockWindow(), Windows::calculateBranchResolutionLatency(), IntervalTimer::dispatchInstruction(), IntervalTimer::getMaxProducerExecTime(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), IntervalTimer::issueMemOp(), RobTimer::printRob(), and RobSmtTimer::printRob().

6.114.3.10 getICacheHitWhere()

```
HitWhere::where_t DynamicMicroOp::getICacheHitWhere ( ) const  [inline]
```

Definition at line 134 of file dynamic_micro_op.h.

References iCacheHitWhere.

Referenced by IntervalTimer::dispatchInstruction(), RobTimer::doDispatch(), and RobSmtTimer::tryDispatch().

6.114.3.11 getICacheLatency()

```
uint32_t DynamicMicroOp::getICacheLatency ( ) const  [inline]
```

Definition at line 137 of file dynamic_micro_op.h.

References iCacheLatency.

Referenced by IntervalTimer::dispatchInstruction(), RobTimer::doDispatch(), and RobSmtTimer::tryDispatch().

6.114.3.12 getIntraInstrDependenciesLength()

```
uint32_t DynamicMicroOp::getIntraInstrDependenciesLength ( ) const  [inline]
```

Definition at line 99 of file dynamic_micro_op.h.

References intraInstructionDependencies.

Referenced by MicroOpPerformanceModel::doSquashing().

6.114.3.13 getLoadAccess()

```
const  Memory::Access & DynamicMicroOp::getLoadAccess ( ) const
```

Definition at line 116 of file dynamic_micro_op.cc.

References address, and getMicroOp().

Referenced by IntervalTimer::blockWindow(), and MemoryDependencies::setDependencies().

6.114.3.14 getMicroOp()

```
const  MicroOp* DynamicMicroOp::getMicroOp ( ) const  [inline]
```

Definition at line 82 of file dynamic_micro_op.h.

References m_uop.

Referenced by RobTimer::countOutstandingMemop(), RobSmtTimer::countOutstandingMemop(), RobTimer::doCommit(), RobSmtTimer::doCommit(), RobTimer::doDispatch(), RobContentionBoomV1::doIssue(), RobContentionNehalem::doIssue(), RobTimer::doIssue(), MicroOpPerformanceModel::doSquashing(), RobTimer::findCpiComponent(), RobSmtTimer::findCpiComponent(), getLoadAccess(), Windows::WindowEntry::getMicroOp(), getStoreAccess(), isLongLatencyLoad(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), RobTimer::printRob(), RobSmtTimer::printRob(), RobSmtTimer::pushInstructions(), RegisterDependencies::setDependencies(), MemoryDependencies::setDependencies(), RobSmtTimer::setDependencies(), RobSmtTimer::setStoreAddressProducers(), RobTimer::simulate(), InstructionTracerPrint::traceInstruction(), InstructionTracerFPStats::traceInstruction(), LoopProfiler::traceInstruction(), LoopTracer::traceInstruction(), RobSmtTimer::tryDispatch(), and RobSmtTimer::tryIssue().

6.114.3.15 getMicroOpTypeOffset()

```
uint32_t DynamicMicroOp::getMicroOpTypeOffset ( ) const  [inline]
```

Definition at line 102 of file dynamic_micro_op.h.

References microOpTypeOffset.

6.114.3.16 getPeriod()

```
SubsecondTime DynamicMicroOp::getPeriod ( ) const  [inline]
```

Definition at line 146 of file dynamic_micro_op.h.

References LOG_ASSERT_ERROR, m_period, and SubsecondTime::Zero().

Referenced by IntervalTimer::dispatchInstruction(), IntervalTimer::dispatchWindow(), and IntervalTimer::update↩
CriticalPath().

6.114.3.17 getSequenceNumber()

```
uint64_t DynamicMicroOp::getSequenceNumber ( ) const  [inline]
```

Definition at line 128 of file dynamic_micro_op.h.

References sequenceNumber.

Referenced by RobTimer::findEntryBySequenceNumber(), RobSmtTimer::findEntryBySequenceNumber(), Windows::WindowEntry::getSequenceNumber(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), RobTimer::printRob(), RobSmtTimer::printRob(), RegisterDependencies::setDependencies(), MemoryDependencies↩
::setDependencies(), RobSmtTimer::setDependencies(), and RobTimer::simulate().

6.114.3.18 getSquashedCount()

```
uint32_t DynamicMicroOp::getSquashedCount ( ) const  [inline]
```

Definition at line 105 of file dynamic_micro_op.h.

References squashedCount.

6.114.3.19 getStoreAccess()

```
const Memory::Access & DynamicMicroOp::getStoreAccess ( ) const
```

Definition at line 122 of file dynamic_micro_op.cc.

References address, and getMicroOp().

Referenced by MemoryDependencies::setDependencies().

6.114.3.20 getType()

```
virtual const char* DynamicMicroOp::getType ( ) const [pure virtual]
```

Implemented in **DynamicMicroOpNehalem** (p. 546), and **DynamicMicroOpBoomV1** (p. 541).

Referenced by getCoreSpecificInfo().

6.114.3.21 isBranchMispredicted()

```
bool DynamicMicroOp::isBranchMispredicted ( ) const [inline]
```

Definition at line 116 of file dynamic_micro_op.h.

References branchMispredicted, MicroOp::isBranch(), LOG_ASSERT_ERROR, and m_uop.

Referenced by IntervalTimer::dispatchInstruction(), RobTimer::doDispatch(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), and RobSmtTimer::tryDispatch().

6.114.3.22 isBranchTaken()

```
bool DynamicMicroOp::isBranchTaken ( ) const [inline]
```

Definition at line 114 of file dynamic_micro_op.h.

References branchTaken, MicroOp::isBranch(), LOG_ASSERT_ERROR, and m_uop.

6.114.3.23 isFirst()

```
bool DynamicMicroOp::isFirst ( ) const [inline]
```

Definition at line 109 of file dynamic_micro_op.h.

References first.

Referenced by LoopTracer::traceInstruction().

6.114.3.24 isLast()

```
bool DynamicMicroOp::isLast ( ) const  [inline]
```

Definition at line 112 of file dynamic_micro_op.h.

References last.

Referenced by IntervalTimer::dispatchWindow(), RobTimer::doCommit(), RobSmtTimer::doCommit(), RobTimer↩ ::doDispatch(), and RobSmtTimer::tryDispatch().

6.114.3.25 isLongLatencyLoad()

```
bool DynamicMicroOp::isLongLatencyLoad ( ) const
```

Definition at line 128 of file dynamic_micro_op.cc.

References execLatency, CoreModel::getLongLatencyCutoff(), getMicroOp(), LOG_ASSERT_ERROR, m_core_↩ model, and m_forceLongLatencyLoad.

Referenced by IntervalTimer::blockWindow(), IntervalTimer::dispatchInstruction(), RobTimer::doIssue(), and Rob↩ SmtTimer::tryIssue().

6.114.3.26 isSquashed()

```
bool DynamicMicroOp::isSquashed ( )  [inline]
```

Definition at line 92 of file dynamic_micro_op.h.

References squashed.

Referenced by MicroOpPerformanceModel::doSquashing().

6.114.3.27 operator delete()

```
static void DynamicMicroOp::operator delete (
             void * ptr )  [inline], [static]
```

Definition at line 80 of file dynamic_micro_op.h.

References Allocator::dealloc().

6.114.3.28 removeDependency()

```
void DynamicMicroOp::removeDependency (
            uint64_t sequenceNumber )
```

Definition at line 86 of file dynamic_micro_op.cc.

References dependencies, dependenciesLength, Tools::index(), intraInstructionDependencies, LOG_ASSERT_↩ ERROR, MAXIMUM_NUMBER_OF_DEPENDENCIES, microOpTypeOffset, sequenceNumber, and Tools::swap().

Referenced by RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), RobSmtTimer::setDependencies(), and RobTimer::simulate().

6.114.3.29 setAddress()

```
void DynamicMicroOp::setAddress (
            const  Memory::Access & loadAccess )  [inline]
```

Definition at line 141 of file dynamic_micro_op.h.

6.114.3.30 setBranchMispredicted()

```
void DynamicMicroOp::setBranchMispredicted (
            bool mispredicted )  [inline]
```

Definition at line 117 of file dynamic_micro_op.h.

6.114.3.31 setBranchTaken()

```
void DynamicMicroOp::setBranchTaken (
            bool _branch_taken )  [inline]
```

Definition at line 115 of file dynamic_micro_op.h.

References branchTaken, MicroOp::isBranch(), LOG_ASSERT_ERROR, and m_uop.

6.114.3.32 setBranchTarget()

```
void DynamicMicroOp::setBranchTarget (
             IntPtr address )  [inline]
```

Definition at line 119 of file dynamic_micro_op.h.

References address.

6.114.3.33 setDCacheHitWhere()

```
void DynamicMicroOp::setDCacheHitWhere (
              HitWhere::where_t _hitWhere )  [inline]
```

Definition at line 132 of file dynamic_micro_op.h.

References dCacheHitWhere.

Referenced by RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), and IntervalTimer::issueMemOp().

6.114.3.34 setExecLatency()

```
void DynamicMicroOp::setExecLatency (
             uint32_t latency )  [inline]
```

Definition at line 126 of file dynamic_micro_op.h.

Referenced by MicroOpPerformanceModel::handleInstruction(), RobTimer::issueInstruction(), RobSmtTimer↩
::issueInstruction(), and IntervalTimer::issueMemOp().

6.114.3.35 setFirst()

```
void DynamicMicroOp::setFirst (
             bool _first )  [inline]
```

Definition at line 108 of file dynamic_micro_op.h.

References first.

6.114.3.36 setForceLongLatencyLoad()

```
void DynamicMicroOp::setForceLongLatencyLoad (
             bool forceLLL )  [inline]
```

Definition at line 144 of file dynamic_micro_op.h.

References m_forceLongLatencyLoad.

Referenced by MicroOpPerformanceModel::handleInstruction().

6.114.3.37 setICacheHitWhere()

```
void DynamicMicroOp::setICacheHitWhere (
              HitWhere::where_t _hitWhere )  [inline]
```

Definition at line 135 of file dynamic_micro_op.h.

References iCacheHitWhere.

6.114.3.38 setICacheLatency()

```
void DynamicMicroOp::setICacheLatency (
             uint32_t _latency )  [inline]
```

Definition at line 138 of file dynamic_micro_op.h.

References iCacheLatency.

6.114.3.39 setIntraInstrDependenciesLength()

```
void DynamicMicroOp::setIntraInstrDependenciesLength (
             uint32_t deps )  [inline]
```

Definition at line 100 of file dynamic_micro_op.h.

References intraInstructionDependencies.

Referenced by MicroOpPerformanceModel::doSquashing().

6.114.3.40 setLast()

```
void DynamicMicroOp::setLast (
             bool _last )  [inline]
```

Definition at line 111 of file dynamic_micro_op.h.

References last.

6.114.3.41 setMicroOpTypeOffset()

```
void DynamicMicroOp::setMicroOpTypeOffset (
            uint32_t offset )  [inline]
```

Definition at line 103 of file dynamic_micro_op.h.

References microOpTypeOffset.

Referenced by MicroOpPerformanceModel::doSquashing().

6.114.3.42 setSequenceNumber()

```
void DynamicMicroOp::setSequenceNumber (
            uint64_t number )  [inline]
```

Definition at line 129 of file dynamic_micro_op.h.

Referenced by Windows::add(), RobTimer::RobEntry::init(), and RobSmtTimer::RobEntry::init().

6.114.3.43 setSquashedCount()

```
void DynamicMicroOp::setSquashedCount (
            uint32_t count )  [inline]
```

Definition at line 106 of file dynamic_micro_op.h.

References squashedCount.

Referenced by MicroOpPerformanceModel::doSquashing().

6.114.3.44 squash()

```
void DynamicMicroOp::squash (
            std::vector<  DynamicMicroOp * > * array = NULL )
```

Definition at line 45 of file dynamic_micro_op.cc.

References squashed.

6.114.4 Member Data Documentation

6.114.4.1 address

Memory::Access DynamicMicroOp::address [private]

The address is valid for UOP_LOAD and UOP_STORE, it contains the load or store address.

Definition at line 27 of file dynamic_micro_op.h.

Referenced by getAddress(), getLoadAccess(), getStoreAccess(), and setBranchTarget().

6.114.4.2 branchMispredicted

bool DynamicMicroOp::branchMispredicted [private]

Is branch mispredicted ? Only for UOP_EXECUTE and branches.

Definition at line 49 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), and isBranchMispredicted().

6.114.4.3 branchTaken

bool DynamicMicroOp::branchTaken [private]

Did a jump occur after this instruction ?

Definition at line 47 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), isBranchTaken(), and setBranchTaken().

6.114.4.4 branchTargetAddress

IntPtr DynamicMicroOp::branchTargetAddress [private]

Branch target address

Definition at line 51 of file dynamic_micro_op.h.

Referenced by getBranchTarget().

6.114.4.5 dCacheHitWhere

```
HitWhere::where_t DynamicMicroOp::dCacheHitWhere  [private]
```

Definition at line 53 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), getDCacheHitWhere(), and setDCacheHitWhere().

6.114.4.6 dependencies

```
uint64_t DynamicMicroOp::dependencies[ MAXIMUM_NUMBER_OF_DEPENDENCIES]  [private]
```

This array contains the dependencies. The uint64_t stored in the array is the sequenceNumber of the dependency.

Definition at line 41 of file dynamic_micro_op.h.

Referenced by addDependency(), DynamicMicroOp(), getDependency(), and removeDependency().

6.114.4.7 dependenciesLength

```
uint32_t DynamicMicroOp::dependenciesLength  [private]
```

This field contains the length of the dependencies array.

Definition at line 39 of file dynamic_micro_op.h.

Referenced by addDependency(), DynamicMicroOp(), getDependenciesLength(), getDependency(), and removeDependency().

6.114.4.8 execLatency

```
uint32_t DynamicMicroOp::execLatency  [private]
```

The latency of the instruction.

Definition at line 44 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), getExecLatency(), and isLongLatencyLoad().

6.114.4.9 first

```
bool DynamicMicroOp::first  [private]
```

These first/last flags are needed in case of squashing, as long as squashed uop doesn't go to rob and we can't use first/last flags of m_uop in such a cases. This microOp is the first microOp of the instruction.

Definition at line 62 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), isFirst(), and setFirst().

6.114.4.10 iCacheHitWhere

```
HitWhere::where_t DynamicMicroOp::iCacheHitWhere  [private]
```

Definition at line 54 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), getICacheHitWhere(), and setICacheHitWhere().

6.114.4.11 iCacheLatency

```
uint32_t DynamicMicroOp::iCacheLatency  [private]
```

Definition at line 55 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), getICacheLatency(), and setICacheLatency().

6.114.4.12 intraInstructionDependencies

```
uint32_t DynamicMicroOp::intraInstructionDependencies  [private]
```

Initially copied from **MicroOp** (p. 764), but can be changed by removeDependency

Definition at line 33 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), getDependency(), getIntraInstrDependenciesLength(), removeDependency(), and setIntraInstrDependenciesLength().

6.114.4.13 last

```
bool DynamicMicroOp::last  [private]
```

This microOp is the last microOp of the instruction.

Definition at line 64 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), isLast(), and setLast().

6.114.4.14 m_core_model

```
const  CoreModel* DynamicMicroOp::m_core_model  [private]
```

Definition at line 18 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), and isLongLatencyLoad().

6.114.4.15 m_forceLongLatencyLoad

```
bool DynamicMicroOp::m_forceLongLatencyLoad  [private]
```

Definition at line 57 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), isLongLatencyLoad(), and setForceLongLatencyLoad().

6.114.4.16 m_period

```
const  SubsecondTime DynamicMicroOp::m_period  [private]
```

Definition at line 21 of file dynamic_micro_op.h.

Referenced by getPeriod().

6.114.4.17 m_uop

```
const  MicroOp* DynamicMicroOp::m_uop  [private]
```

Definition at line 17 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), getBranchTarget(), getMicroOp(), isBranchMispredicted(), isBranchTaken(), and setBranchTaken().

6.114.4.18 microOpTypeOffset

```
uint32_t DynamicMicroOp::microOpTypeOffset  [private]
```

Initially copied from **MicroOp** (p. 764), but can be changed in case of squashing

Definition at line 35 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), getDependency(), getMicroOpTypeOffset(), removeDependency(), and set↩ MicroOpTypeOffset().

6.114.4.19 sequenceNumber

```
uint64_t DynamicMicroOp::sequenceNumber  [private]
```

The sequence number of the microOperation. Unique (per thread) !

Definition at line 24 of file dynamic_micro_op.h.

Referenced by addDependency(), DynamicMicroOp(), getDependency(), getSequenceNumber(), and remove↩ Dependency().

6.114.4.20 squashed

```
bool DynamicMicroOp::squashed  [private]
```

The microop has been squashed

Definition at line 30 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), isSquashed(), and squash().

6.114.4.21 squashedCount

```
uint32_t DynamicMicroOp::squashedCount  [private]
```

Used in doSquashing. Contains number of squashed preceding uops of the instruction

Definition at line 37 of file dynamic_micro_op.h.

Referenced by DynamicMicroOp(), getSquashedCount(), and setSquashedCount().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/micro_op/ **dynamic_micro_op.h**
- common/performance_model/performance_models/micro_op/ **dynamic_micro_op.cc**

6.115 DynamicMicroOpBoomV1 Class Reference

`#include <dynamic_micro_op_boom_v1.h>`

Inheritance diagram for DynamicMicroOpBoomV1:

Public Types

- enum **uop_port_t** { **UOP_PORT0**, **UOP_PORT1**, **UOP_PORT2**, **UOP_PORT012**, **UOP_PORT_SIZE** }
- enum **uop_alu_t** { **UOP_ALU_NONE** = 0, **UOP_ALU_TRIG**, **UOP_ALU_SIZE** }
- enum **uop_bypass_t** { **UOP_BYPASS_NONE**, **UOP_BYPASS_LOAD_FP**, **UOP_BYPASS_FP_STORE**, **UOP_BYPASS_SIZE** }

Public Member Functions

- virtual const char * **getType** () const
- **DynamicMicroOpBoomV1** (const **MicroOp** *uop, const **CoreModel** *core_model, **ComponentPeriod** period)
- **uop_port_t** **getPort** (void) const
- **uop_bypass_t** **getBypassType** (void) const
- **uop_alu_t** **getAlu** (void) const

Static Public Member Functions

- static **uop_port_t** **getPort** (const **MicroOp** *uop)
- static **uop_bypass_t** **getBypassType** (const **MicroOp** *uop)
- static **uop_alu_t** **getAlu** (const **MicroOp** *uop)
- static const char * **PortTypeString** (**DynamicMicroOpBoomV1::uop_port_t** port)

Public Attributes

- **uop_port_t** **uop_port**
- **uop_alu_t** **uop_alu**
- **uop_bypass_t** **uop_bypass**

6.115.1 Detailed Description

Definition at line 8 of file dynamic_micro_op_boom_v1.h.

6.115.2 Member Enumeration Documentation

6.115.2.1 uop_alu_t

enum **DynamicMicroOpBoomV1::uop_alu_t**

Enumerator

UOP_ALU_NONE	
UOP_ALU_TRIG	
UOP_ALU_SIZE	

Definition at line 20 of file dynamic_micro_op_boom_v1.h.

6.115.2.2 uop_bypass_t

enum **DynamicMicroOpBoomV1::uop_bypass_t**

Enumerator

UOP_BYPASS_NONE	
UOP_BYPASS_LOAD_FP	
UOP_BYPASS_FP_STORE	
UOP_BYPASS_SIZE	

Definition at line 27 of file dynamic_micro_op_boom_v1.h.

6.115.2.3 uop_port_t

enum **DynamicMicroOpBoomV1::uop_port_t**

Enumerator

UOP_PORT0	
UOP_PORT1	
UOP_PORT2	
UOP_PORT012	
UOP_PORT_SIZE	

Definition at line 11 of file dynamic_micro_op_boom_v1.h.

6.115.3 Constructor & Destructor Documentation

6.115.3.1 DynamicMicroOpBoomV1()

```
DynamicMicroOpBoomV1::DynamicMicroOpBoomV1 (
            const  MicroOp * uop,
```

```
            const  CoreModel * core_model,
             ComponentPeriod period )
```

Definition at line 83 of file dynamic_micro_op_boom_v1.cc.

6.115.4 Member Function Documentation

6.115.4.1 getAlu() [1/2]

```
DynamicMicroOpBoomV1::uop_alu_t DynamicMicroOpBoomV1::getAlu (
            const  MicroOp * uop )  [static]
```

Definition at line 55 of file dynamic_micro_op_boom_v1.cc.

References MicroOp::getInstructionOpcode(), rv_op_div, rv_op_divd, rv_op_divu, rv_op_divud, rv_op_divuw, rv_op_divw, rv_op_fdiv_d, rv_op_fdiv_q, rv_op_fdiv_s, rv_op_fsqrt_d, rv_op_fsqrt_q, rv_op_fsqrt_s, UOP_ALU_NONE, UOP_ALU_TRIG, MicroOp::UOP_EXECUTE, and MicroOp::uop_type.

Referenced by RobContentionBoomV1::doIssue(), and RobContentionBoomV1::tryIssue().

6.115.4.2 getAlu() [2/2]

```
uop_alu_t DynamicMicroOpBoomV1::getAlu (
            void  ) const  [inline]
```

Definition at line 45 of file dynamic_micro_op_boom_v1.h.

References uop_alu.

Referenced by CoreModelBoomV1::createDynamicMicroOp().

6.115.4.3 getBypassType() [1/2]

```
DynamicMicroOpBoomV1::uop_bypass_t DynamicMicroOpBoomV1::getBypassType (
            const  MicroOp * uop )  [static]
```

Definition at line 37 of file dynamic_micro_op_boom_v1.cc.

References MicroOp::getSubtype(), MicroOp::isFpLoadStore(), UOP_BYPASS_FP_STORE, UOP_BYPASS_LOAD_FP, UOP_BYPASS_NONE, MicroOp::UOP_SUBTYPE_LOAD, and MicroOp::UOP_SUBTYPE_STORE.

Referenced by CoreModelBoomV1::getBypassLatency().

6.115.4.4 getBypassType() [2/2]

```
uop_bypass_t DynamicMicroOpBoomV1::getBypassType (
            void ) const [inline]
```

Definition at line 44 of file dynamic_micro_op_boom_v1.h.

References uop_bypass.

Referenced by CoreModelBoomV1::createDynamicMicroOp().

6.115.4.5 getPort() [1/2]

```
DynamicMicroOpBoomV1::uop_port_t DynamicMicroOpBoomV1::getPort (
            const MicroOp * uop ) [static]
```

Definition at line 24 of file dynamic_micro_op_boom_v1.cc.

References MicroOp::getInstructionOpcode(), riscvinstr::has_div, riscvinstr::has_fdiv, riscvinstr::has_fpu, riscvinstr::has_mul, instrlist, riscvinstr::is_memory, UOP_PORT0, UOP_PORT012, UOP_PORT1, and UOP_↩ PORT2.

Referenced by RobContentionBoomV1::tryIssue().

6.115.4.6 getPort() [2/2]

```
uop_port_t DynamicMicroOpBoomV1::getPort (
            void ) const [inline]
```

Definition at line 43 of file dynamic_micro_op_boom_v1.h.

References uop_port.

Referenced by CoreModelBoomV1::createDynamicMicroOp().

6.115.4.7 getType()

```
const char * DynamicMicroOpBoomV1::getType ( ) const [virtual]
```

Implements **DynamicMicroOp** (p. 527).

Definition at line 6 of file dynamic_micro_op_boom_v1.cc.

6.115.4.8 PortTypeString()

```
const char * DynamicMicroOpBoomV1::PortTypeString (
             DynamicMicroOpBoomV1::uop_port_t port )  [static]
```

Definition at line 11 of file dynamic_micro_op_boom_v1.cc.

References LOG_PRINT_ERROR, UOP_PORT0, UOP_PORT012, UOP_PORT1, and UOP_PORT2.

Referenced by IntervalContentionBoomV1::IntervalContentionBoomV1().

6.115.5 Member Data Documentation

6.115.5.1 uop_alu

```
uop_alu_t DynamicMicroOpBoomV1::uop_alu
```

Definition at line 25 of file dynamic_micro_op_boom_v1.h.

Referenced by CoreModelBoomV1::createDynamicMicroOp(), and getAlu().

6.115.5.2 uop_bypass

```
uop_bypass_t DynamicMicroOpBoomV1::uop_bypass
```

Definition at line 33 of file dynamic_micro_op_boom_v1.h.

Referenced by CoreModelBoomV1::createDynamicMicroOp(), and getBypassType().

6.115.5.3 uop_port

```
uop_port_t DynamicMicroOpBoomV1::uop_port
```

Definition at line 18 of file dynamic_micro_op_boom_v1.h.

Referenced by CoreModelBoomV1::createDynamicMicroOp(), and getPort().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/core_model/ **dynamic_micro_op_boom_v1.h**
- common/performance_model/performance_models/core_model/ **dynamic_micro_op_boom_v1.cc**

6.116 DynamicMicroOpNehalem Class Reference

`#include <dynamic_micro_op_nehalem.h>`

Inheritance diagram for DynamicMicroOpNehalem:

Public Types

- enum **uop_port_t** {
  UOP_PORT0, **UOP_PORT1**, **UOP_PORT2**, **UOP_PORT34**,
  UOP_PORT5, **UOP_PORT05**, **UOP_PORT015**, **UOP_PORT_SIZE** }
- enum **uop_alu_t** { **UOP_ALU_NONE** = 0, **UOP_ALU_TRIG**, **UOP_ALU_SIZE** }
- enum **uop_bypass_t** { **UOP_BYPASS_NONE**, **UOP_BYPASS_LOAD_FP**, **UOP_BYPASS_FP_STORE**, **UOP_BYPASS_SIZE** }

Public Member Functions

- virtual const char * **getType** () const
- **DynamicMicroOpNehalem** (const **MicroOp** *uop, const **CoreModel** *core_model, **ComponentPeriod** period)
- **uop_port_t** **getPort** (void) const
- **uop_bypass_t** **getBypassType** (void) const
- **uop_alu_t** **getAlu** (void) const

Static Public Member Functions

- static **uop_port_t** **getPort** (const **MicroOp** *uop)
- static **uop_bypass_t** **getBypassType** (const **MicroOp** *uop)
- static **uop_alu_t** **getAlu** (const **MicroOp** *uop)
- static const char * **PortTypeString** (**DynamicMicroOpNehalem::uop_port_t** port)

Public Attributes

- **uop_port_t** **uop_port**
- **uop_alu_t** **uop_alu**
- **uop_bypass_t** **uop_bypass**

6.116.1 Detailed Description

Definition at line 12 of file dynamic_micro_op_nehalem.h.

6.116.2 Member Enumeration Documentation

6.116.2.1 uop_alu_t

enum **DynamicMicroOpNehalem::uop_alu_t**

Enumerator

UOP_ALU_NONE	
UOP_ALU_TRIG	
UOP_ALU_SIZE	

Definition at line 27 of file dynamic_micro_op_nehalem.h.

6.116.2.2 uop_bypass_t

enum **DynamicMicroOpNehalem::uop_bypass_t**

Enumerator

UOP_BYPASS_NONE	
UOP_BYPASS_LOAD_FP	
UOP_BYPASS_FP_STORE	
UOP_BYPASS_SIZE	

Definition at line 34 of file dynamic_micro_op_nehalem.h.

6.116.2.3 uop_port_t

enum **DynamicMicroOpNehalem::uop_port_t**

Enumerator

UOP_PORT0	
UOP_PORT1	
UOP_PORT2	
UOP_PORT34	
UOP_PORT5	
UOP_PORT05	
UOP_PORT015	
UOP_PORT_SIZE	

Definition at line 15 of file dynamic_micro_op_nehalem.h.

6.116.3 Constructor & Destructor Documentation

6.116.3.1 DynamicMicroOpNehalem()

```
DynamicMicroOpNehalem::DynamicMicroOpNehalem (
            const  MicroOp * uop,
            const  CoreModel * core_model,
             ComponentPeriod period )
```

Definition at line 204 of file dynamic_micro_op_nehalem.cc.

6.116.4 Member Function Documentation

6.116.4.1 getAlu() [1/2]

```
DynamicMicroOpNehalem::uop_alu_t DynamicMicroOpNehalem::getAlu (
            const  MicroOp * uop )  [static]
```

Definition at line 161 of file dynamic_micro_op_nehalem.cc.

References MicroOp::getInstructionOpcode(), UOP_ALU_NONE, UOP_ALU_TRIG, MicroOp::UOP_EXECUTE, and MicroOp::uop_type.

Referenced by RobContentionNehalem::doIssue(), and RobContentionNehalem::tryIssue().

6.116.4.2 getAlu() [2/2]

```
uop_alu_t DynamicMicroOpNehalem::getAlu (
            void  ) const  [inline]
```

Definition at line 52 of file dynamic_micro_op_nehalem.h.

References uop_alu.

Referenced by CoreModelNehalem::createDynamicMicroOp().

6.116.4.3 getBypassType() [1/2]

```
DynamicMicroOpNehalem::uop_bypass_t DynamicMicroOpNehalem::getBypassType (
            const  MicroOp * uop )  [static]
```

Definition at line 143 of file dynamic_micro_op_nehalem.cc.

References MicroOp::getSubtype(), MicroOp::isFpLoadStore(), UOP_BYPASS_FP_STORE, UOP_BYPASS_LO↩AD_FP, UOP_BYPASS_NONE, MicroOp::UOP_SUBTYPE_LOAD, and MicroOp::UOP_SUBTYPE_STORE.

Referenced by CoreModelNehalem::getBypassLatency().

6.116.4.4 getBypassType() [2/2]

```
uop_bypass_t DynamicMicroOpNehalem::getBypassType (
            void  ) const  [inline]
```

Definition at line 51 of file dynamic_micro_op_nehalem.h.

References uop_bypass.

Referenced by CoreModelNehalem::createDynamicMicroOp().

6.116.4.5 getPort() [1/2]

```
DynamicMicroOpNehalem::uop_port_t DynamicMicroOpNehalem::getPort (
            const  MicroOp * uop )  [static]
```

Definition at line 25 of file dynamic_micro_op_nehalem.cc.

References MicroOp::getInstructionOpcode(), MicroOp::getOperandSize(), LOG_PRINT_ERROR, UOP_PORT0, UOP_PORT015, UOP_PORT1, UOP_PORT2, UOP_PORT34, UOP_PORT5, MicroOp::uop_subtype, MicroOp↩ ::UOP_SUBTYPE_BRANCH, MicroOp::UOP_SUBTYPE_FP_ADDSUB, MicroOp::UOP_SUBTYPE_FP_MULDIV, MicroOp::UOP_SUBTYPE_GENERIC, MicroOp::UOP_SUBTYPE_LOAD, and MicroOp::UOP_SUBTYPE_STO↩ RE.

Referenced by RobContentionNehalem::tryIssue().

6.116.4.6 getPort() [2/2]

```
uop_port_t DynamicMicroOpNehalem::getPort (
            void  ) const  [inline]
```

Definition at line 50 of file dynamic_micro_op_nehalem.h.

References uop_port.

Referenced by CoreModelNehalem::createDynamicMicroOp().

6.116.4.7 getType()

```
const char * DynamicMicroOpNehalem::getType ( ) const  [virtual]
```

Implements **DynamicMicroOp** (p. 527).

Definition at line 4 of file dynamic_micro_op_nehalem.cc.

6.116.4.8 PortTypeString()

```
const char * DynamicMicroOpNehalem::PortTypeString (
            DynamicMicroOpNehalem::uop_port_t port )  [static]
```

Definition at line 9 of file dynamic_micro_op_nehalem.cc.

References LOG_PRINT_ERROR, UOP_PORT0, UOP_PORT015, UOP_PORT05, UOP_PORT1, UOP_PORT2, UOP_PORT34, and UOP_PORT5.

Referenced by IntervalContentionNehalem::IntervalContentionNehalem().

6.116.5 Member Data Documentation

6.116.5.1 uop_alu

```
uop_alu_t DynamicMicroOpNehalem::uop_alu
```

Definition at line 32 of file dynamic_micro_op_nehalem.h.

Referenced by CoreModelNehalem::createDynamicMicroOp(), and getAlu().

6.116.5.2 uop_bypass

```
uop_bypass_t DynamicMicroOpNehalem::uop_bypass
```

Definition at line 40 of file dynamic_micro_op_nehalem.h.

Referenced by CoreModelNehalem::createDynamicMicroOp(), and getBypassType().

6.116.5.3 uop_port

```
uop_port_t DynamicMicroOpNehalem::uop_port
```

Definition at line 25 of file dynamic_micro_op_nehalem.h.

Referenced by CoreModelNehalem::createDynamicMicroOp(), and getPort().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/core_model/ **dynamic_micro_op_nehalem.h**
- common/performance_model/performance_models/core_model/ **dynamic_micro_op_nehalem.cc**

6.117 CircularLog::event_t Struct Reference

Public Attributes

- **UInt64 time**
- const char * **type**
- const char * **msg**
- **UInt64 args** [6]

6.117.1 Detailed Description

Definition at line 35 of file circular_log.h.

6.117.2 Member Data Documentation

6.117.2.1 args

UInt64 CircularLog::event_t::args[6]

Definition at line 39 of file circular_log.h.

Referenced by CircularLog::writeEntry().

6.117.2.2 msg

const char* CircularLog::event_t::msg

Definition at line 38 of file circular_log.h.

Referenced by CircularLog::insert(), and CircularLog::writeEntry().

6.117.2.3 time

UInt64 CircularLog::event_t::time

Definition at line 36 of file circular_log.h.

Referenced by CircularLog::insert(), and CircularLog::writeEntry().

6.117.2.4 type

```
const char* CircularLog::event_t::type
```

Definition at line 37 of file circular_log.h.

Referenced by CircularLog::insert(), and CircularLog::writeEntry().

The documentation for this struct was generated from the following file:

- common/misc/ **circular_log.h**

6.118 FastForwardPerformanceManager Class Reference

```
#include <fastforward_performance_manager.h>
```

Public Member Functions

- **FastForwardPerformanceManager** ()
- void **enable** ()
- void **disable** ()

Static Public Member Functions

- static **FastForwardPerformanceManager** * **create** ()

Protected Member Functions

- void **recalibrateInstructionsCallback** (**core_id_t** core_id)

Private Member Functions

- void **instr_count** (**core_id_t** core_id)
- void **periodic** (**SubsecondTime** time)
- void **step** ()

Static Private Member Functions

- static **SInt64** **hook_instr_count** (**UInt64** self, **UInt64** core_id)
- static **SInt64** **hook_periodic** (**UInt64** self, **UInt64** time)

Private Attributes

- const **SubsecondTime** **m_sync_interval**
- bool **m_enabled**
- **SubsecondTime** **m_target_sync_time**

Friends

- class **FastforwardPerformanceModel**

6.118.1 Detailed Description

Definition at line 7 of file fastforward_performance_manager.h.

6.118.2 Constructor & Destructor Documentation

6.118.2.1 FastForwardPerformanceManager()

```
FastForwardPerformanceManager::FastForwardPerformanceManager ( )
```

Definition at line 23 of file fastforward_performance_manager.cc.

References HookType::HOOK_INSTR_COUNT, hook_instr_count(), HookType::HOOK_PERIODIC, and hook_↩periodic().

Referenced by create().

6.118.3 Member Function Documentation

6.118.3.1 create()

```
FastForwardPerformanceManager * FastForwardPerformanceManager::create ( )  [static]
```

Definition at line 11 of file fastforward_performance_manager.cc.

References FastForwardPerformanceManager(), and LOG_PRINT_ERROR.

Referenced by Simulator::start().

6.118.3.2 disable()

```
void FastForwardPerformanceManager::disable ( )
```

Definition at line 47 of file fastforward_performance_manager.cc.

References Core::disableInstructionsCallback(), Core::disablePerformanceModels(), PerformanceModel::get↩ElapsedTime(), Thread::getId(), Core::getPerformanceModel(), Core::getThread(), m_enabled, SubsecondTime↩::MaxTime(), PerformanceModel::setFastForward(), and SubsecondTime::Zero().

6.118.3.3 enable()

```
void FastForwardPerformanceManager::enable ( )
```

Definition at line 33 of file fastforward_performance_manager.cc.

References m_enabled, and step().

6.118.3.4 hook_instr_count()

```
static SInt64 FastForwardPerformanceManager::hook_instr_count (
              UInt64 self,
              UInt64 core_id ) [inline], [static], [private]
```

Definition at line 26 of file fastforward_performance_manager.h.

Referenced by FastForwardPerformanceManager().

6.118.3.5 hook_periodic()

```
static SInt64 FastForwardPerformanceManager::hook_periodic (
              UInt64 self,
              UInt64 time ) [inline], [static], [private]
```

Definition at line 27 of file fastforward_performance_manager.h.

Referenced by FastForwardPerformanceManager().

6.118.3.6 instr_count()

```
void FastForwardPerformanceManager::instr_count (
              core_id_t core_id ) [private]
```

Definition at line 81 of file fastforward_performance_manager.cc.

References m_enabled, and SubsecondTime::Zero().

6.118.3.7 periodic()

```
void FastForwardPerformanceManager::periodic (
              SubsecondTime time ) [private]
```

Definition at line 72 of file fastforward_performance_manager.cc.

References m_enabled, and step().

6.118.3.8 recalibrateInstructionsCallback()

```
void FastForwardPerformanceManager::recalibrateInstructionsCallback (
              core_id_t core_id )  [protected]
```

Definition at line 117 of file fastforward_performance_manager.cc.

References SubsecondTime::divideRounded(), PerformanceModel::getElapsedTime(), PerformanceModel::get↩ FastforwardPerformanceModel(), Core::getPerformanceModel(), m_target_sync_time, FastforwardPerformance↩ Model::setCurrentCPI(), and Core::setInstructionsCallback().

Referenced by step().

6.118.3.9 step()

```
void FastForwardPerformanceManager::step ( )  [private]
```

Definition at line 99 of file fastforward_performance_manager.cc.

References PerformanceModel::getElapsedTime(), m_sync_interval, m_target_sync_time, recalibrateInstructions↩ Callback(), PerformanceModel::setFastForward(), and SubsecondTime::Zero().

Referenced by enable(), and periodic().

6.118.4 Friends And Related Function Documentation

6.118.4.1 FastforwardPerformanceModel

```
friend class  FastforwardPerformanceModel  [friend]
```

Definition at line 17 of file fastforward_performance_manager.h.

6.118.5 Member Data Documentation

6.118.5.1 m_enabled

```
bool FastForwardPerformanceManager::m_enabled  [private]
```

Definition at line 23 of file fastforward_performance_manager.h.

Referenced by disable(), enable(), instr_count(), and periodic().

6.118.5.2 m_sync_interval

```
const SubsecondTime FastForwardPerformanceManager::m_sync_interval [private]
```

Definition at line 22 of file fastforward_performance_manager.h.

Referenced by step().

6.118.5.3 m_target_sync_time

```
SubsecondTime FastForwardPerformanceManager::m_target_sync_time [private]
```

Definition at line 24 of file fastforward_performance_manager.h.

Referenced by recalibrateInstructionsCallback(), and step().

The documentation for this class was generated from the following files:

- common/system/ **fastforward_performance_manager.h**
- common/system/ **fastforward_performance_manager.cc**

6.119 FastforwardPerformanceModel Class Reference

`#include <fastforward_performance_model.h>`

Public Member Functions

- **FastforwardPerformanceModel** (**Core** ∗core, **PerformanceModel** ∗perf)
- ∼**FastforwardPerformanceModel** ()
- **SubsecondTime getCurrentCPI** () const
- void **setCurrentCPI** (**SubsecondTime** cpi)
- void **incrementElapsedTime** (**SubsecondTime** latency)
- void **incrementElapsedTime** (**SubsecondTime** latency, **SubsecondTime** &cpiComponent)
- void **notifyElapsedTimeUpdate** ()
- void **countInstructions** (**IntPtr** address, **UInt32** count)
- void **handleMemoryLatency** (**SubsecondTime** latency, **HitWhere::where_t** hit_where)
- void **handleBranchMispredict** ()
- void **queuePseudoInstruction** (**PseudoInstruction** ∗i)
- **SubsecondTime getFastforwardedTime** (void) const

Private Attributes

- **Core** ∗ **m_core**
- **PerformanceModel** ∗ **m_perf**
- const bool **m_include_memory_latency**
- const bool **m_include_branch_mispredict**
- **ComponentLatency m_branch_misprediction_penalty**
- **SubsecondTime m_cpi**
- **SubsecondTime m_fastforwarded_time**
- **SubsecondTime m_cpiBase**
- **SubsecondTime m_cpiBranchPredictor**
- std::vector< **SubsecondTime** > **m_cpiDataCache**

6.119.1 Detailed Description

Definition at line 6 of file fastforward_performance_model.h.

6.119.2 Constructor & Destructor Documentation

6.119.2.1 FastforwardPerformanceModel()

```
FastforwardPerformanceModel::FastforwardPerformanceModel (
              Core * core,
              PerformanceModel * perf )
```

Definition at line 10 of file fastforward_performance_model.cc.

References Core::getId(), HitWhereIsValid(), HitWhereString(), m_cpiBase, m_cpiBranchPredictor, m_cpiDataCache, m_fastforwarded_time, HitWhere::NUM_HITWHERES, registerStatsMetric(), HitWhere::WHERE_FIRST, and SubsecondTime::Zero().

6.119.2.2 ~FastforwardPerformanceModel()

```
FastforwardPerformanceModel::~FastforwardPerformanceModel ( )  [inline]
```

Definition at line 24 of file fastforward_performance_model.h.

6.119.3 Member Function Documentation

6.119.3.1 countInstructions()

```
void FastforwardPerformanceModel::countInstructions (
              IntPtr address,
              UInt32 count )
```

Definition at line 60 of file fastforward_performance_model.cc.

References incrementElapsedTime(), m_cpi, and m_cpiBase.

Referenced by PerformanceModel::countInstructions().

6.119.3.2 getCurrentCPI()

```
SubsecondTime FastforwardPerformanceModel::getCurrentCPI ( ) const  [inline]
```

Definition at line 26 of file fastforward_performance_model.h.

References m_cpi.

Referenced by SamplingManager::recalibrateInstructionsCallback().

6.119.3.3 getFastforwardedTime()

```
SubsecondTime FastforwardPerformanceModel::getFastforwardedTime (
            void  ) const  [inline]
```

Definition at line 37 of file fastforward_performance_model.h.

References m_fastforwarded_time.

Referenced by MagicServer::disablePerformance().

6.119.3.4 handleBranchMispredict()

```
void FastforwardPerformanceModel::handleBranchMispredict ( )
```

Definition at line 73 of file fastforward_performance_model.cc.

References ComponentLatency::getLatency(), incrementElapsedTime(), m_branch_misprediction_penalty, m_↩cpiBranchPredictor, and m_include_branch_mispredict.

Referenced by PerformanceModel::handleBranchMispredict().

6.119.3.5 handleMemoryLatency()

```
void FastforwardPerformanceModel::handleMemoryLatency (
              SubsecondTime latency,
              HitWhere::where_t hit_where )
```

Definition at line 66 of file fastforward_performance_model.cc.

References incrementElapsedTime(), m_cpiDataCache, and m_include_memory_latency.

Referenced by PerformanceModel::handleMemoryLatency().

6.119.3.6 incrementElapsedTime() [1/2]

```
void FastforwardPerformanceModel::incrementElapsedTime (
            SubsecondTime latency )
```

Definition at line 37 of file fastforward_performance_model.cc.

References m_cpiBase.

Referenced by countInstructions(), SamplingManager::disableFastForward(), handleBranchMispredict(), handle↩MemoryLatency(), and queuePseudoInstruction().

6.119.3.7 incrementElapsedTime() [2/2]

```
void FastforwardPerformanceModel::incrementElapsedTime (
            SubsecondTime latency,
            SubsecondTime & cpiComponent )
```

Definition at line 43 of file fastforward_performance_model.cc.

References PerformanceModel::incrementElapsedTime(), m_fastforwarded_time, and m_perf.

6.119.3.8 notifyElapsedTimeUpdate()

```
void FastforwardPerformanceModel::notifyElapsedTimeUpdate ( )
```

Definition at line 51 of file fastforward_performance_model.cc.

References Core::getId(), and m_core.

Referenced by PerformanceModel::handleIdleInstruction(), PerformanceModel::incrementIdleElapsedTime(), and queuePseudoInstruction().

6.119.3.9 queuePseudoInstruction()

```
void FastforwardPerformanceModel::queuePseudoInstruction (
            PseudoInstruction * i )
```

Definition at line 80 of file fastforward_performance_model.cc.

References PseudoInstruction::getCost(), Instruction::getType(), incrementElapsedTime(), Instruction::isIdle(), L↩OG_ASSERT_ERROR, m_core, and notifyElapsedTimeUpdate().

Referenced by PerformanceModel::queuePseudoInstruction().

6.119.3.10 setCurrentCPI()

```
void FastforwardPerformanceModel::setCurrentCPI (
              SubsecondTime cpi )  [inline]
```

Definition at line 27 of file fastforward_performance_model.h.

References m_cpi.

Referenced by PeriodicSampling::callbackDetailed(), and FastForwardPerformanceManager::recalibrate↩ InstructionsCallback().

6.119.4 Member Data Documentation

6.119.4.1 m_branch_misprediction_penalty

```
ComponentLatency FastforwardPerformanceModel::m_branch_misprediction_penalty  [private]
```

Definition at line 13 of file fastforward_performance_model.h.

Referenced by handleBranchMispredict().

6.119.4.2 m_core

```
Core* FastforwardPerformanceModel::m_core  [private]
```

Definition at line 9 of file fastforward_performance_model.h.

Referenced by notifyElapsedTimeUpdate(), and queuePseudoInstruction().

6.119.4.3 m_cpi

```
SubsecondTime FastforwardPerformanceModel::m_cpi  [private]
```

Definition at line 15 of file fastforward_performance_model.h.

Referenced by countInstructions(), getCurrentCPI(), and setCurrentCPI().

6.119.4.4 m_cpiBase

SubsecondTime FastforwardPerformanceModel::m_cpiBase [private]

Definition at line 18 of file fastforward_performance_model.h.

Referenced by countInstructions(), FastforwardPerformanceModel(), and incrementElapsedTime().

6.119.4.5 m_cpiBranchPredictor

SubsecondTime FastforwardPerformanceModel::m_cpiBranchPredictor [private]

Definition at line 19 of file fastforward_performance_model.h.

Referenced by FastforwardPerformanceModel(), and handleBranchMispredict().

6.119.4.6 m_cpiDataCache

std::vector< **SubsecondTime**> FastforwardPerformanceModel::m_cpiDataCache [private]

Definition at line 20 of file fastforward_performance_model.h.

Referenced by FastforwardPerformanceModel(), and handleMemoryLatency().

6.119.4.7 m_fastforwarded_time

SubsecondTime FastforwardPerformanceModel::m_fastforwarded_time [private]

Definition at line 16 of file fastforward_performance_model.h.

Referenced by FastforwardPerformanceModel(), getFastforwardedTime(), and incrementElapsedTime().

6.119.4.8 m_include_branch_mispredict

const bool FastforwardPerformanceModel::m_include_branch_mispredict [private]

Definition at line 12 of file fastforward_performance_model.h.

Referenced by handleBranchMispredict().

6.119.4.9 m_include_memory_latency

```
const bool FastforwardPerformanceModel::m_include_memory_latency  [private]
```

Definition at line 11 of file fastforward_performance_model.h.

Referenced by handleMemoryLatency().

6.119.4.10 m_perf

```
PerformanceModel* FastforwardPerformanceModel::m_perf  [private]
```

Definition at line 10 of file fastforward_performance_model.h.

Referenced by incrementElapsedTime().

The documentation for this class was generated from the following files:

- common/performance_model/ **fastforward_performance_model.h**
- common/performance_model/ **fastforward_performance_model.cc**

6.120 FaultinjectionManager Class Reference

```
#include <fault_injection.h>
```

Public Member Functions

- **FaultinjectionManager** (**fault_type_t** type, **fault_injector_t** injector)
- **FaultInjector** * **getFaultInjector** (**UInt32** core_id, **MemComponent::component_t** mem_component)
- void **applyFault** (**Core** *core, **IntPtr** read_address, **UInt32** data_size, **MemoryResult** &memres, **Byte** *data, const **Byte** *fault)

Static Public Member Functions

- static **FaultinjectionManager** * **create** ()

Private Types

- enum **fault_type_t** { **FAULT_TYPE_TOGGLE**, **FAULT_TYPE_SET0**, **FAULT_TYPE_SET1** }
- enum **fault_injector_t** { **FAULT_INJECTOR_NONE**, **FAULT_INJECTOR_RANDOM** }

Private Attributes

- **fault_type_t** **m_type**
- **fault_injector_t** **m_injector**

6.120.1 Detailed Description

Definition at line 9 of file fault_injection.h.

6.120.2 Member Enumeration Documentation

6.120.2.1 fault_injector_t

enum **FaultinjectionManager::fault_injector_t** [private]

Enumerator

FAULT_INJECTOR_NONE	
FAULT_INJECTOR_RANDOM	

Definition at line 19 of file fault_injection.h.

6.120.2.2 fault_type_t

enum **FaultinjectionManager::fault_type_t** [private]

Enumerator

FAULT_TYPE_TOGGLE	
FAULT_TYPE_SET0	
FAULT_TYPE_SET1	

Definition at line 12 of file fault_injection.h.

6.120.3 Constructor & Destructor Documentation

6.120.3.1 FaultinjectionManager()

```
FaultinjectionManager::FaultinjectionManager (
              fault_type_t type,
              fault_injector_t injector )
```

Definition at line 36 of file fault_injection.cc.

Referenced by create().

6.120.4 Member Function Documentation

6.120.4.1 applyFault()

```
void FaultinjectionManager::applyFault (
              Core * core,
              IntPtr read_address,
              UInt32 data_size,
              MemoryResult & memres,
              Byte * data,
             const  Byte * fault )
```

Definition at line 57 of file fault_injection.cc.

References FAULT_TYPE_SET0, FAULT_TYPE_SET1, FAULT_TYPE_TOGGLE, and m_type.

6.120.4.2 create()

```
FaultinjectionManager * FaultinjectionManager::create ( )  [static]
```

Definition at line 8 of file fault_injection.cc.

References FAULT_INJECTOR_NONE, FAULT_INJECTOR_RANDOM, FAULT_TYPE_SET0, FAULT_TYPE_↩ SET1, FAULT_TYPE_TOGGLE, FaultinjectionManager(), and LOG_PRINT_ERROR.

Referenced by Simulator::start().

6.120.4.3 getFaultInjector()

```
FaultInjector * FaultinjectionManager::getFaultInjector (
              UInt32 core_id,
              MemComponent::component_t mem_component )
```

Definition at line 43 of file fault_injection.cc.

References FAULT_INJECTOR_NONE, FAULT_INJECTOR_RANDOM, and m_injector.

6.120.5 Member Data Documentation

6.120.5.1 m_injector

```
fault_injector_t FaultinjectionManager::m_injector  [private]
```

Definition at line 23 of file fault_injection.h.

Referenced by getFaultInjector().

6.120.5.2 m_type

```
fault_type_t FaultinjectionManager::m_type  [private]
```

Definition at line 17 of file fault_injection.h.

Referenced by applyFault().

The documentation for this class was generated from the following files:

- common/fault_injection/ **fault_injection.h**
- common/fault_injection/ **fault_injection.cc**

6.121 FaultInjector Class Reference

```
#include <fault_injection.h>
```

Inheritance diagram for FaultInjector:

Public Member Functions

- **FaultInjector** (**UInt32** core_id, **MemComponent::component_t** mem_component)
- virtual void **preRead** (**IntPtr** addr, **IntPtr** location, **UInt32** data_size, **Byte** ∗fault, **SubsecondTime** time)
- virtual void **postWrite** (**IntPtr** addr, **IntPtr** location, **UInt32** data_size, **Byte** ∗fault, **SubsecondTime** time)

Protected Attributes

- **UInt32 m_core_id**
- **MemComponent::component_t m_mem_component**

6.121.1 Detailed Description

Definition at line 35 of file fault_injection.h.

6.121.2 Constructor & Destructor Documentation

6.121.2.1 FaultInjector()

```
FaultInjector::FaultInjector (
              UInt32 core_id,
              MemComponent::component_t mem_component )
```

Definition at line 76 of file fault_injection.cc.

6.121.3 Member Function Documentation

6.121.3.1 postWrite()

```
void FaultInjector::postWrite (
              IntPtr addr,
              IntPtr location,
              UInt32 data_size,
              Byte * fault,
              SubsecondTime time )  [virtual]
```

Reimplemented in **FaultInjectorRandom** (p. 565).

Definition at line 91 of file fault_injection.cc.

Referenced by Cache::accessSingleLine(), Cache::insertSingleLine(), and PrL1PrL2DramDirectoryMSI::Dram↩ Cntlr::putDataToDram().

6.121.3.2 preRead()

```
void FaultInjector::preRead (
              IntPtr addr,
              IntPtr location,
              UInt32 data_size,
              Byte * fault,
              SubsecondTime time )  [virtual]
```

Reimplemented in **FaultInjectorRandom** (p. 565).

Definition at line 83 of file fault_injection.cc.

Referenced by Cache::accessSingleLine(), and PrL1PrL2DramDirectoryMSI::DramCntlr::getDataFromDram().

6.121.4 Member Data Documentation

6.121.4.1 m_core_id

UInt32 FaultInjector::m_core_id [protected]

Definition at line 38 of file fault_injection.h.

Referenced by FaultInjectorRandom::preRead().

6.121.4.2 m_mem_component

MemComponent::component_t FaultInjector::m_mem_component [protected]

Definition at line 39 of file fault_injection.h.

Referenced by FaultInjectorRandom::preRead().

The documentation for this class was generated from the following files:

- common/fault_injection/ **fault_injection.h**
- common/fault_injection/ **fault_injection.cc**

6.122 FaultInjectorRandom Class Reference

#include <fault_injector_random.h>

Inheritance diagram for FaultInjectorRandom:

Public Member Functions

- **FaultInjectorRandom** (**UInt32** core_id, **MemComponent::component_t** mem_component)
- virtual void **preRead** (**IntPtr** addr, **IntPtr** location, **UInt32** data_size, **Byte** *fault, **SubsecondTime** time)
- virtual void **postWrite** (**IntPtr** addr, **IntPtr** location, **UInt32** data_size, **Byte** *fault, **SubsecondTime** time)

Private Attributes

- bool **m_active**
- **UInt64 m_rng**

Additional Inherited Members

6.122.1 Detailed Description

Definition at line 6 of file fault_injector_random.h.

6.122.2 Constructor & Destructor Documentation

6.122.2.1 FaultInjectorRandom()

```
FaultInjectorRandom::FaultInjectorRandom (
              UInt32 core_id,
              MemComponent::component_t mem_component )
```

Definition at line 4 of file fault_injector_random.cc.

References MemComponent::L1_DCACHE, and m_active.

6.122.3 Member Function Documentation

6.122.3.1 postWrite()

```
void FaultInjectorRandom::postWrite (
              IntPtr addr,
              IntPtr location,
              UInt32 data_size,
              Byte * fault,
              SubsecondTime time )  [virtual]
```

Reimplemented from **FaultInjector** (p. 563).

Definition at line 34 of file fault_injector_random.cc.

6.122.3.2 preRead()

```
void FaultInjectorRandom::preRead (
              IntPtr addr,
              IntPtr location,
              UInt32 data_size,
              Byte * fault,
              SubsecondTime time )  [virtual]
```

Reimplemented from **FaultInjector** (p. 563).

Definition at line 15 of file fault_injector_random.cc.

References m_active, FaultInjector::m_core_id, FaultInjector::m_mem_component, m_rng, MemComponent↩ String(), and rng_next().

6.122.4 Member Data Documentation

6.122.4.1 m_active

```
bool FaultInjectorRandom::m_active  [private]
```

Definition at line 15 of file fault_injector_random.h.

Referenced by FaultInjectorRandom(), and preRead().

6.122.4.2 m_rng

```
UInt64 FaultInjectorRandom::m_rng  [private]
```

Definition at line 16 of file fault_injector_random.h.

Referenced by preRead().

The documentation for this class was generated from the following files:

- common/fault_injection/ **fault_injector_random.h**
- common/fault_injection/ **fault_injector_random.cc**

6.123 config::FileNotFound Class Reference

```
#include <config_exceptions.hpp>
```

Inheritance diagram for config::FileNotFound:

Public Member Functions

- **FileNotFound** (const String &filename)
- virtual ∼**FileNotFound** () throw ()
- virtual const char ∗ **what** () const throw ()

Private Attributes

- String **m_filename**

6.123.1 Detailed Description

Definition at line 45 of file config_exceptions.hpp.

6.123.2 Constructor & Destructor Documentation

6.123.2.1 FileNotFound()

```
config::FileNotFound::FileNotFound (
            const String & filename )  [inline]
```

Definition at line 48 of file config_exceptions.hpp.

References m_filename.

6.123.2.2 ~FileNotFound()

```
virtual config::FileNotFound::~FileNotFound ( ) throw ( )   [inline], [virtual]
```

Definition at line 52 of file config_exceptions.hpp.

6.123.3 Member Function Documentation

6.123.3.1 what()

```
virtual const char* config::FileNotFound::what ( ) const throw ( )   [inline], [virtual]
```

Definition at line 53 of file config_exceptions.hpp.

References m_filename.

6.123.4 Member Data Documentation

6.123.4.1 m_filename

```
String config::FileNotFound::m_filename  [private]
```

Definition at line 58 of file config_exceptions.hpp.

Referenced by FileNotFound(), and what().

The documentation for this class was generated from the following file:

- common/config/ **config_exceptions.hpp**

6.124 FloatDistribution Class Reference

```
#include <distribution.h>
```

Inheritance diagram for FloatDistribution:

Public Member Functions

- virtual ~**FloatDistribution** ()
- virtual double **next** ()=0

6.124.1 Detailed Description

Definition at line 8 of file distribution.h.

6.124.2 Constructor & Destructor Documentation

6.124.2.1 ~FloatDistribution()

```
virtual FloatDistribution::~FloatDistribution ( )  [inline], [virtual]
```

Definition at line 11 of file distribution.h.

6.124.3 Member Function Documentation

6.124.3.1 next()

```
virtual double FloatDistribution::next ( )  [pure virtual]
```

Implemented in **NormalFloatDistribution** (p. 875).

The documentation for this class was generated from the following file:

- common/misc/ **distribution.h**

6.125 FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc > Class Template Reference

```
#include <FSBAllocator.hh>
```

Classes

- struct **BlocksVector**
- class **MemBlock**

Public Member Functions

- void * **allocate** ()
- void **deallocate** (void *ptr)

Private Types

- typedef std::size_t **Data_t**

Private Attributes

- **BlocksVector blocksVector**
- std::vector< **Data_t** > **blocksWithFree**

Static Private Attributes

- static const **Data_t BlockElements** = 512
- static const **Data_t DSize** = sizeof(**Data_t**)
- static const **Data_t ElemSizeInDSize** = (ElemSize + (**DSize**-1)) / **DSize**
- static const **Data_t UnitSizeInDSize** = **ElemSizeInDSize** + 1
- static const **Data_t BlockSize** = **BlockElements** * **UnitSizeInDSize**

6.125.1 Detailed Description

template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
class FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >

Definition at line 41 of file FSBAllocator.hh.

6.125.2 Member Typedef Documentation

6.125.2.1 Data_t

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
typedef std::size_t  FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::  Data_↩
t  [private]
```

Definition at line 43 of file FSBAllocator.hh.

6.125.3 Member Function Documentation

6.125.3.1 allocate()

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
void*  FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::allocate ( )  [inline]
```

Definition at line 124 of file FSBAllocator.hh.

6.125.3.2 deallocate()

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
void  FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::deallocate (
            void * ptr )  [inline]
```

Definition at line 149 of file FSBAllocator.hh.

6.125.4 Member Data Documentation

6.125.4.1 BlockElements

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
const Data_t FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlockElements =
512 [static], [private]
```

Definition at line 44 of file FSBAllocator.hh.

Referenced by FSBAllocator_ElemAllocator< sizeof(DataElement)+sizeof(T), 0, T >::allocate(), and FSB↩ Allocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::isFull().

6.125.4.2 BlockSize

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
const Data_t FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlockSize =
BlockElements* UnitSizeInDSize [static], [private]
```

Definition at line 49 of file FSBAllocator.hh.

6.125.4.3 blocksVector

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
BlocksVector FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::blocksVector
[private]
```

Definition at line 120 of file FSBAllocator.hh.

Referenced by FSBAllocator_ElemAllocator< sizeof(DataElement)+sizeof(T), 0, T >::allocate(), and FSB↩ Allocator_ElemAllocator< sizeof(DataElement)+sizeof(T), 0, T >::deallocate().

6.125.4.4 blocksWithFree

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
std::vector< Data_t> FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::blocks↩
WithFree [private]
```

Definition at line 121 of file FSBAllocator.hh.

Referenced by FSBAllocator_ElemAllocator< sizeof(DataElement)+sizeof(T), 0, T >::allocate(), and FSB↩ Allocator_ElemAllocator< sizeof(DataElement)+sizeof(T), 0, T >::deallocate().

6.125.4.5 DSize

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
const Data_t FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::DSize = sizeof(
Data_t) [static], [private]
```

Definition at line 46 of file FSBAllocator.hh.

6.125.4.6 ElemSizeInDSize

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
const Data_t FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::ElemSizeInDSize =
(ElemSize + ( DSize-1)) / DSize [static], [private]
```

Definition at line 47 of file FSBAllocator.hh.

Referenced by FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::allocate(), and FSBAllocator_ElemAllocator< sizeof(DataElement)+sizeof(T), 0, T >::deallocate().

6.125.4.7 UnitSizeInDSize

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
const Data_t FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::UnitSizeInDSize =
ElemSizeInDSize + 1 [static], [private]
```

Definition at line 48 of file FSBAllocator.hh.

Referenced by FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::allocate().

The documentation for this class was generated from the following file:

- common/misc/ **FSBAllocator.hh**

6.126 PthreadThread::FuncData Struct Reference

Public Member Functions

- **FuncData** (**ThreadFunc** f, void *a)

Public Attributes

- **ThreadFunc func**
- void * **arg**

6.126.1 Detailed Description

Definition at line 17 of file pthread_thread.h.

6.126.2 Constructor & Destructor Documentation

6.126.2.1 FuncData()

```
PthreadThread::FuncData::FuncData (
            ThreadFunc f,
            void * a )  [inline]
```

Definition at line 22 of file pthread_thread.h.

6.126.3 Member Data Documentation

6.126.3.1 arg

```
void* PthreadThread::FuncData::arg
```

Definition at line 20 of file pthread_thread.h.

Referenced by PthreadThread::run(), and PthreadThread::spawnedThreadFunc().

6.126.3.2 func

```
ThreadFunc PthreadThread::FuncData::func
```

Definition at line 19 of file pthread_thread.h.

Referenced by PthreadThread::run(), and PthreadThread::spawnedThreadFunc().

The documentation for this struct was generated from the following file:

- common/misc/ **pthread_thread.h**

6.127 FunctionTracer Class Reference

```
#include <log.h>
```

Public Member Functions

- **FunctionTracer** (const char *file, int line, const char *fn)
- ~**FunctionTracer** ()

Private Attributes

- const char * **m_file**
- int **m_line**
- const char * **m_fn**

6.127.1 Detailed Description

Definition at line 172 of file log.h.

6.127.2 Constructor & Destructor Documentation

6.127.2.1 FunctionTracer()

```
FunctionTracer::FunctionTracer (
            const char * file,
            int line,
            const char * fn )  [inline]
```

Definition at line 175 of file log.h.

References __LOG_PRINT, m_file, m_fn, m_line, and Log::None.

6.127.2.2 ~FunctionTracer()

```
FunctionTracer::~FunctionTracer ( )  [inline]
```

Definition at line 183 of file log.h.

References __LOG_PRINT, m_file, m_fn, m_line, and Log::None.

6.127.3 Member Data Documentation

6.127.3.1 m_file

```
const char* FunctionTracer::m_file  [private]
```

Definition at line 189 of file log.h.

Referenced by FunctionTracer(), and ~FunctionTracer().

6.127.3.2 m_fn

```
const char* FunctionTracer::m_fn  [private]
```

Definition at line 191 of file log.h.

Referenced by FunctionTracer(), and ~FunctionTracer().

6.127.3.3 m_line

```
int FunctionTracer::m_line  [private]
```

Definition at line 190 of file log.h.

Referenced by FunctionTracer(), and ~FunctionTracer().

The documentation for this class was generated from the following file:

- common/misc/ **log.h**

6.128 SyscallServer::futex_args_t Struct Reference

```
#include <syscall_server.h>
```

Public Attributes

- int * **uaddr**
- int **op**
- int **val**
- const struct timespec * **timeout**
- int **val2**
- int * **uaddr2**
- int **val3**

6.128.1 Detailed Description

Definition at line 49 of file syscall_server.h.

6.128.2 Member Data Documentation

6.128.2.1 op

```
int SyscallServer::futex_args_t::op
```

Definition at line 51 of file syscall_server.h.

Referenced by SyscallServer::handleFutexCall(), and SyscallMdl::handleFutexCall().

6.128.2.2 timeout

```
const struct timespec* SyscallServer::futex_args_t::timeout
```

Definition at line 53 of file syscall_server.h.

Referenced by SyscallServer::handleFutexCall(), and SyscallMdl::handleFutexCall().

6.128.2.3 uaddr

```
int* SyscallServer::futex_args_t::uaddr
```

Definition at line 50 of file syscall_server.h.

Referenced by SyscallServer::handleFutexCall(), and SyscallMdl::handleFutexCall().

6.128.2.4 uaddr2

```
int* SyscallServer::futex_args_t::uaddr2
```

Definition at line 55 of file syscall_server.h.

Referenced by SyscallServer::handleFutexCall(), and SyscallMdl::handleFutexCall().

6.128.2.5 val

```
int SyscallServer::futex_args_t::val
```

Definition at line 52 of file syscall_server.h.

Referenced by SyscallServer::handleFutexCall(), and SyscallMdl::handleFutexCall().

6.128.2.6 val2

```
int SyscallServer::futex_args_t::val2
```

Definition at line 54 of file syscall_server.h.

Referenced by SyscallServer::handleFutexCall(), and SyscallMdl::handleFutexCall().

6.128.2.7 val3

```
int SyscallServer::futex_args_t::val3
```

Definition at line 56 of file syscall_server.h.

Referenced by SyscallServer::handleFutexCall(), and SyscallMdl::handleFutexCall().

The documentation for this struct was generated from the following file:

- common/system/ **syscall_server.h**

6.129 SyscallMdl::futex_counters_t Struct Reference

Public Attributes

- uint64_t **count** [16]
- **SubsecondTime delay** [16]

6.129.1 Detailed Description

Definition at line 56 of file syscall_model.h.

6.129.2 Member Data Documentation

6.129.2.1 count

```
uint64_t SyscallMdl::futex_counters_t::count[16]
```

Definition at line 58 of file syscall_model.h.

Referenced by SyscallMdl::futexCount(), and SyscallMdl::SyscallMdl().

6.129.2.2 delay

```
SubsecondTime SyscallMdl::futex_counters_t::delay[16]
```

Definition at line 59 of file syscall_model.h.

Referenced by SyscallMdl::futexCount(), and SyscallMdl::SyscallMdl().

The documentation for this struct was generated from the following file:

- common/core/ **syscall_model.h**

6.130 Fxsupport Class Reference

```
#include <fxsupport.h>
```

Public Member Functions

- void **fxsave** ()
- void **fxrstor** ()

Static Public Member Functions

- static void **init** ()
- static void **fini** ()
- static **Fxsupport** * **getSingleton** ()

Private Member Functions

- **Fxsupport** (**core_id_t** core_count)
- ~**Fxsupport** ()

Private Attributes

- char ** **m_fx_buf**
- uint64_t ** **m_ref_count**
- **core_id_t** **m_core_count**

Static Private Attributes

- static **Fxsupport** * **m_singleton**

6.130.1 Detailed Description

Definition at line 6 of file fxsupport.h.

6.130.2 Constructor & Destructor Documentation

6.130.2.1 Fxsupport()

```
Fxsupport::Fxsupport (
            core_id_t core_count )  [private]
```

Definition at line 8 of file fxsupport.cc.

References __attribute__, m_core_count, m_fx_buf, and m_ref_count.

Referenced by init().

6.130.2.2 ~Fxsupport()

```
Fxsupport::~Fxsupport ( )  [private]
```

Definition at line 24 of file fxsupport.cc.

References m_core_count, m_fx_buf, and m_ref_count.

6.130.3 Member Function Documentation

6.130.3.1 fini()

```
void Fxsupport::fini ( )  [static]
```

Definition at line 41 of file fxsupport.cc.

References m_singleton.

6.130.3.2 fxrstor()

```
void Fxsupport::fxrstor ( )
```

Definition at line 73 of file fxsupport.cc.

References LOG_PRINT, m_fx_buf, and m_ref_count.

Referenced by ScopedFxsave::~ScopedFxsave().

6.130.3.3 fxsave()

```
void Fxsupport::fxsave ( )
```

Definition at line 53 of file fxsupport.cc.

References LOG_PRINT, m_fx_buf, and m_ref_count.

Referenced by ScopedFxsave::ScopedFxsave().

6.130.3.4 getSingleton()

```
Fxsupport * Fxsupport::getSingleton ( )  [static]
```

Definition at line 47 of file fxsupport.cc.

References m_singleton.

Referenced by ScopedFxsave::ScopedFxsave(), and ScopedFxsave::~ScopedFxsave().

6.130.3.5 init()

```
void Fxsupport::init ( )  [static]
```

Definition at line 34 of file fxsupport.cc.

References Fxsupport(), and m_singleton.

Referenced by Simulator::start().

6.130.4 Member Data Documentation

6.130.4.1 m_core_count

```
core_id_t Fxsupport::m_core_count  [private]
```

Definition at line 24 of file fxsupport.h.

Referenced by Fxsupport(), and ~Fxsupport().

6.130.4.2 m_fx_buf

```
char** Fxsupport::m_fx_buf  [private]
```

Definition at line 21 of file fxsupport.h.

Referenced by fxrstor(), fxsave(), Fxsupport(), and ~Fxsupport().

6.130.4.3 m_ref_count

```
uint64_t** Fxsupport::m_ref_count  [private]
```

Definition at line 23 of file fxsupport.h.

Referenced by fxrstor(), fxsave(), Fxsupport(), and ~Fxsupport().

6.130.4.4 m_singleton

```
Fxsupport * Fxsupport::m_singleton  [static], [private]
```

Definition at line 26 of file fxsupport.h.

Referenced by fini(), getSingleton(), and init().

The documentation for this class was generated from the following files:

- common/misc/ **fxsupport.h**
- common/misc/ **fxsupport.cc**

6.131 GenericInstruction Class Reference

```
#include <instruction.h>
```

Inheritance diagram for GenericInstruction:

Public Member Functions

- **GenericInstruction** (**OperandList** &operands)

Additional Inherited Members

6.131.1 Detailed Description

Definition at line 97 of file instruction.h.

6.131.2 Constructor & Destructor Documentation

6.131.2.1 GenericInstruction()

```
GenericInstruction::GenericInstruction (
              OperandList & operands )  [inline]
```

Definition at line 100 of file instruction.h.

The documentation for this class was generated from the following file:

- common/performance_model/ **instruction.h**

6.132 GhbPrefetcher::GHBEntry Struct Reference

Public Member Functions

- **GHBEntry** ()

Public Attributes

- **UInt32 nextIndex**
- **SInt64 delta**
- **UInt32 generation**

6.132.1 Detailed Description

Definition at line 18 of file ghb_prefetcher.h.

6.132.2 Constructor & Destructor Documentation

6.132.2.1 GHBEntry()

```
GhbPrefetcher::GHBEntry::GHBEntry ( )  [inline]
```

Definition at line 23 of file ghb_prefetcher.h.

6.132.3 Member Data Documentation

6.132.3.1 delta

```
SInt64 GhbPrefetcher::GHBEntry::delta
```

Definition at line 21 of file ghb_prefetcher.h.

6.132.3.2 generation

```
UInt32 GhbPrefetcher::GHBEntry::generation
```

Definition at line 22 of file ghb_prefetcher.h.

6.132.3.3 nextIndex

```
UInt32 GhbPrefetcher::GHBEntry::nextIndex
```

Definition at line 20 of file ghb_prefetcher.h.

The documentation for this struct was generated from the following file:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **ghb_prefetcher.h**

6.133 GhbPrefetcher Class Reference

```
#include <ghb_prefetcher.h>
```

Inheritance diagram for GhbPrefetcher:

Classes

- struct **GHBEntry**
- struct **TableEntry**

Public Member Functions

- **GhbPrefetcher** (String configName, **core_id_t** core_id)
- std::vector< **IntPtr** > **getNextAddress** (**IntPtr** currentAddress, **core_id_t** core_id)
- ~**GhbPrefetcher** ()

Private Attributes

- **UInt32 m_prefetchWidth**
- **UInt32 m_prefetchDepth**
- **IntPtr m_lastAddress**
- **UInt32 m_ghbSize**
- **UInt32 m_ghbHead**
- **UInt32 m_generation**
- std::vector< **GHBEntry** > **m_ghb**
- **UInt32 m_tableSize**
- **UInt32 m_tableHead**
- std::vector< **TableEntry** > **m_ghbTable**

Static Private Attributes

- static const **SInt64 INVALID_DELTA** = INT64_MAX
- static const **UInt32 INVALID_INDEX** = UINT32_MAX

Additional Inherited Members

6.133.1 Detailed Description

Definition at line 6 of file ghb_prefetcher.h.

6.133.2 Constructor & Destructor Documentation

6.133.2.1 GhbPrefetcher()

```
GhbPrefetcher::GhbPrefetcher (
             String configName,
              core_id_t core_id )
```

Definition at line 7 of file ghb_prefetcher.cc.

6.133.2.2 ~GhbPrefetcher()

```
GhbPrefetcher::~GhbPrefetcher ( )
```

Definition at line 21 of file ghb_prefetcher.cc.

6.133.3 Member Function Documentation

6.133.3.1 getNextAddress()

```
std::vector< IntPtr > GhbPrefetcher::getNextAddress (
            IntPtr currentAddress,
            core_id_t core_id ) [virtual]
```

Implements **Prefetcher** (p. 938).

Definition at line 26 of file ghb_prefetcher.cc.

References INVALID_ADDRESS, INVALID_DELTA, INVALID_INDEX, m_generation, m_ghb, m_ghbHead, m_↩ ghbSize, m_ghbTable, m_lastAddress, m_prefetchDepth, m_prefetchWidth, m_tableHead, and m_tableSize.

6.133.4 Member Data Documentation

6.133.4.1 INVALID_DELTA

```
const SInt64 GhbPrefetcher::INVALID_DELTA = INT64_MAX [static], [private]
```

Definition at line 15 of file ghb_prefetcher.h.

Referenced by getNextAddress().

6.133.4.2 INVALID_INDEX

```
const UInt32 GhbPrefetcher::INVALID_INDEX = UINT32_MAX [static], [private]
```

Definition at line 16 of file ghb_prefetcher.h.

Referenced by getNextAddress().

6.133.4.3 m_generation

UInt32 GhbPrefetcher::m_generation [private]

Definition at line 42 of file ghb_prefetcher.h.

Referenced by getNextAddress().

6.133.4.4 m_ghb

std::vector< **GHBEntry**> GhbPrefetcher::m_ghb [private]

Definition at line 43 of file ghb_prefetcher.h.

Referenced by getNextAddress().

6.133.4.5 m_ghbHead

UInt32 GhbPrefetcher::m_ghbHead [private]

Definition at line 41 of file ghb_prefetcher.h.

Referenced by getNextAddress().

6.133.4.6 m_ghbSize

UInt32 GhbPrefetcher::m_ghbSize [private]

Definition at line 40 of file ghb_prefetcher.h.

Referenced by getNextAddress().

6.133.4.7 m_ghbTable

std::vector< **TableEntry**> GhbPrefetcher::m_ghbTable [private]

Definition at line 47 of file ghb_prefetcher.h.

Referenced by getNextAddress().

6.133.4.8 m_lastAddress

```
IntPtr GhbPrefetcher::m_lastAddress [private]
```

Definition at line 37 of file ghb_prefetcher.h.

Referenced by getNextAddress().

6.133.4.9 m_prefetchDepth

```
UInt32 GhbPrefetcher::m_prefetchDepth [private]
```

Definition at line 35 of file ghb_prefetcher.h.

Referenced by getNextAddress().

6.133.4.10 m_prefetchWidth

```
UInt32 GhbPrefetcher::m_prefetchWidth [private]
```

Definition at line 34 of file ghb_prefetcher.h.

Referenced by getNextAddress().

6.133.4.11 m_tableHead

```
UInt32 GhbPrefetcher::m_tableHead [private]
```

Definition at line 46 of file ghb_prefetcher.h.

Referenced by getNextAddress().

6.133.4.12 m_tableSize

```
UInt32 GhbPrefetcher::m_tableSize [private]
```

Definition at line 45 of file ghb_prefetcher.h.

Referenced by getNextAddress().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **ghb_prefetcher.h**
- common/core/memory_subsystem/parametric_dram_directory_msi/ **ghb_prefetcher.cc**

6.134 GlobalPredictor Class Reference

`#include <global_predictor.h>`

Inheritance diagram for GlobalPredictor:

Classes

- class **Way**

Public Member Functions

- **GlobalPredictor** (**UInt32** entries, **UInt32** tag_bitwidth, **UInt32** ways)
- bool **predict** (**IntPtr** ip, **IntPtr** target)
- **BranchPredictorReturnValue lookup** (**IntPtr** ip, **IntPtr** target)
- void **update** (bool predicted, bool actual, **IntPtr** ip, **IntPtr** target)
- bool **predict** (**IntPtr** ip, **IntPtr** target, **IntPtr** pir)
- **BranchPredictorReturnValue lookup** (**IntPtr** ip, **IntPtr** target, **IntPtr** pir)
- void **update** (bool predicted, bool actual, **IntPtr** ip, **IntPtr** target, **IntPtr** pir)
- void **evict** (**IntPtr** ip, **IntPtr** pir)

Private Member Functions

- void **gen_index_tag** (**IntPtr** ip, **IntPtr** pir, **UInt32** &index, **UInt32** &tag)

Private Attributes

- **UInt64 m_lru_use_count**
- **UInt32 m_num_ways**
- std::vector< **Way** > **m_ways**

Additional Inherited Members

6.134.1 Detailed Description

Definition at line 12 of file global_predictor.h.

6.134.2 Constructor & Destructor Documentation

6.134.2.1 GlobalPredictor()

```
GlobalPredictor::GlobalPredictor (
              UInt32 entries,
              UInt32 tag_bitwidth,
              UInt32 ways )  [inline]
```

Definition at line 17 of file global_predictor.h.

6.134.3 Member Function Documentation

6.134.3.1 evict()

```
void GlobalPredictor::evict (
              IntPtr ip,
              IntPtr pir )  [inline]
```

Definition at line 103 of file global_predictor.h.

References gen_index_tag(), m_num_ways, and m_ways.

Referenced by PentiumMBranchPredictor::update().

6.134.3.2 gen_index_tag()

```
void GlobalPredictor::gen_index_tag (
              IntPtr ip,
              IntPtr pir,
              UInt32 & index,
              UInt32 & tag )  [inline], [private]
```

Definition at line 147 of file global_predictor.h.

Referenced by evict(), lookup(), predict(), and update().

6.134.3.3 lookup() [1/2]

```
BranchPredictorReturnValue GlobalPredictor::lookup (
            IntPtr ip,
            IntPtr target ) [inline]
```

Definition at line 25 of file global_predictor.h.

References LOG_PRINT_ERROR.

Referenced by PentiumMBranchPredictor::predict().

6.134.3.4 lookup() [2/2]

```
BranchPredictorReturnValue GlobalPredictor::lookup (
            IntPtr ip,
            IntPtr target,
            IntPtr pir ) [inline]
```

Definition at line 44 of file global_predictor.h.

References gen_index_tag(), BranchPredictorReturnValue::hit, BranchPredictorReturnValue::InvalidBranch, m_↩num_ways, m_ways, and BranchPredictorReturnValue::prediction.

6.134.3.5 predict() [1/2]

```
bool GlobalPredictor::predict (
            IntPtr ip,
            IntPtr target ) [inline], [virtual]
```

Implements **BranchPredictor** (p. 125).

Definition at line 24 of file global_predictor.h.

References LOG_PRINT_ERROR.

6.134.3.6 predict() [2/2]

```
bool GlobalPredictor::predict (
            IntPtr ip,
            IntPtr target,
            IntPtr pir ) [inline]
```

Definition at line 28 of file global_predictor.h.

References gen_index_tag(), m_num_ways, and m_ways.

6.134.3.7 update() [1/2]

```
void GlobalPredictor::update (
             bool predicted,
             bool actual,
              IntPtr ip,
              IntPtr target )  [inline], [virtual]
```

Implements **BranchPredictor** (p. 126).

Definition at line 26 of file global_predictor.h.

References LOG_PRINT_ERROR.

Referenced by PentiumMBranchPredictor::update().

6.134.3.8 update() [2/2]

```
void GlobalPredictor::update (
             bool predicted,
             bool actual,
              IntPtr ip,
              IntPtr target,
              IntPtr pir )  [inline]
```

Definition at line 65 of file global_predictor.h.

References gen_index_tag(), m_lru_use_count, m_num_ways, and m_ways.

6.134.4 Member Data Documentation

6.134.4.1 m_lru_use_count

```
UInt64 GlobalPredictor::m_lru_use_count  [private]
```

Definition at line 153 of file global_predictor.h.

Referenced by update().

6.134.4.2 m_num_ways

```
UInt32 GlobalPredictor::m_num_ways  [private]
```

Definition at line 154 of file global_predictor.h.

Referenced by evict(), lookup(), predict(), and update().

6.134.4.3 m_ways

```
std::vector< Way> GlobalPredictor::m_ways  [private]
```

Definition at line 155 of file global_predictor.h.

Referenced by evict(), lookup(), predict(), and update().

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **global_predictor.h**

6.135 std::hash< HitWhere::where_t > Struct Reference

```
#include <hit_where.h>
```

Public Member Functions

- size_t **operator()** (const **HitWhere::where_t** &type) const

6.135.1 Detailed Description

Definition at line 41 of file hit_where.h.

6.135.2 Member Function Documentation

6.135.2.1 operator()()

```
size_t std::hash<  HitWhere::where_t >::operator() (
            const  HitWhere::where_t & type ) const  [inline]
```

Definition at line 42 of file hit_where.h.

The documentation for this struct was generated from the following file:

- common/performance_model/ **hit_where.h**

6.136 std::hash< HookType::hook_type_t > Struct Reference

```
#include <hooks_manager.h>
```

Public Member Functions

- size_t **operator()** (const **HookType::hook_type_t** &type) const

6.136.1 Detailed Description

Definition at line 48 of file hooks_manager.h.

6.136.2 Member Function Documentation

6.136.2.1 operator()()

```
size_t std::hash<  HookType::hook_type_t >::operator() (
            const  HookType::hook_type_t & type ) const  [inline]
```

Definition at line 49 of file hooks_manager.h.

The documentation for this struct was generated from the following file:

- common/system/ **hooks_manager.h**

6.137 std::hash< REG > Struct Reference

```
#include <spin_loop_detection.h>
```

Public Member Functions

- size_t **operator()** (const REG &reg) const

6.137.1 Detailed Description

Definition at line 11 of file spin_loop_detection.h.

6.137.2 Member Function Documentation

6.137.2.1 operator()()

```
size_t std::hash< REG >::operator() (
            const REG & reg ) const  [inline]
```

Definition at line 12 of file spin_loop_detection.h.

The documentation for this struct was generated from the following file:

- pin/ **spin_loop_detection.h**

6.138 std::hash< std::deque< T > > Struct Template Reference

```
#include <routine_tracer.h>
```

Public Member Functions

- size_t **operator()** (const std::deque< T > &a) const

6.138.1 Detailed Description

template<typename T>
struct std::hash< std::deque< T > >

Definition at line 14 of file routine_tracer.h.

6.138.2 Member Function Documentation

6.138.2.1 operator()()

```
template<typename T >
size_t std::hash< std::deque< T > >::operator() (
            const std::deque< T > & a ) const  [inline]
```

Definition at line 16 of file routine_tracer.h.

The documentation for this struct was generated from the following file:

- common/system/ **routine_tracer.h**

6.139 hash_table Struct Reference

```
#include <util.h>
```

Public Attributes

- intptr_t **addr**
- int **grptime**
- int **prty**
- int **inum**
- struct **hash_table** * **nxt**

6.139.1 Detailed Description

Definition at line 31 of file util.h.

6.139.2 Member Data Documentation

6.139.2.1 addr

```
intptr_t hash_table::addr
```

Definition at line 32 of file util.h.

6.139.2.2 grptime

```
int hash_table::grptime
```

Definition at line 33 of file util.h.

6.139.2.3 inum

```
int hash_table::inum
```

Definition at line 35 of file util.h.

6.139.2.4 nxt

```
struct hash_table* hash_table::nxt
```

Definition at line 36 of file util.h.

Referenced by UHT_Add_to_free_list(), and UHT_Get_from_free_list().

6.139.2.5 prty

```
int hash_table::prty
```

Definition at line 34 of file util.h.

The documentation for this struct was generated from the following file:

- common/core/memory_subsystem/cheetah/ **util.h**

6.140 HashMapSet< T > Class Template Reference

```
#include <hash_map_set.h>
```

Public Member Functions

- **HashMapSet** (**UInt32** num_buckets, **UInt32**(*hash_fn)(T, **UInt32**, **UInt32**), **UInt32** hash_fn_param)
- ~**HashMapSet** ()
- void **insert** (T elem)
- **UInt32 count** (T elem)
- void **erase** (T elem)
- void **clear** ()

Private Attributes

- std::set< T > ** **m_set_list**
- **UInt32 m_num_buckets**
- **UInt32**(* **m_hash_fn**)(T, **UInt32**, **UInt32**)
- **UInt32 m_hash_fn_param**

6.140.1 Detailed Description

template<class T>
class HashMapSet< T >

Definition at line 9 of file hash_map_set.h.

6.140.2 Constructor & Destructor Documentation

6.140.2.1 HashMapSet()

```
template<class T >
HashMapSet< T >::  HashMapSet (
              UInt32 num_buckets,
              UInt32(*)(T,  UInt32,  UInt32) hash_fn,
              UInt32 hash_fn_param )  [inline]
```

Definition at line 18 of file hash_map_set.h.

References HashMapSet< T >::m_num_buckets, and HashMapSet< T >::m_set_list.

6.140.2.2 ~HashMapSet()

```
template<class T >
HashMapSet< T >::~ HashMapSet ( )  [inline]
```

Definition at line 30 of file hash_map_set.h.

6.140.3 Member Function Documentation

6.140.3.1 clear()

```
template<class T >
void  HashMapSet< T >::clear ( )  [inline]
```

Definition at line 49 of file hash_map_set.h.

References HashMapSet< T >::m_num_buckets, and HashMapSet< T >::m_set_list.

6.140.3.2 count()

```
template<class T >
UInt32  HashMapSet< T >::count (
             T elem )  [inline]
```

Definition at line 39 of file hash_map_set.h.

References HashMapSet< T >::m_hash_fn, HashMapSet< T >::m_hash_fn_param, HashMapSet< T >::m_↩ num_buckets, and HashMapSet< T >::m_set_list.

6.140.3.3 erase()

```
template<class T >
void  HashMapSet< T >::erase (
            T elem )  [inline]
```

Definition at line 44 of file hash_map_set.h.

References HashMapSet< T >::m_hash_fn, HashMapSet< T >::m_hash_fn_param, HashMapSet< T >::m_↩ num_buckets, and HashMapSet< T >::m_set_list.

6.140.3.4 insert()

```
template<class T >
void  HashMapSet< T >::insert (
            T elem )  [inline]
```

Definition at line 32 of file hash_map_set.h.

References HashMapSet< T >::m_hash_fn, HashMapSet< T >::m_hash_fn_param, HashMapSet< T >::m_↩ num_buckets, and HashMapSet< T >::m_set_list.

6.140.4 Member Data Documentation

6.140.4.1 m_hash_fn

```
template<class T >
UInt32(*  HashMapSet< T >::m_hash_fn) (T,  UInt32,  UInt32)  [private]
```

Definition at line 14 of file hash_map_set.h.

Referenced by HashMapSet< T >::count(), HashMapSet< T >::erase(), and HashMapSet< T >::insert().

6.140.4.2 m_hash_fn_param

```
template<class T >
UInt32  HashMapSet< T >::m_hash_fn_param  [private]
```

Definition at line 15 of file hash_map_set.h.

Referenced by HashMapSet< T >::count(), HashMapSet< T >::erase(), and HashMapSet< T >::insert().

6.140.4.3 m_num_buckets

```
template<class T >
UInt32 HashMapSet< T >::m_num_buckets [private]
```

Definition at line 13 of file hash_map_set.h.

Referenced by HashMapSet< T >::clear(), HashMapSet< T >::count(), HashMapSet< T >::erase(), HashMapSet< T >::HashMapSet(), and HashMapSet< T >::insert().

6.140.4.4 m_set_list

```
template<class T >
std::set<T>** HashMapSet< T >::m_set_list [private]
```

Definition at line 12 of file hash_map_set.h.

Referenced by HashMapSet< T >::clear(), HashMapSet< T >::count(), HashMapSet< T >::erase(), HashMapSet< T >::HashMapSet(), and HashMapSet< T >::insert().

The documentation for this class was generated from the following file:

- common/misc/ **hash_map_set.h**

6.141 HitWhere Class Reference

`#include <hit_where.h>`

Public Types

- enum **where_t** {
  WHERE_FIRST = 0, **L1I** = MemComponent::L1_ICACHE, **L1_OWN** = MemComponent::L1_DCACHE,
  L2_OWN = MemComponent::L2_CACHE,
  L3_OWN = MemComponent::L3_CACHE, **L4_OWN** = MemComponent::L4_CACHE, **MISS**, **NUCA_CACHE**,
  DRAM_CACHE, **DRAM**, **DRAM_LOCAL**, **DRAM_REMOTE**,
  CACHE_REMOTE, **SIBLING**, **L1_SIBLING** = MemComponent::L1_DCACHE + SIBLING, **L2_SIBLING** = MemComponent::L2_CACHE + SIBLING,
  L3_SIBLING = MemComponent::L3_CACHE + SIBLING, **L4_SIBLING** = MemComponent::L4_CACHE + SIBLING, **UNKNOWN**, **PREDICATE_FALSE**,
  PREFETCH_NO_MAPPING, **NUM_HITWHERES** }

6.141.1 Detailed Description

Definition at line 9 of file hit_where.h.

6.141.2 Member Enumeration Documentation

6.141.2.1 where_t

```
enum HitWhere::where_t
```

Enumerator

WHERE_FIRST	
L1I	
L1_OWN	
L2_OWN	
L3_OWN	
L4_OWN	
MISS	
NUCA_CACHE	
DRAM_CACHE	
DRAM	
DRAM_LOCAL	
DRAM_REMOTE	
CACHE_REMOTE	
SIBLING	
L1_SIBLING	
L2_SIBLING	
L3_SIBLING	
L4_SIBLING	
UNKNOWN	
PREDICATE_FALSE	
PREFETCH_NO_MAPPING	
NUM_HITWHERES	

Definition at line 12 of file hit_where.h.

The documentation for this class was generated from the following file:

- common/performance_model/ **hit_where.h**

6.142 HooksManager::HookCallback Struct Reference

`#include <hooks_manager.h>`

Public Member Functions

- **HookCallback** (**HookCallbackFunc** _func, **UInt64** _arg, **HookCallbackOrder** _order)

Public Attributes

- **HookCallbackFunc func**
- **UInt64 arg**
- **HookCallbackOrder order**

6.142.1 Detailed Description

Definition at line 67 of file hooks_manager.h.

6.142.2 Constructor & Destructor Documentation

6.142.2.1 HookCallback()

```
HooksManager::HookCallback::HookCallback (
              HookCallbackFunc _func,
              UInt64 _arg,
              HookCallbackOrder _order )  [inline]
```

Definition at line 71 of file hooks_manager.h.

6.142.3 Member Data Documentation

6.142.3.1 arg

```
UInt64 HooksManager::HookCallback::arg
```

Definition at line 69 of file hooks_manager.h.

6.142.3.2 func

```
HookCallbackFunc HooksManager::HookCallback::func
```

Definition at line 68 of file hooks_manager.h.

6.142.3.3 order

```
HookCallbackOrder HooksManager::HookCallback::order
```

Definition at line 70 of file hooks_manager.h.

The documentation for this struct was generated from the following file:

- common/system/ **hooks_manager.h**

6.143 HooksManager Class Reference

```
#include <hooks_manager.h>
```

Classes

- struct **HookCallback**
- struct **ThreadCreate**
- struct **ThreadMigrate**
- struct **ThreadResume**
- struct **ThreadStall**
- struct **ThreadTime**

Public Types

- enum **HookCallbackOrder** { **ORDER_NOTIFY_PRE**, **ORDER_ACTION**, **ORDER_NOTIFY_POST**, **NU↩ M_HOOK_ORDER** }
- typedef **SInt64**(∗ **HookCallbackFunc**) (**UInt64**, **UInt64**)

Public Member Functions

- **HooksManager** ()
- void **init** ()
- void **fini** ()
- void **registerHook** (**HookType::hook_type_t** type, **HookCallbackFunc** func, **UInt64** argument, **Hook↩ CallbackOrder** order= **ORDER_NOTIFY_PRE**)
- **SInt64 callHooks** (**HookType::hook_type_t** type, **UInt64** argument, bool expect_return=false)

Private Attributes

- std::unordered_map< **HookType::hook_type_t**, std::vector< **HookCallback** > > **m_registry**

6.143.1 Detailed Description

Definition at line 56 of file hooks_manager.h.

6.143.2 Member Typedef Documentation

6.143.2.1 HookCallbackFunc

```
typedef SInt64(* HooksManager::HookCallbackFunc) ( UInt64, UInt64)
```

Definition at line 66 of file hooks_manager.h.

6.143.3 Member Enumeration Documentation

6.143.3.1 HookCallbackOrder

```
enum HooksManager::HookCallbackOrder
```

Enumerator

ORDER_NOTIFY_PRE	
ORDER_ACTION	
ORDER_NOTIFY_POST	
NUM_HOOK_ORDER	

Definition at line 59 of file hooks_manager.h.

6.143.4 Constructor & Destructor Documentation

6.143.4.1 HooksManager()

```
HooksManager::HooksManager ( )
```

Definition at line 34 of file hooks_manager.cc.

6.143.5 Member Function Documentation

6.143.5.1 callHooks()

```
SInt64 HooksManager::callHooks (
            HookType::hook_type_t type,
            UInt64 argument,
           bool expect_return = false )
```

Definition at line 43 of file hooks_manager.cc.

References m_registry, and NUM_HOOK_ORDER.

Referenced by Simulator::start(), and Simulator::∼Simulator().

6.143.5.2 fini()

```
void HooksManager::fini ( )
```

Definition at line 50 of file hooks_manager_init.cc.

References HooksPy::fini().

Referenced by Simulator::∼Simulator().

6.143.5.3 init()

```
void HooksManager::init ( )
```

Definition at line 44 of file hooks_manager_init.cc.

References HooksPy::init().

Referenced by Simulator::start().

6.143.5.4 registerHook()

```
void HooksManager::registerHook (
              HookType::hook_type_t type,
              HookCallbackFunc func,
              UInt64 argument,
              HookCallbackOrder order =  ORDER_NOTIFY_PRE )
```

Definition at line 38 of file hooks_manager.cc.

References m_registry.

6.143.6 Member Data Documentation

6.143.6.1 m_registry

```
std::unordered_map< HookType::hook_type_t, std::vector< HookCallback> > HooksManager::m_↩
registry  [private]
```

Definition at line 104 of file hooks_manager.h.

Referenced by callHooks(), and registerHook().

The documentation for this class was generated from the following files:

- common/system/ **hooks_manager.h**
- common/system/ **hooks_manager.cc**
- common/system/ **hooks_manager_init.cc**

6.144 HooksPy Class Reference

```
#include <hooks_py.h>
```

Classes

- class **PyBbv**
- class **PyConfig**
- class **PyControl**
- class **PyDvfs**
- class **PyHooks**
- class **PyMem**
- class **PyStats**
- class **PyThread**

Static Public Member Functions

- static void **init** (void)
- static void **setup** (void)
- static void **fini** (void)
- static PyObject * **callPythonFunction** (PyObject *pFunc, PyObject *pArgs)

Static Private Attributes

- static bool **pyInit** = false

6.144.1 Detailed Description

Definition at line 11 of file hooks_py.h.

6.144.2 Member Function Documentation

6.144.2.1 callPythonFunction()

```
PyObject * HooksPy::callPythonFunction (
            PyObject * pFunc,
            PyObject * pArgs )  [static]
```

Definition at line 71 of file hooks_py.cc.

Referenced by hookCallbackInt(), hookCallbackMagicMarkerType(), hookCallbackNone(), hookCallbackString(), hookCallbackSubsecondTime(), hookCallbackSyscallEnter(), hookCallbackSyscallExit(), hookCallbackThreadCreateType(), hookCallbackThreadMigrateType(), hookCallbackThreadResumeType(), hookCallbackThreadStallType(), hookCallbackThreadTimeType(), and statsCallback().

6.144.2.2 fini()

```
void HooksPy::fini (
            void  )  [static]
```

Definition at line 65 of file hooks_py.cc.

References pyInit.

Referenced by HooksManager::fini().

6.144.2.3 init()

```
void HooksPy::init (
            void  )  [static]
```

Definition at line 9 of file hooks_py.cc.

References itostr(), pyInit, and setup().

Referenced by HooksManager::init().

6.144.2.4 setup()

```
void HooksPy::setup (
            void  )  [static]
```

Definition at line 34 of file hooks_py.cc.

References LOG_ASSERT_ERROR, pyInit, HooksPy::PyConfig::setup(), HooksPy::PyStats::setup(), HooksPy::↩ PyHooks::setup(), HooksPy::PyDvfs::setup(), HooksPy::PyControl::setup(), HooksPy::PyBbv::setup(), HooksPy::↩ PyMem::setup(), and HooksPy::PyThread::setup().

Referenced by init().

6.144.3 Member Data Documentation

6.144.3.1 pyInit

```
bool HooksPy::pyInit = false  [static], [private]
```

Definition at line 19 of file hooks_py.h.

Referenced by fini(), init(), and setup().

The documentation for this class was generated from the following files:

- common/scripting/ **hooks_py.h**
- common/scripting/ **hooks_py.cc**

6.145 SyscallMdl::HookSyscallEnter Struct Reference

`#include <syscall_model.h>`

Public Attributes

- **thread_id_t thread_id**
- **core_id_t core_id**
- **SubsecondTime time**
- **IntPtr syscall_number**
- **syscall_args_t args**

6.145.1 Detailed Description

Definition at line 24 of file syscall_model.h.

6.145.2 Member Data Documentation

6.145.2.1 args

syscall_args_t SyscallMdl::HookSyscallEnter::args

Definition at line 30 of file syscall_model.h.

Referenced by hookCallbackSyscallEnter(), and SyscallMdl::runEnter().

6.145.2.2 core_id

core_id_t SyscallMdl::HookSyscallEnter::core_id

Definition at line 27 of file syscall_model.h.

Referenced by hookCallbackSyscallEnter(), and SyscallMdl::runEnter().

6.145.2.3 syscall_number

IntPtr SyscallMdl::HookSyscallEnter::syscall_number

Definition at line 29 of file syscall_model.h.

Referenced by hookCallbackSyscallEnter(), and SyscallMdl::runEnter().

6.145.2.4 thread_id

thread_id_t SyscallMdl::HookSyscallEnter::thread_id

Definition at line 26 of file syscall_model.h.

Referenced by hookCallbackSyscallEnter(), and SyscallMdl::runEnter().

6.145.2.5 time

SubsecondTime SyscallMdl::HookSyscallEnter::time

Definition at line 28 of file syscall_model.h.

Referenced by hookCallbackSyscallEnter(), and SyscallMdl::runEnter().

The documentation for this struct was generated from the following file:

- common/core/ **syscall_model.h**

6.146 SyscallMdl::HookSyscallExit Struct Reference

#include <syscall_model.h>

Public Attributes

- **thread_id_t thread_id**
- **core_id_t core_id**
- **SubsecondTime time**
- **IntPtr ret_val**
- bool **emulated**

6.146.1 Detailed Description

Definition at line 33 of file syscall_model.h.

6.146.2 Member Data Documentation

6.146.2.1 core_id

```
core_id_t SyscallMdl::HookSyscallExit::core_id
```

Definition at line 36 of file syscall_model.h.

Referenced by hookCallbackSyscallExit(), and SyscallMdl::runExit().

6.146.2.2 emulated

```
bool SyscallMdl::HookSyscallExit::emulated
```

Definition at line 39 of file syscall_model.h.

Referenced by hookCallbackSyscallExit(), and SyscallMdl::runExit().

6.146.2.3 ret_val

```
IntPtr SyscallMdl::HookSyscallExit::ret_val
```

Definition at line 38 of file syscall_model.h.

Referenced by hookCallbackSyscallExit(), and SyscallMdl::runExit().

6.146.2.4 thread_id

```
thread_id_t SyscallMdl::HookSyscallExit::thread_id
```

Definition at line 35 of file syscall_model.h.

Referenced by hookCallbackSyscallExit(), and SyscallMdl::runExit().

6.146.2.5 time

```
SubsecondTime SyscallMdl::HookSyscallExit::time
```

Definition at line 37 of file syscall_model.h.

Referenced by hookCallbackSyscallExit(), and SyscallMdl::runExit().

The documentation for this struct was generated from the following file:

- common/core/ **syscall_model.h**

6.147 HookType Class Reference

`#include <hooks_manager.h>`

Public Types

- enum **hook_type_t** {
  HOOK_PERIODIC, **HOOK_PERIODIC_INS**, **HOOK_SIM_START**, **HOOK_SIM_END**,
  HOOK_ROI_BEGIN, **HOOK_ROI_END**, **HOOK_CPUFREQ_CHANGE**, **HOOK_MAGIC_MARKER**,
  HOOK_MAGIC_USER, **HOOK_INSTR_COUNT**, **HOOK_THREAD_CREATE**, **HOOK_THREAD_START**,
  HOOK_THREAD_EXIT, **HOOK_THREAD_STALL**, **HOOK_THREAD_RESUME**, **HOOK_THREAD_MI↩ GRATE**,
  HOOK_INSTRUMENT_MODE, **HOOK_PRE_STAT_WRITE**, **HOOK_SYSCALL_ENTER**, **HOOK_SYS↩ CALL_EXIT**,
  HOOK_APPLICATION_START, **HOOK_APPLICATION_EXIT**, **HOOK_APPLICATION_ROI_BEGIN**, **H↩ OOK_APPLICATION_ROI_END**,
  HOOK_SIGUSR1, **HOOK_TYPES_MAX** }

Static Public Attributes

- static const char * **hook_type_names** [ ]

6.147.1 Detailed Description

Definition at line 11 of file hooks_manager.h.

6.147.2 Member Enumeration Documentation

6.147.2.1 hook_type_t

enum **HookType::hook_type_t**

Enumerator

HOOK_PERIODIC	
HOOK_PERIODIC_INS	
HOOK_SIM_START	
HOOK_SIM_END	
HOOK_ROI_BEGIN	
HOOK_ROI_END	
HOOK_CPUFREQ_CHANGE	
HOOK_MAGIC_MARKER	
HOOK_MAGIC_USER	
HOOK_INSTR_COUNT	
HOOK_THREAD_CREATE	
HOOK_THREAD_START	

Enumerator

HOOK_THREAD_EXIT	
HOOK_THREAD_STALL	
HOOK_THREAD_RESUME	
HOOK_THREAD_MIGRATE	
HOOK_INSTRUMENT_MODE	
HOOK_PRE_STAT_WRITE	
HOOK_SYSCALL_ENTER	
HOOK_SYSCALL_EXIT	
HOOK_APPLICATION_START	
HOOK_APPLICATION_EXIT	
HOOK_APPLICATION_ROI_BEGIN	
HOOK_APPLICATION_ROI_END	
HOOK_SIGUSR1	
HOOK_TYPES_MAX	

Definition at line 14 of file hooks_manager.h.

6.147.3 Member Data Documentation

6.147.3.1 hook_type_names

```
const char * HookType::hook_type_names  [static]
```

Initial value:

```
= {
   "HOOK_PERIODIC",
   "HOOK_PERIODIC_INS",
   "HOOK_SIM_START",
   "HOOK_SIM_END",
   "HOOK_ROI_BEGIN",
   "HOOK_ROI_END",
   "HOOK_CPUFREQ_CHANGE",
   "HOOK_MAGIC_MARKER",
   "HOOK_MAGIC_USER",
   "HOOK_INSTR_COUNT",
   "HOOK_THREAD_CREATE",
   "HOOK_THREAD_START",
   "HOOK_THREAD_EXIT",
   "HOOK_THREAD_STALL",
   "HOOK_THREAD_RESUME",
   "HOOK_THREAD_MIGRATE",
   "HOOK_INSTRUMENT_MODE",
   "HOOK_PRE_STAT_WRITE",
   "HOOK_SYSCALL_ENTER",
   "HOOK_SYSCALL_EXIT",
   "HOOK_APPLICATION_START",
   "HOOK_APPLICATION_EXIT",
   "HOOK_APPLICATION_ROI_BEGIN",
   "HOOK_APPLICATION_ROI_END",
   "HOOK_SIGUSR1",
}
```

Definition at line 43 of file hooks_manager.h.

Referenced by HooksPy::PyHooks::setup().

The documentation for this class was generated from the following files:

- common/system/ **hooks_manager.h**
- common/system/ **hooks_manager.cc**

6.148 NetworkModel::Hop Struct Reference

`#include <network_model.h>`

Public Attributes

- **SInt32 final_dest**
- **SInt32 next_dest**
- **subsecond_time_t time**

6.148.1 Detailed Description

Definition at line 33 of file network_model.h.

6.148.2 Member Data Documentation

6.148.2.1 final_dest

SInt32 NetworkModel::Hop::final_dest

Definition at line 35 of file network_model.h.

Referenced by NetworkModelEMeshHopByHop::addHop(), NetworkModelEMeshHopCounter::routePacket(), NetworkModelMagic::routePacket(), and NetworkModelBus::routePacket().

6.148.2.2 next_dest

SInt32 NetworkModel::Hop::next_dest

Definition at line 36 of file network_model.h.

Referenced by NetworkModelEMeshHopByHop::addHop(), NetworkModelEMeshHopCounter::routePacket(), NetworkModelMagic::routePacket(), and NetworkModelBus::routePacket().

6.148.2.3 time

subsecond_time_t NetworkModel::Hop::time

Definition at line 37 of file network_model.h.

Referenced by NetworkModelEMeshHopByHop::addHop(), NetworkModelEMeshHopCounter::routePacket(), NetworkModelMagic::routePacket(), and NetworkModelBus::routePacket().

The documentation for this struct was generated from the following file:

- common/network/ **network_model.h**

6.149 IndirectBranchTargetBuffer Class Reference

`#include <ibtb.h>`

Inheritance diagram for IndirectBranchTargetBuffer:

Public Member Functions

- **IndirectBranchTargetBuffer** (**UInt32** entries, **UInt32** tag_bitwidth)
- bool **predict** (**IntPtr** ip, **IntPtr** target)
- void **update** (bool predicted, bool actual, **IntPtr** ip, **IntPtr** target)

Private Member Functions

- void **gen_index_tag** (**IntPtr** ip, **UInt32** &index, **UInt32** &tag)

Private Attributes

- **UInt32 m_num_entries**
- **UInt32 m_tag_bitwidth**
- boost::scoped_array< uint8_t > **m_table**

Additional Inherited Members

6.149.1 Detailed Description

Definition at line 9 of file ibtb.h.

6.149.2 Constructor & Destructor Documentation

6.149.2.1 IndirectBranchTargetBuffer()

```
IndirectBranchTargetBuffer::IndirectBranchTargetBuffer (
            UInt32 entries,
            UInt32 tag_bitwidth )  [inline]
```

Definition at line 14 of file ibtb.h.

References m_table.

6.149.3 Member Function Documentation

6.149.3.1 gen_index_tag()

```
void IndirectBranchTargetBuffer::gen_index_tag (
              IntPtr ip,
              UInt32 & index,
              UInt32 & tag )  [inline], [private]
```

Definition at line 61 of file ibtb.h.

Referenced by predict(), and update().

6.149.3.2 predict()

```
bool IndirectBranchTargetBuffer::predict (
              IntPtr ip,
              IntPtr target )  [inline], [virtual]
```

Implements **BranchPredictor** (p. 125).

Definition at line 28 of file ibtb.h.

References gen_index_tag(), and m_table.

6.149.3.3 update()

```
void IndirectBranchTargetBuffer::update (
             bool predicted,
             bool actual,
              IntPtr ip,
              IntPtr target )  [inline], [virtual]
```

Implements **BranchPredictor** (p. 126).

Definition at line 44 of file ibtb.h.

References gen_index_tag(), and m_table.

6.149.4 Member Data Documentation

6.149.4.1 m_num_entries

```
UInt32 IndirectBranchTargetBuffer::m_num_entries [private]
```

Definition at line 57 of file ibtb.h.

6.149.4.2 m_table

```
boost::scoped_array<uint8_t> IndirectBranchTargetBuffer::m_table [private]
```

Definition at line 59 of file ibtb.h.

Referenced by IndirectBranchTargetBuffer(), predict(), and update().

6.149.4.3 m_tag_bitwidth

```
UInt32 IndirectBranchTargetBuffer::m_tag_bitwidth [private]
```

Definition at line 58 of file ibtb.h.

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **ibtb.h**

6.150 InstMode Class Reference

`#include <inst_mode.h>`

Public Types

- enum **inst_mode_t** { **INVALID** = 0, **DETAILED**, **CACHE_ONLY**, **FAST_FORWARD** }

Static Public Member Functions

- static **inst_mode_t** **fromString** (const String str)

Static Public Attributes

- static **inst_mode_t** **inst_mode_init** = **InstMode::INVALID**
- static **inst_mode_t** **inst_mode_roi** = **InstMode::INVALID**
- static **inst_mode_t** **inst_mode_end** = **InstMode::INVALID**

Static Private Member Functions

- static void **updateInstrumentationMode** ()

Static Private Attributes

- static **inst_mode_t** **inst_mode** = **InstMode::INVALID**

Friends

- class **Simulator**

6.150.1 Detailed Description

Definition at line 6 of file inst_mode.h.

6.150.2 Member Enumeration Documentation

6.150.2.1 inst_mode_t

enum **InstMode::inst_mode_t**

Enumerator

INVALID	
DETAILED	
CACHE_ONLY	
FAST_FORWARD	

Definition at line 9 of file inst_mode.h.

6.150.3 Member Function Documentation

6.150.3.1 fromString()

```
InstMode::inst_mode_t InstMode::fromString (
            const String str )  [static]
```

Definition at line 24 of file inst_mode.cc.

References CACHE_ONLY, DETAILED, FAST_FORWARD, and LOG_PRINT_ERROR.

Referenced by Simulator::start().

6.150.3.2 updateInstrumentationMode()

```
void InstMode::updateInstrumentationMode ( )  [static], [private]
```

Definition at line 6 of file inst_mode.cc.

Referenced by Simulator::setInstrumentationMode().

6.150.4 Friends And Related Function Documentation

6.150.4.1 Simulator

```
friend class Simulator [friend]
```

Definition at line 20 of file inst_mode.h.

6.150.5 Member Data Documentation

6.150.5.1 inst_mode

```
InstMode::inst_mode_t InstMode::inst_mode = InstMode::INVALID [static], [private]
```

Definition at line 16 of file inst_mode.h.

Referenced by Simulator::getInstrumentationMode(), and Simulator::setInstrumentationMode().

6.150.5.2 inst_mode_end

```
InstMode::inst_mode_t InstMode::inst_mode_end = InstMode::INVALID [static]
```

Definition at line 12 of file inst_mode.h.

Referenced by MagicServer::disablePerformance(), Simulator::printInstModeSummary(), and Simulator::start().

6.150.5.3 inst_mode_init

```
InstMode::inst_mode_t InstMode::inst_mode_init = InstMode::INVALID [static]
```

Definition at line 12 of file inst_mode.h.

Referenced by Simulator::printInstModeSummary(), and Simulator::start().

6.150.5.4 inst_mode_roi

```
InstMode::inst_mode_t InstMode::inst_mode_roi =  InstMode::INVALID  [static]
```

Definition at line 12 of file inst_mode.h.

Referenced by Simulator::disablePerformanceModels(), MagicServer::enablePerformance(), Simulator::enable↩ PerformanceModels(), Simulator::printInstModeSummary(), and Simulator::start().

The documentation for this class was generated from the following files:

- common/system/ **inst_mode.h**
- common/system/ **inst_mode.cc**

6.151 LoopTracer::Instr Class Reference

Public Member Functions

- **Instr** ()
- **Instr** (const **Instruction** ∗ **instruction**, **UInt8 uop_num**)

Public Attributes

- const **Instruction** ∗ **instruction**
- **UInt8 uop_num**
- std::map< uint64_t, int64_t > **issued**

6.151.1 Detailed Description

Definition at line 30 of file loop_tracer.h.

6.151.2 Constructor & Destructor Documentation

6.151.2.1 Instr() [1/2]

```
LoopTracer::Instr::Instr ( )  [inline]
```

Definition at line 32 of file loop_tracer.h.

6.151.2.2 Instr() [2/2]

```
LoopTracer::Instr::Instr (
            const  Instruction * instruction,
             UInt8 uop_num )  [inline]
```

Definition at line 33 of file loop_tracer.h.

6.151.3 Member Data Documentation

6.151.3.1 instruction

```
const  Instruction* LoopTracer::Instr::instruction
```

Definition at line 34 of file loop_tracer.h.

6.151.3.2 issued

```
std::map<uint64_t, int64_t> LoopTracer::Instr::issued
```

Definition at line 36 of file loop_tracer.h.

6.151.3.3 uop_num

```
 UInt8 LoopTracer::Instr::uop_num
```

Definition at line 35 of file loop_tracer.h.

The documentation for this class was generated from the following file:

- common/performance_model/instruction_tracers/ **loop_tracer.h**

6.152 InstrCountSampling Class Reference

```
#include <instr_count_sampling.h>
```

Inheritance diagram for InstrCountSampling:

Public Member Functions

- virtual void **startSampling** (**SubsecondTime** until)
- virtual **InstrumentLevel::Level** **requestedInstrumentation** ()

Additional Inherited Members

6.152.1 Detailed Description

Definition at line 6 of file instr_count_sampling.h.

6.152.2 Member Function Documentation

6.152.2.1 requestedInstrumentation()

```
virtual InstrumentLevel::Level InstrCountSampling::requestedInstrumentation ( ) [inline],
[virtual]
```

Implements **SamplingProvider** (p. 1117).

Definition at line 11 of file instr_count_sampling.h.

References InstrumentLevel::INSTR.

6.152.2.2 startSampling()

```
virtual void InstrCountSampling::startSampling (
            SubsecondTime until ) [inline], [virtual]
```

Implements **SamplingProvider** (p. 1117).

Definition at line 9 of file instr_count_sampling.h.

The documentation for this class was generated from the following file:

- common/sampling/ **instr_count_sampling.h**

6.153 Instruction Class Reference

```
#include <instruction.h>
```

Inheritance diagram for Instruction:

Public Member Functions

- **Instruction** (**InstructionType** type, **OperandList** &operands)
- **Instruction** (**InstructionType** type)
- virtual ~**Instruction** ()
- virtual **SubsecondTime** **getCost** (**Core** *core) const
- **InstructionType** **getType** () const
- String **getTypeName** () const
- bool **isPseudo** () const
- bool **isIdle** () const
- const **OperandList** & **getOperands** () const
- void **setAddress** (**IntPtr** addr)
- **IntPtr** **getAddress** () const
- void **setSize** (**UInt32** size)
- **UInt32** **getSize** () const
- void **setAtomic** (bool atomic)
- bool **isAtomic** () const
- void **setDisassembly** (String str)
- const String & **getDisassembly** (void) const
- void **setMicroOps** (const std::vector< const **MicroOp** * > *uops)
- const std::vector< const **MicroOp** * > * **getMicroOps** (void) const

Static Public Member Functions

- static void **initializeStaticInstructionModel** ()

Protected Attributes

- **OperandList** **m_operands**

Private Types

- typedef std::vector< unsigned int > **StaticInstructionCosts**

Private Attributes

- **InstructionType** **m_type**
- String **m_disas**
- const std::vector< const **MicroOp** * > * **m_uops**
- **IntPtr** **m_addr**
- **UInt32** **m_size**
- bool **m_atomic**

Static Private Attributes

- static **StaticInstructionCosts** **m_instruction_costs**

6.153.1 Detailed Description

Definition at line 40 of file instruction.h.

6.153.2 Member Typedef Documentation

6.153.2.1 StaticInstructionCosts

```
typedef std::vector<unsigned int>  Instruction::StaticInstructionCosts  [private]
```

Definition at line 81 of file instruction.h.

6.153.3 Constructor & Destructor Documentation

6.153.3.1 Instruction() [1/2]

```
Instruction::Instruction (
              InstructionType type,
              OperandList & operands )
```

Definition at line 13 of file instruction.cc.

6.153.3.2 Instruction() [2/2]

```
Instruction::Instruction (
              InstructionType type )
```

Definition at line 21 of file instruction.cc.

6.153.3.3 ~Instruction()

```
virtual Instruction::~Instruction ( )  [inline], [virtual]
```

Definition at line 48 of file instruction.h.

6.153.4 Member Function Documentation

6.153.4.1 getAddress()

```
IntPtr Instruction::getAddress ( ) const  [inline]
```

Definition at line 64 of file instruction.h.

References m_addr.

Referenced by DynamicInstruction::accessMemory(), InstructionModeling::handleInstruction(), MicroOp↩ PerformanceModel::handleInstruction(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), Interval↩ Timer::issueMemOp(), RobTimer::printRob(), RobSmtTimer::printRob(), LoopProfiler::traceInstruction(), and LoopTracer::traceInstruction().

6.153.4.2 getCost()

```
SubsecondTime Instruction::getCost (
            Core * core ) const  [virtual]
```

Reimplemented in **SpawnInstruction** (p. 1265), **SyncInstruction** (p. 1318), and **PseudoInstruction** (p. 948).

Definition at line 41 of file instruction.cc.

References Core::getDvfsDomain(), LOG_ASSERT_ERROR, m_instruction_costs, m_type, and MAX_INSTRU↩ CTION_COUNT.

Referenced by DynamicInstruction::getCost(), and OneIPCPerformanceModel::handleInstruction().

6.153.4.3 getDisassembly()

```
const String& Instruction::getDisassembly (
            void  ) const  [inline]
```

Definition at line 72 of file instruction.h.

References m_disas.

Referenced by MicroOpPerformanceModel::handleInstruction(), RobTimer::printRob(), RobSmtTimer::printRob(), MicroOp::toShortString(), and LoopTracer::traceInstruction().

6.153.4.4 getMicroOps()

```
const std::vector<const  MicroOp *>* Instruction::getMicroOps (
            void  ) const  [inline]
```

Definition at line 77 of file instruction.h.

References m_uops.

Referenced by MicroOpPerformanceModel::handleInstruction().

6.153.4.5 getOperands()

```
const OperandList& Instruction::getOperands ( ) const [inline]
```

Definition at line 60 of file instruction.h.

References m_operands.

Referenced by OneIPCPerformanceModel::handleInstruction(), and MicroOpPerformanceModel::handle↩ Instruction().

6.153.4.6 getSize()

```
UInt32 Instruction::getSize ( ) const [inline]
```

Definition at line 66 of file instruction.h.

References m_size.

Referenced by MicroOpPerformanceModel::handleInstruction().

6.153.4.7 getType()

```
InstructionType Instruction::getType ( ) const
```

Definition at line 28 of file instruction.cc.

References m_type.

Referenced by getTypeName(), PerformanceModel::handleIdleInstruction(), OneIPCPerformanceModel::handle↩ Instruction(), MicroOpPerformanceModel::handleInstruction(), isIdle(), OneIPCPerformanceModel::isModeled(), isPseudo(), PerformanceModel::queuePseudoInstruction(), and FastforwardPerformanceModel::queuePseudo↩ Instruction().

6.153.4.8 getTypeName()

```
String Instruction::getTypeName ( ) const
```

Definition at line 33 of file instruction.cc.

References getType(), LOG_ASSERT_ERROR, m_type, and MAX_INSTRUCTION_COUNT.

Referenced by MicroOpPerformanceModel::handleInstruction().

6.153.4.9 initializeStaticInstructionModel()

```
void Instruction::initializeStaticInstructionModel ( )  [static]
```

Definition at line 48 of file instruction.cc.

References m_instruction_costs, and MAX_INSTRUCTION_COUNT.

Referenced by Simulator::start().

6.153.4.10 isAtomic()

```
bool Instruction::isAtomic ( ) const  [inline]
```

Definition at line 69 of file instruction.h.

References m_atomic.

Referenced by DynamicInstruction::accessMemory().

6.153.4.11 isIdle()

```
bool Instruction::isIdle ( ) const  [inline]
```

Definition at line 55 of file instruction.h.

References getType(), INST_DELAY, INST_RECV, and INST_SYNC.

Referenced by PerformanceModel::iterate(), PerformanceModel::queuePseudoInstruction(), and Fastforward↩ PerformanceModel::queuePseudoInstruction().

6.153.4.12 isPseudo()

```
bool Instruction::isPseudo ( ) const  [inline]
```

Definition at line 53 of file instruction.h.

References getType(), and INST_PSEUDO_MISC.

Referenced by MicroOpPerformanceModel::handleInstruction(), OneIPCPerformanceModel::isModeled(), and DynamicInstruction::∼DynamicInstruction().

6.153.4.13 setAddress()

```
void Instruction::setAddress (
              IntPtr addr )  [inline]
```

Definition at line 63 of file instruction.h.

References m_addr.

Referenced by TraceThread::decode(), and InstructionModeling::decodeInstruction().

6.153.4.14 setAtomic()

```
void Instruction::setAtomic (
             bool atomic )  [inline]
```

Definition at line 68 of file instruction.h.

References m_atomic.

Referenced by TraceThread::decode(), and InstructionModeling::decodeInstruction().

6.153.4.15 setDisassembly()

```
void Instruction::setDisassembly (
             String str )  [inline]
```

Definition at line 71 of file instruction.h.

References m_disas.

Referenced by TraceThread::decode(), and InstructionModeling::decodeInstruction().

6.153.4.16 setMicroOps()

```
void Instruction::setMicroOps (
             const std::vector< const  MicroOp * > * uops )  [inline]
```

Definition at line 74 of file instruction.h.

References m_uops.

Referenced by TraceThread::decode(), and InstructionModeling::decodeInstruction().

6.153.4.17 setSize()

```
void Instruction::setSize (
              UInt32 size )  [inline]
```

Definition at line 65 of file instruction.h.

References m_size.

Referenced by TraceThread::decode(), and InstructionModeling::decodeInstruction().

6.153.5 Member Data Documentation

6.153.5.1 m_addr

```
IntPtr Instruction::m_addr  [private]
```

Definition at line 89 of file instruction.h.

Referenced by getAddress(), and setAddress().

6.153.5.2 m_atomic

```
bool Instruction::m_atomic  [private]
```

Definition at line 91 of file instruction.h.

Referenced by isAtomic(), and setAtomic().

6.153.5.3 m_disas

```
String Instruction::m_disas  [private]
```

Definition at line 85 of file instruction.h.

Referenced by getDisassembly(), and setDisassembly().

6.153.5.4 m_instruction_costs

```
Instruction::StaticInstructionCosts Instruction::m_instruction_costs [static], [private]
```

Definition at line 82 of file instruction.h.

Referenced by getCost(), and initializeStaticInstructionModel().

6.153.5.5 m_operands

```
OperandList Instruction::m_operands [protected]
```

Definition at line 94 of file instruction.h.

Referenced by getOperands().

6.153.5.6 m_size

```
UInt32 Instruction::m_size [private]
```

Definition at line 90 of file instruction.h.

Referenced by getSize(), and setSize().

6.153.5.7 m_type

```
InstructionType Instruction::m_type [private]
```

Definition at line 84 of file instruction.h.

Referenced by getCost(), getType(), and getTypeName().

6.153.5.8 m_uops

```
const std::vector<const MicroOp *>* Instruction::m_uops [private]
```

Definition at line 87 of file instruction.h.

Referenced by getMicroOps(), and setMicroOps().

The documentation for this class was generated from the following files:

- common/performance_model/ **instruction.h**
- common/performance_model/ **instruction.cc**

6.154 InstructionDecoder Class Reference

`#include <instruction_decoder.h>`

Static Public Member Functions

- static const std::vector< const **MicroOp** * > * **decode** (**IntPtr** address, const xed_decoded_inst_t *ins, **Instruction** *ins_ptr)
- static const std::vector< const **MicroOp** * > * **decode** (**IntPtr** address, const dl::DecodedInst *ins, **Instruction** *ins_ptr)

Static Private Member Functions

- static void **addSrcs** (std::set< xed_reg_enum_t > regs, **MicroOp** *uop)
- static void **addAddrs** (std::set< xed_reg_enum_t > regs, **MicroOp** *uop)
- static void **addDsts** (std::set< xed_reg_enum_t > regs, **MicroOp** *uop)
- static unsigned int **getNumExecs** (const xed_decoded_inst_t *ins, int numLoads, int numStores)
- static void **addSrcs** (std::set< dl::Decoder::decoder_reg > regs, **MicroOp** *uop)
- static void **addAddrs** (std::set< dl::Decoder::decoder_reg > regs, **MicroOp** *uop)
- static void **addDsts** (std::set< dl::Decoder::decoder_reg > regs, **MicroOp** *uop)
- static unsigned int **getNumExecs** (const dl::DecodedInst *ins, int numLoads, int numStores)

6.154.1 Detailed Description

Definition at line 16 of file instruction_decoder.h.

6.154.2 Member Function Documentation

6.154.2.1 addAddrs() [1/2]

```
void InstructionDecoder::addAddrs (
            std::set< dl::Decoder::decoder_reg > regs,
             MicroOp * uop )  [static], [private]
```

Definition at line 25 of file instruction_decoder_wlib.cc.

References MicroOp::addAddressRegister().

6.154.2.2 addAddrs() [2/2]

```
void InstructionDecoder::addAddrs (
            std::set< xed_reg_enum_t > regs,
             MicroOp * uop )  [static], [private]
```

Definition at line 22 of file instruction_decoder.cc.

References MicroOp::addAddressRegister().

Referenced by decode().

6.154.2.3 addDsts() [1/2]

```
void InstructionDecoder::addDsts (
            std::set< dl::Decoder::decoder_reg > regs,
             MicroOp * uop )  [static], [private]
```

Definition at line 36 of file instruction_decoder_wlib.cc.

References MicroOp::addDestinationRegister().

6.154.2.4 addDsts() [2/2]

```
void InstructionDecoder::addDsts (
            std::set< xed_reg_enum_t > regs,
             MicroOp * uop )  [static], [private]
```

Definition at line 31 of file instruction_decoder.cc.

References MicroOp::addDestinationRegister().

Referenced by decode().

6.154.2.5 addSrcs() [1/2]

```
void InstructionDecoder::addSrcs (
            std::set< dl::Decoder::decoder_reg > regs,
             MicroOp * uop )  [static], [private]
```

Definition at line 14 of file instruction_decoder_wlib.cc.

References MicroOp::addSourceRegister().

6.154.2.6 addSrcs() [2/2]

```
void InstructionDecoder::addSrcs (
            std::set< xed_reg_enum_t > regs,
             MicroOp * uop )  [static], [private]
```

Definition at line 13 of file instruction_decoder.cc.

References MicroOp::addSourceRegister().

Referenced by decode().

6.154.2.7 decode() [1/2]

```
const std::vector< const  MicroOp * > * InstructionDecoder::decode (
             IntPtr address,
            const dl::DecodedInst * ins,
             Instruction * ins_ptr )  [static]
```

Definition at line 57 of file instruction_decoder_wlib.cc.

References addAddrs(), addDsts(), addSrcs(), getNumExecs(), LOG_ASSERT_ERROR, Memory::make_access(), MicroOp::makeExecute(), MicroOp::makeLoad(), MicroOp::makeStore(), MicroOp::setDecodedInstruction(), MicroOp::setFirst(), MicroOp::setInstruction(), MicroOp::setInstructionPointer(), MicroOp::setInterrupt(), MicroOp::setIsX87(), MicroOp::setLast(), MicroOp::setMemBarrier(), MicroOp::setOperandSize(), and MicroOp::setSerializing().

6.154.2.8 decode() [2/2]

```
const std::vector< const  MicroOp * > * InstructionDecoder::decode (
             IntPtr address,
            const xed_decoded_inst_t * ins,
             Instruction * ins_ptr )  [static]
```

Definition at line 91 of file instruction_decoder.cc.

References addAddrs(), addDsts(), addSrcs(), getNumExecs(), LOG_ASSERT_ERROR, Memory::make_access(), MicroOp::makeExecute(), MicroOp::makeLoad(), MicroOp::makeStore(), MicroOp::setFirst(), MicroOp::setInstruction(), MicroOp::setInstructionPointer(), MicroOp::setInterrupt(), MicroOp::setIsX87(), MicroOp::setLast(), MicroOp::setMemBarrier(), MicroOp::setOperandSize(), and MicroOp::setSerializing().

Referenced by TraceThread::decode(), and InstructionModeling::decodeInstruction().

6.154.2.9 getNumExecs() [1/2]

```
unsigned int InstructionDecoder::getNumExecs (
            const dl::DecodedInst * ins,
            int numLoads,
            int numStores )  [static], [private]
```

Definition at line 47 of file instruction_decoder_wlib.cc.

6.154.2.10 getNumExecs() [2/2]

```
unsigned int InstructionDecoder::getNumExecs (
            const xed_decoded_inst_t * ins,
            int numLoads,
            int numStores )  [static], [private]
```

Definition at line 40 of file instruction_decoder.cc.

Referenced by decode().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/micro_op/ **instruction_decoder.h**
- common/performance_model/performance_models/micro_op/ **instruction_decoder_wlib.h**
- common/performance_model/performance_models/micro_op/ **instruction_decoder.cc**
- common/performance_model/performance_models/micro_op/ **instruction_decoder_wlib.cc**

6.155 InstructionModeling Class Reference

`#include <instruction_modeling.h>`

Static Public Member Functions

- static VOID **addInstructionModeling** (TRACE trace, INS ins, **InstMode::inst_mode_t** inst_mode)
- static **Instruction** * **decodeInstruction** (INS ins)
- static void **handleInstruction** (THREADID thread_id, **Instruction** *instruction)
- static void **handleBasicBlock** (THREADID thread_id)
- static VOID **countInstructions** (THREADID threadid, ADDRINT address, INT32 count)
- static VOID **accessInstructionCacheWarmup** (THREADID threadid, ADDRINT address, UINT32 size)

6.155.1 Detailed Description

Definition at line 10 of file instruction_modeling.h.

6.155.2 Member Function Documentation

6.155.2.1 accessInstructionCacheWarmup()

```
VOID InstructionModeling::accessInstructionCacheWarmup (
            THREADID threadid,
            ADDRINT address,
            UINT32 size )  [static]
```

Definition at line 369 of file instruction_modeling.cc.

References Core::accessMemoryFast(), localStore, and Core::READ.

Referenced by traceCallback().

6.155.2.2 addInstructionModeling()

```
VOID InstructionModeling::addInstructionModeling (
            TRACE trace,
            INS ins,
             InstMode::inst_mode_t inst_mode )  [static]
```

Definition at line 172 of file instruction_modeling.cc.

References lite::addMemoryModeling(), addSpinLoopDetection(), decodeInstruction(), handleBranch(), handle↩ BranchWarming(), handleCpuid(), handleInstruction(), handleMagic(), handlePause(), handleRdtsc(), lite::handle↩ Syscall(), INSTR_IF_CACHEONLY, INSTR_IF_DETAILED, INSTRUMENT, and INSTRUMENT_PREDICATED.

Referenced by traceCallback().

6.155.2.3 countInstructions()

```
VOID InstructionModeling::countInstructions (
            THREADID threadid,
            ADDRINT address,
            INT32 count )  [static]
```

Definition at line 342 of file instruction_modeling.cc.

References Core::countInstructions(), Core::getPerformanceModel(), localStore, PerformanceModel::queue↩ PseudoInstruction(), and SyncInstruction::UNSCHEDULED.

Referenced by traceCallback().

6.155.2.4 decodeInstruction()

```
Instruction * InstructionModeling::decodeInstruction (
            INS ins )  [static]
```

Definition at line 292 of file instruction_modeling.cc.

References InstructionDecoder::decode(), fillOperandList(), INST_DIV, INST_FDIV, INST_FMUL, INST_MUL, Instruction::setAddress(), Instruction::setAtomic(), Instruction::setDisassembly(), Instruction::setMicroOps(), and Instruction::setSize().

Referenced by addInstructionModeling().

6.155.2.5 handleBasicBlock()

```
void InstructionModeling::handleBasicBlock (
            THREADID thread_id )  [static]
```

Definition at line 38 of file instruction_modeling.cc.

References Thread::getCore(), PerformanceModel::getElapsedTime(), Core::getPerformanceModel(), PerformanceModel::iterate(), localStore, PerformanceModel::queueInstruction(), and Thread::reschedule().

Referenced by traceCallback().

6.155.2.6 handleInstruction()

```
void InstructionModeling::handleInstruction (
            THREADID thread_id,
             Instruction * instruction )  [static]
```

Definition at line 26 of file instruction_modeling.cc.

References PerformanceModel::createDynamicInstruction(), Instruction::getAddress(), Thread::getCore(), Core::getPerformanceModel(), localStore, and PerformanceModel::queueInstruction().

Referenced by addInstructionModeling().

The documentation for this class was generated from the following files:

- pin/ **instruction_modeling.h**
- pin/ **instruction_modeling.cc**

6.156 InstructionTracer Class Reference

`#include <instruction_tracer.h>`

Inheritance diagram for InstructionTracer:

Classes

- struct **uop_times_t**

Public Member Functions

- virtual ~**InstructionTracer** ()
- virtual void **traceInstruction** (const **DynamicMicroOp** *uop, **uop_times_t** *times)=0

Static Public Member Functions

- static void **init** ()
- static **InstructionTracer** * **create** (const **Core** *core)

6.156.1 Detailed Description

Definition at line 10 of file instruction_tracer.h.

6.156.2 Constructor & Destructor Documentation

6.156.2.1 ~InstructionTracer()

`virtual InstructionTracer::~InstructionTracer ( ) [inline], [virtual]`

Definition at line 16 of file instruction_tracer.h.

6.156.3 Member Function Documentation

6.156.3.1 create()

```
InstructionTracer * InstructionTracer::create (
            const  Core * core )  [static]
```

Definition at line 19 of file instruction_tracer.cc.

References LOG_PRINT_ERROR.

Referenced by PerformanceModel::PerformanceModel().

6.156.3.2 init()

```
void InstructionTracer::init ( )  [static]
```

Definition at line 10 of file instruction_tracer.cc.

References InstructionTracerFPStats::init().

Referenced by Simulator::start().

6.156.3.3 traceInstruction()

```
virtual void InstructionTracer::traceInstruction (
            const  DynamicMicroOp * uop,
             uop_times_t * times )  [pure virtual]
```

Implemented in **LoopTracer** (p. 704), **LoopProfiler** (p. 701), **InstructionTracerFPStats** (p. 637), and **InstructionTracerPrint** (p. 639).

Referenced by PerformanceModel::traceInstruction().

The documentation for this class was generated from the following files:

- common/performance_model/instruction_tracers/ **instruction_tracer.h**
- common/performance_model/instruction_tracers/ **instruction_tracer.cc**

6.157 InstructionTracerFPStats Class Reference

```
#include <instruction_tracer_fpstats.h>
```

Inheritance diagram for InstructionTracerFPStats:

Public Member Functions

- **InstructionTracerFPStats** (const **Core** *core)
- virtual void **traceInstruction** (const **DynamicMicroOp** *uop, **uop_times_t** *times)

Static Public Member Functions

- static void **init** ()

Private Attributes

- const **Core** * **m_core**
- std::unordered_map< int, uint64_t > **m_iclasses**

6.157.1 Detailed Description

Definition at line 12 of file instruction_tracer_fpstats.h.

6.157.2 Constructor & Destructor Documentation

6.157.2.1 InstructionTracerFPStats()

```
InstructionTracerFPStats::InstructionTracerFPStats (
            const  Core * core )
```

Definition at line 16 of file instruction_tracer_fpstats.cc.

References fp_iclasses, Core::getId(), m_iclasses, and registerStatsMetric().

6.157.3 Member Function Documentation

6.157.3.1 init()

```
void InstructionTracerFPStats::init ( )  [static]
```

Definition at line 32 of file instruction_tracer_fpstats.cc.

References fp_iclasses, and ThreadStatNamedStat::registerStat().

Referenced by InstructionTracer::init().

6.157.3.2 traceInstruction()

```
void InstructionTracerFPStats::traceInstruction (
            const  DynamicMicroOp * uop,
             uop_times_t * times )  [virtual]
```

Implements **InstructionTracer** (p. 636).

Definition at line 41 of file instruction_tracer_fpstats.cc.

References MicroOp::getInstructionOpcode(), DynamicMicroOp::getMicroOp(), MicroOp::isFirst(), and m_iclasses.

6.157.4 Member Data Documentation

6.157.4.1 m_core

```
const  Core* InstructionTracerFPStats::m_core  [private]
```

Definition at line 20 of file instruction_tracer_fpstats.h.

6.157.4.2 m_iclasses

```
std::unordered_map<int, uint64_t> InstructionTracerFPStats::m_iclasses  [private]
```

Definition at line 21 of file instruction_tracer_fpstats.h.

Referenced by InstructionTracerFPStats(), and traceInstruction().

The documentation for this class was generated from the following files:

- common/performance_model/instruction_tracers/ **instruction_tracer_fpstats.h**
- common/performance_model/instruction_tracers/ **instruction_tracer_fpstats.cc**

6.158 InstructionTracerPrint Class Reference

```
#include <instruction_tracer_print.h>
```

Inheritance diagram for InstructionTracerPrint:

Public Member Functions

- **InstructionTracerPrint** (const **Core** *core)
- virtual void **traceInstruction** (const **DynamicMicroOp** *uop, **uop_times_t** *times)

Private Attributes

- const **Core** * **m_core**

Additional Inherited Members

6.158.1 Detailed Description

Definition at line 6 of file instruction_tracer_print.h.

6.158.2 Constructor & Destructor Documentation

6.158.2.1 InstructionTracerPrint()

```
InstructionTracerPrint::InstructionTracerPrint (
            const  Core * core )  [inline]
```

Definition at line 9 of file instruction_tracer_print.h.

6.158.3 Member Function Documentation

6.158.3.1 traceInstruction()

```
void InstructionTracerPrint::traceInstruction (
            const  DynamicMicroOp * uop,
             uop_times_t * times )  [virtual]
```

Implements **InstructionTracer** (p. 636).

Definition at line 9 of file instruction_tracer_print.cc.

References Core::getId(), MicroOp::getInstructionOpcode(), DynamicMicroOp::getMicroOp(), and m_core.

6.158.4 Member Data Documentation

6.158.4.1 m_core

```
const Core* InstructionTracerPrint::m_core [private]
```

Definition at line 15 of file instruction_tracer_print.h.

Referenced by traceInstruction().

The documentation for this class was generated from the following files:

- common/performance_model/instruction_tracers/ **instruction_tracer_print.h**
- common/performance_model/instruction_tracers/ **instruction_tracer_print.cc**

6.159 IntervalContention Class Reference

```
#include <interval_contention.h>
```

Inheritance diagram for IntervalContention:

Public Member Functions

- virtual void **clearFunctionalUnitStats** ()=0
- virtual void **addFunctionalUnitStats** (const **DynamicMicroOp** *uop)=0
- virtual void **removeFunctionalUnitStats** (const **DynamicMicroOp** *uop)=0
- virtual uint64_t **getEffectiveCriticalPathLength** (uint64_t critical_path_length, bool update_reason)=0

Static Public Member Functions

- static **IntervalContention** * **createIntervalContentionModel** (**Core** *core, const **CoreModel** *core_↩ model)

6.159.1 Detailed Description

Definition at line 14 of file interval_contention.h.

6.159.2 Member Function Documentation

6.159.2.1 addFunctionalUnitStats()

```
virtual void IntervalContention::addFunctionalUnitStats (
            const  DynamicMicroOp * uop )  [pure virtual]
```

Implemented in **IntervalContentionBoomV1** (p. 643), and **IntervalContentionNehalem** (p. 645).

Referenced by Windows::addFunctionalUnitStats().

6.159.2.2 clearFunctionalUnitStats()

```
virtual void IntervalContention::clearFunctionalUnitStats ( )  [pure virtual]
```

Implemented in **IntervalContentionBoomV1** (p. 643), and **IntervalContentionNehalem** (p. 645).

Referenced by Windows::clearFunctionalUnitStats().

6.159.2.3 createIntervalContentionModel()

```
static  IntervalContention* IntervalContention::createIntervalContentionModel (
             Core * core,
            const  CoreModel * core_model )  [static]
```

6.159.2.4 getEffectiveCriticalPathLength()

```
virtual uint64_t IntervalContention::getEffectiveCriticalPathLength (
            uint64_t critical_path_length,
            bool update_reason )  [pure virtual]
```

Implemented in **IntervalContentionBoomV1** (p. 643), and **IntervalContentionNehalem** (p. 646).

Referenced by Windows::getEffectiveCriticalPathLength().

6.159.2.5 removeFunctionalUnitStats()

```
virtual void IntervalContention::removeFunctionalUnitStats (
            const  DynamicMicroOp * uop )  [pure virtual]
```

Implemented in **IntervalContentionBoomV1** (p. 643), and **IntervalContentionNehalem** (p. 646).

Referenced by Windows::removeFunctionalUnitStats().

The documentation for this class was generated from the following file:

- common/performance_model/performance_models/interval_performance_model/ **interval_contention.h**

6.160 IntervalContentionBoomV1 Class Reference

`#include <interval_contention_boom_v1.h>`

Inheritance diagram for IntervalContentionBoomV1:

Public Member Functions

- **IntervalContentionBoomV1** (const **Core** *core, const **CoreModel** *core_model)
- virtual void **clearFunctionalUnitStats** ()
- virtual void **addFunctionalUnitStats** (const **DynamicMicroOp** *uop)
- virtual void **removeFunctionalUnitStats** (const **DynamicMicroOp** *uop)
- virtual uint64_t **getEffectiveCriticalPathLength** (uint64_t critical_path_length, bool update_reason)

Private Attributes

- const **CoreModel** * **m_core_model**
- uint64_t **m_count_byport** [8]
- uint64_t **m_cpContrByPort** [**DynamicMicroOpBoomV1::UOP_PORT_SIZE**]

Additional Inherited Members

6.160.1 Detailed Description

Definition at line 11 of file interval_contention_boom_v1.h.

6.160.2 Constructor & Destructor Documentation

6.160.2.1 IntervalContentionBoomV1()

```
IntervalContentionBoomV1::IntervalContentionBoomV1 (
            const  Core * core,
            const  CoreModel * core_model )
```

Definition at line 9 of file interval_contention_boom_v1.cc.

References Core::getId(), m_cpContrByPort, DynamicMicroOpBoomV1::PortTypeString(), registerStatsMetric(), and DynamicMicroOpBoomV1::UOP_PORT_SIZE.

6.160.3 Member Function Documentation

6.160.3.1 addFunctionalUnitStats()

```
void IntervalContentionBoomV1::addFunctionalUnitStats (
            const  DynamicMicroOp * uop )  [virtual]
```

Implements **IntervalContention** (p. 640).

Definition at line 28 of file interval_contention_boom_v1.cc.

References DynamicMicroOp::getCoreSpecificInfo(), and m_count_byport.

6.160.3.2 clearFunctionalUnitStats()

```
void IntervalContentionBoomV1::clearFunctionalUnitStats ( )  [virtual]
```

Implements **IntervalContention** (p. 641).

Definition at line 20 of file interval_contention_boom_v1.cc.

References m_count_byport, and DynamicMicroOpBoomV1::UOP_PORT_SIZE.

6.160.3.3 getEffectiveCriticalPathLength()

```
uint64_t IntervalContentionBoomV1::getEffectiveCriticalPathLength (
            uint64_t critical_path_length,
            bool update_reason )  [virtual]
```

Implements **IntervalContention** (p. 641).

Definition at line 38 of file interval_contention_boom_v1.cc.

References LOG_ASSERT_ERROR, m_count_byport, m_cpContrByPort, DynamicMicroOpBoomV1::UOP_P↩ ORT0, DynamicMicroOpBoomV1::UOP_PORT012, DynamicMicroOpBoomV1::UOP_PORT1, DynamicMicroOp↩ BoomV1::UOP_PORT2, and DynamicMicroOpBoomV1::UOP_PORT_SIZE.

6.160.3.4 removeFunctionalUnitStats()

```
void IntervalContentionBoomV1::removeFunctionalUnitStats (
            const  DynamicMicroOp * uop )  [virtual]
```

Implements **IntervalContention** (p. 641).

Definition at line 33 of file interval_contention_boom_v1.cc.

References DynamicMicroOp::getCoreSpecificInfo(), and m_count_byport.

6.160.4 Member Data Documentation

6.160.4.1 m_core_model

```
const CoreModel* IntervalContentionBoomV1::m_core_model [private]
```

Definition at line 13 of file interval_contention_boom_v1.h.

6.160.4.2 m_count_byport

```
uint64_t IntervalContentionBoomV1::m_count_byport[8] [private]
```

Definition at line 15 of file interval_contention_boom_v1.h.

Referenced by addFunctionalUnitStats(), clearFunctionalUnitStats(), getEffectiveCriticalPathLength(), and removeFunctionalUnitStats().

6.160.4.3 m_cpContrByPort

```
uint64_t IntervalContentionBoomV1::m_cpContrByPort[ DynamicMicroOpBoomV1::UOP_PORT_SIZE] [private]
```

Definition at line 16 of file interval_contention_boom_v1.h.

Referenced by getEffectiveCriticalPathLength(), and IntervalContentionBoomV1().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/interval_performance_model/ **interval_contention_↩ boom_v1.h**
- common/performance_model/performance_models/interval_performance_model/ **interval_contention_↩ boom_v1.cc**

6.161 IntervalContentionNehalem Class Reference

```
#include <interval_contention_nehalem.h>
```

Inheritance diagram for IntervalContentionNehalem:

Public Member Functions

- **IntervalContentionNehalem** (const **Core** *core, const **CoreModel** *core_model)
- virtual void **clearFunctionalUnitStats** ()
- virtual void **addFunctionalUnitStats** (const **DynamicMicroOp** *uop)
- virtual void **removeFunctionalUnitStats** (const **DynamicMicroOp** *uop)
- virtual uint64_t **getEffectiveCriticalPathLength** (uint64_t critical_path_length, bool update_reason)

Private Attributes

- const **CoreModel** * **m_core_model**
- uint64_t **m_count_byport** [8]
- uint64_t **m_cpContrByPort** [**DynamicMicroOpNehalem::UOP_PORT_SIZE**]

Additional Inherited Members

6.161.1 Detailed Description

Definition at line 11 of file interval_contention_nehalem.h.

6.161.2 Constructor & Destructor Documentation

6.161.2.1 IntervalContentionNehalem()

```
IntervalContentionNehalem::IntervalContentionNehalem (
            const  Core * core,
            const  CoreModel * core_model )
```

Definition at line 9 of file interval_contention_nehalem.cc.

References Core::getId(), m_cpContrByPort, DynamicMicroOpNehalem::PortTypeString(), registerStatsMetric(), and DynamicMicroOpNehalem::UOP_PORT_SIZE.

6.161.3 Member Function Documentation

6.161.3.1 addFunctionalUnitStats()

```
void IntervalContentionNehalem::addFunctionalUnitStats (
            const  DynamicMicroOp * uop )  [virtual]
```

Implements **IntervalContention** (p. 640).

Definition at line 28 of file interval_contention_nehalem.cc.

References DynamicMicroOp::getCoreSpecificInfo(), and m_count_byport.

6.161.3.2 clearFunctionalUnitStats()

```
void IntervalContentionNehalem::clearFunctionalUnitStats ( )  [virtual]
```

Implements **IntervalContention** (p. 641).

Definition at line 20 of file interval_contention_nehalem.cc.

References m_count_byport, and DynamicMicroOpNehalem::UOP_PORT_SIZE.

6.161.3.3 getEffectiveCriticalPathLength()

```
uint64_t IntervalContentionNehalem::getEffectiveCriticalPathLength (
            uint64_t critical_path_length,
            bool update_reason )  [virtual]
```

Implements **IntervalContention** (p. 641).

Definition at line 38 of file interval_contention_nehalem.cc.

References LOG_ASSERT_ERROR, m_count_byport, m_cpContrByPort, DynamicMicroOpNehalem::UOP_PO↩
RT0, DynamicMicroOpNehalem::UOP_PORT015, DynamicMicroOpNehalem::UOP_PORT05, DynamicMicroOp↩
Nehalem::UOP_PORT1, DynamicMicroOpNehalem::UOP_PORT5, and DynamicMicroOpNehalem::UOP_PORT↩
_SIZE.

6.161.3.4 removeFunctionalUnitStats()

```
void IntervalContentionNehalem::removeFunctionalUnitStats (
            const  DynamicMicroOp * uop )  [virtual]
```

Implements **IntervalContention** (p. 641).

Definition at line 33 of file interval_contention_nehalem.cc.

References DynamicMicroOp::getCoreSpecificInfo(), and m_count_byport.

6.161.4 Member Data Documentation

6.161.4.1 m_core_model

```
const  CoreModel* IntervalContentionNehalem::m_core_model  [private]
```

Definition at line 13 of file interval_contention_nehalem.h.

6.161.4.2 m_count_byport

```
uint64_t IntervalContentionNehalem::m_count_byport[8]  [private]
```

Definition at line 15 of file interval_contention_nehalem.h.

Referenced by addFunctionalUnitStats(), clearFunctionalUnitStats(), getEffectiveCriticalPathLength(), and removeFunctionalUnitStats().

6.161.4.3 m_cpContrByPort

```
uint64_t IntervalContentionNehalem::m_cpContrByPort[ DynamicMicroOpNehalem::UOP_PORT_SIZE]
[private]
```

Definition at line 16 of file interval_contention_nehalem.h.

Referenced by getEffectiveCriticalPathLength(), and IntervalContentionNehalem().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/interval_performance_model/ **interval_contention_↩ nehalem.h**
- common/performance_model/performance_models/interval_performance_model/ **interval_contention_↩ nehalem.cc**

6.162 IntervalPerformanceModel Class Reference

```
#include <interval_performance_model.h>
```

Inheritance diagram for IntervalPerformanceModel:

Public Member Functions

- **IntervalPerformanceModel** (**Core** ∗core, int misprediction_penalty)
- ∼**IntervalPerformanceModel** ()

Protected Member Functions

- virtual boost::tuple< uint64_t, uint64_t > **simulate** (const std::vector< **DynamicMicroOp** ∗ > &insts)
- virtual void **notifyElapsedTimeUpdate** ()

Private Attributes

- **IntervalTimer interval_timer**

Additional Inherited Members

6.162.1 Detailed Description

Definition at line 6 of file interval_performance_model.h.

6.162.2 Constructor & Destructor Documentation

6.162.2.1 IntervalPerformanceModel()

```
IntervalPerformanceModel::IntervalPerformanceModel (
              Core * core,
             int mispredict_penalty )
```

Definition at line 6 of file interval_performance_model.cc.

6.162.2.2 ~IntervalPerformanceModel()

```
IntervalPerformanceModel::~IntervalPerformanceModel ( )
```

Definition at line 19 of file interval_performance_model.cc.

References IntervalTimer::free(), and interval_timer.

6.162.3 Member Function Documentation

6.162.3.1 notifyElapsedTimeUpdate()

```
void IntervalPerformanceModel::notifyElapsedTimeUpdate ( )  [protected], [virtual]
```

Implements **MicroOpPerformanceModel** (p. 788).

Definition at line 29 of file interval_performance_model.cc.

References ComponentTime::getCycleCount(), interval_timer, PerformanceModel::m_elapsed_time, and IntervalTimer::synchronize().

6.162.3.2 simulate()

```
boost::tuple< uint64_t, uint64_t > IntervalPerformanceModel::simulate (
            const std::vector<  DynamicMicroOp * > & insts )  [protected], [virtual]
```

Implements **MicroOpPerformanceModel** (p. 789).

Definition at line 24 of file interval_performance_model.cc.

References interval_timer, and IntervalTimer::simulate().

6.162.4 Member Data Documentation

6.162.4.1 interval_timer

```
IntervalTimer IntervalPerformanceModel::interval_timer  [private]
```

Definition at line 17 of file interval_performance_model.h.

Referenced by notifyElapsedTimeUpdate(), simulate(), and ~IntervalPerformanceModel().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/ **interval_performance_model.h**
- common/performance_model/performance_models/ **interval_performance_model.cc**

6.163 IntervalTimer Class Reference

```
#include <interval_timer.h>
```

Public Member Functions

- **IntervalTimer** (**Core** *core, **PerformanceModel** *perf, const **CoreModel** *core_model, int misprediction↩ _penalty, int dispatch_width, int window_size, bool do_functional_unit_contention)
- ~**IntervalTimer** ()
- void **free** ()
- boost::tuple< uint64_t, uint64_t > **simulate** (const std::vector< **DynamicMicroOp** * > &insts)
- void **synchronize** (uint64_t time)

Protected Member Functions

- boost::tuple< uint64_t, uint64_t > **dispatchWindow** ()
- uint32_t **calculateCurrentDispatchRate** ()
- void **issueMemOp** (**Windows::WindowEntry** &micro_op)
- uint64_t **dispatchInstruction** (**Windows::WindowEntry** &instruction, **StopDispatchReason** &continue↩ Dispatching)
- void **updateCriticalPath** (**Windows::WindowEntry** &microOp, uint64_t &latency)
- void **blockWindow** ()
- uint64_t **getMaxProducerExecTime** (**Windows::WindowEntry** &instruction)

Private Attributes

- **Core** * **m_core**
- const **CoreModel** * **m_core_model**
- const uint32_t **m_dispatch_width**
- const uint32_t **m_branch_misprediction_penalty**
- **UInt64** **m_mem_dep_mask**
- **UInt64** **m_lll_dep_mask**
- **FixedPoint** **m_remaining_dispatch_bandwidth**
- uint64_t **m_max_store_completion_time**
- uint64_t **m_max_load_completion_time**
- **ContentionModel** **m_loadstore_contention**
- **Windows** * **m_windows**
- **PerformanceModel** * **m_perf_model**
- const **ComponentPeriod** * **m_frequency_domain**
- **UInt64** **m_uop_type_count** [**MicroOp::UOP_SUBTYPE_SIZE**]
- **UInt64** **m_uops_total**
- **UInt64** **m_uops_x87**
- **UInt64** **m_uops_pause**
- uint64_t **m_numICacheOverlapped**
- uint64_t **m_numBPredOverlapped**
- uint64_t **m_numDCacheOverlapped**
- uint64_t **m_numLongLatencyLoads**
- uint64_t **m_numTotalLongLatencyLoadLatency**
- uint64_t **m_numSerializationInsns**
- uint64_t **m_totalSerializationLatency**
- uint64_t **m_totalHiddenDCacheLatency**
- uint64_t **m_totalHiddenLongerDCacheLatency**
- uint64_t **m_numHiddenLongerDCacheLatency**
- uint64_t **m_outstandingLongLatencyInsns**
- uint64_t **m_outstandingLongLatencyCycles**
- uint64_t **m_lastAccountedMemoryCycle**
- uint64_t **m_numMfenceInsns**
- uint64_t **m_totalMfenceLatency**
- **SubsecondTime** **m_cpiBase**
- **SubsecondTime** **m_cpiBranchPredictor**
- **SubsecondTime** **m_cpiSerialization**
- **SubsecondTime** **m_cpiLongLatency**
- std::vector< **SubsecondTime** > **m_cpiInstructionCache**
- std::vector< **SubsecondTime** > **m_cpiDataCache**
- uint64_t **m_cpiBaseStopDispatch** [**STOP_DISPATCH_SIZE**]
- uint64_t **m_cpContrByType** [**CPCONTR_TYPE_SIZE**]

6.163.1 Detailed Description

Definition at line 36 of file interval_timer.h.

6.163.2 Constructor & Destructor Documentation

6.163.2.1 IntervalTimer()

```
IntervalTimer::IntervalTimer (
              Core * core,
              PerformanceModel * perf,
             const  CoreModel * core_model,
             int misprediction_penalty,
             int dispatch_width,
             int window_size,
             bool do_functional_unit_contention )
```

Definition at line 23 of file interval_timer.cc.

References CPCONTR_TYPE_SIZE, CpContrTypeString(), Core::getId(), MicroOp::getSubtypeString(), HitWhereIsValid(), HitWhereString(), isPower2(), itostr(), LOG_ASSERT_ERROR, m_cpContrByType, m_cpiBase, m_cpiBaseStopDispatch, m_cpiBranchPredictor, m_cpiDataCache, m_cpiInstructionCache, m_cpiLongLatency, m_cpiSerialization, m_lastAccountedMemoryCycle, m_lll_dep_mask, m_mem_dep_mask, m_numBPredOverlapped, m_numDCacheOverlapped, m_numHiddenLongerDCacheLatency, m_numICacheOverlapped, m_numLongLatencyLoads, m_numMfenceInsns, m_numSerializationInsns, m_numTotalLongLatencyLoadLatency, m_outstandingLongLatencyCycles, m_outstandingLongLatencyInsns, m_totalHiddenDCacheLatency, m_totalHiddenLongerDCacheLatency, m_totalMfenceLatency, m_totalSerializationLatency, m_uop_type_count, m_uops_pause, m_uops_total, m_uops_x87, HitWhere::NUM_HITWHERES, registerStatsMetric(), STOP_DISPATCH_SIZE, StopDispatchReasonString(), MicroOp::UOP_SUBTYPE_SIZE, HitWhere::WHERE_FIRST, and SubsecondTime::Zero().

6.163.2.2 ~IntervalTimer()

```
IntervalTimer::~IntervalTimer ( )
```

Definition at line 159 of file interval_timer.cc.

References free().

6.163.3 Member Function Documentation

6.163.3.1 blockWindow()

```
void IntervalTimer::blockWindow ( )  [protected]
```

An microOperation blocks the window: mark the micro-ops that overlap with this micro-op. Start the loads that overlap with this micro-op.

Definition at line 583 of file interval_timer.cc.

References Windows::WindowEntry::addOverlapFlag(), Windows::WindowEntry::BPRED_OVERLAP, Windows::WindowEntry::clearDependent(), Windows::WindowEntry::DCACHE_OVERLAP, Windows::getCriticalPathHead(), DynamicMicroOp::getDependency(), Windows::WindowEntry::getDynMicroOp(), DynamicMicroOp::getExecLatency(), Windows::getInstruction(), DynamicMicroOp::getLoadAccess(), Windows::WindowEntry::getMicroOp(), Windows::getWindowIterator(), Windows::Iterator::hasNext(), Windows::WindowEntry::hasOverlapFlag(), Windows::WindowEntry::ICACHE_OVERLAP, MicroOp::isBranch(), Windows::WindowEntry::isDependent(),

Windows::WindowEntry::isIndependent(), MicroOp::isLoad(), DynamicMicroOp::isLongLatencyLoad(), MicroOp::isMemBarrier(), MicroOp::isSerializing(), issueMemOp(), m_lll_dep_mask, m_numBPredOverlapped, m_numDCacheOverlapped, m_numHiddenLongerDCacheLatency, m_numICacheOverlapped, m_totalHiddenDCacheLatency, m_totalHiddenLongerDCacheLatency, m_windows, Windows::Iterator::next(), Memory::Access::phys, Windows::WindowEntry::setDataDependent(), Windows::WindowEntry::setExecTime(), Windows::WindowEntry::setIndependentMiss(), and Windows::windowContains().

Referenced by dispatchInstruction().

6.163.3.2 calculateCurrentDispatchRate()

```
uint32_t IntervalTimer::calculateCurrentDispatchRate ( )  [protected]
```

Definition at line 310 of file interval_timer.cc.

References TFixedPoint< __UINT64_C(0x4000)>::floor(), Windows::getCriticalPathLength(), Windows::getEffectiveCriticalPathLength(), Windows::getOldWindowLength(), m_dispatch_width, m_remaining_dispatch_bandwidth, and m_windows.

Referenced by dispatchWindow().

6.163.3.3 dispatchInstruction()

```
uint64_t IntervalTimer::dispatchInstruction (
              Windows::WindowEntry & instruction,
              StopDispatchReason & continueDispatching )  [protected]
```

Definition at line 356 of file interval_timer.cc.

References blockWindow(), Windows::WindowEntry::BPRED_OVERLAP, Windows::calculateBranchResolutionLatency(), Windows::clearOldWindow(), Windows::WindowEntry::cphead, Windows::WindowEntry::cptail, Windows::WindowEntry::DCACHE_OVERLAP, ContentionModel::getBarrierCompletionTime(), CoreModel::getBypassLatency(), ContentionModel::getCompletionTime(), Windows::getCriticalPathHead(), Windows::getCriticalPathLength(), Windows::getCriticalPathTail(), DynamicMicroOp::getDCacheHitWhere(), Windows::WindowEntry::getDynMicroOp(), DynamicMicroOp::getExecLatency(), Windows::WindowEntry::getExecTime(), DynamicMicroOp::getICacheHitWhere(), DynamicMicroOp::getICacheLatency(), PerformanceModel::getInstructionCount(), getMaxProducerExecTime(), Windows::WindowEntry::getMicroOp(), Windows::getMinimalFlushLatency(), DynamicMicroOp::getPeriod(), Windows::WindowEntry::hasOverlapFlag(), Windows::WindowEntry::ICACHE_OVERLAP, MicroOp::isBranch(), DynamicMicroOp::isBranchMispredicted(), MicroOp::isExecute(), MicroOp::isInterrupt(), MicroOp::isLoad(), DynamicMicroOp::isLongLatencyLoad(), MicroOp::isMemBarrier(), MicroOp::isSerializing(), MicroOp::isStore(), issueMemOp(), HitWhere::L1I, m_branch_misprediction_penalty, m_core_model, m_cpiBranchPredictor, m_cpiDataCache, m_cpiInstructionCache, m_cpiSerialization, m_dispatch_width, m_lastAccountedMemoryCycle, m_loadstore_contention, m_max_load_completion_time, m_max_store_completion_time, m_numLongLatencyLoads, m_numMfenceInsns, m_numSerializationInsns, m_numTotalLongLatencyLoadLatency, m_outstandingLongLatencyCycles, m_outstandingLongLatencyInsns, m_perf_model, m_totalSerializationLatency, m_windows, Windows::WindowEntry::maxProducer, Windows::WindowEntry::setExecTime(), STOP_DISPATCH_BRANCH_MISPREDICT, STOP_DISPATCH_ICACHE_MISS, and updateCriticalPath().

Referenced by dispatchWindow().

6.163.3.4 dispatchWindow()

```
boost::tuple< uint64_t, uint64_t > IntervalTimer::dispatchWindow ( )  [protected]
```

Definition at line 225 of file interval_timer.cc.

References ADD_STOP_DISPATCH_REASON, calculateCurrentDispatchRate(), CPCONTR_TYPE_SIZE, CpContrTypeString(), dispatchInstruction(), Windows::dispatchInstruction(), Windows::getCpContrFraction(), Windows::getCriticalPathLength(), Windows::WindowEntry::getDynMicroOp(), Windows::getEffectiveCriticalPathLength(), PerformanceModel::getInstructionCount(), Windows::getInstructionToDispatch(), Windows::WindowEntry::getMicroOp(), Core::getPerformanceModel(), DynamicMicroOp::getPeriod(), ComponentPeriod::getPeriod(), DynamicMicroOp::isLast(), m_core, Windows::m_cpcontr_bytype, Windows::m_cpcontr_total, m_cpContrByType, m_cpiBase, m_cpiBaseStopDispatch, m_dispatch_width, m_frequency_domain, m_perf_model, m_windows, STOP_DISPATCH_DISPATCH_RATE, STOP_DISPATCH_DISPATCH_WIDTH, STOP_DISPATCH_NO_REASON, STOP_DISPATCH_WINDOW_EMPTY, PerformanceModel::traceInstruction(), and Windows::wIsEmpty().

Referenced by simulate().

6.163.3.5 free()

```
void IntervalTimer::free ( )
```

Definition at line 164 of file interval_timer.cc.

References m_windows.

Referenced by IntervalPerformanceModel::~IntervalPerformanceModel(), and ~IntervalTimer().

6.163.3.6 getMaxProducerExecTime()

```
uint64_t IntervalTimer::getMaxProducerExecTime (
              Windows::WindowEntry & instruction )  [protected]
```

Definition at line 689 of file interval_timer.cc.

References DynamicMicroOp::getDependency(), Windows::WindowEntry::getDynMicroOp(), DynamicMicroOp::getExecLatency(), Windows::WindowEntry::getExecTime(), Windows::WindowEntry::getFetchTime(), Windows::getInstruction(), Windows::getOldestInstruction(), m_windows, and Windows::oldWindowContains().

Referenced by dispatchInstruction().

6.163.3.7 issueMemOp()

```
void IntervalTimer::issueMemOp (
            Windows::WindowEntry & micro_op )  [protected]
```

Definition at line 331 of file interval_timer.cc.

References Core::accessMemory(), Memory::Access::address, SubsecondTime::divideRounded(), Instruction↩ ::getAddress(), DynamicMicroOp::getAddress(), DynamicMicroOp::getDCacheHitWhere(), Core::getDvfs↩ Domain(), Windows::WindowEntry::getDynMicroOp(), DynamicMicroOp::getExecLatency(), MicroOp::get↩ Instruction(), MicroOp::getMemoryAccessSize(), Windows::WindowEntry::getMicroOp(), ComponentPeriod↩ ::getPeriod(), MemoryResult::hit_where, MicroOp::isLoad(), MicroOp::isStore(), MemoryResult::latency, m_↩ core, Core::MEM_MODELED_RETURN, Core::NONE, Core::READ, DynamicMicroOp::setDCacheHitWhere(), DynamicMicroOp::setExecLatency(), HitWhere::UNKNOWN, and Core::WRITE.

Referenced by blockWindow(), and dispatchInstruction().

6.163.3.8 simulate()

```
boost::tuple< uint64_t, uint64_t > IntervalTimer::simulate (
            const std::vector< DynamicMicroOp * > & insts )
```

Definition at line 180 of file interval_timer.cc.

References Windows::add(), Memory::Access::address, dispatchWindow(), LOG_PRINT_WARNING_ONCE, m↩ _mem_dep_mask, m_uop_type_count, m_uops_pause, m_uops_total, m_uops_x87, m_windows, and Windows↩ ::wIsFull().

Referenced by IntervalPerformanceModel::simulate().

6.163.3.9 synchronize()

```
void IntervalTimer::synchronize (
            uint64_t time )  [inline]
```

Definition at line 49 of file interval_timer.h.

Referenced by IntervalPerformanceModel::notifyElapsedTimeUpdate().

6.163.3.10 updateCriticalPath()

```
void IntervalTimer::updateCriticalPath (
             Windows::WindowEntry & micro_op,
            uint64_t & latency )  [protected]
```

Move a micro_op to the old window. If it extends the critical path by more than the long-latency cut-off, clear the window instead.

Definition at line 564 of file interval_timer.cc.

References Windows::clearOldWindow(), Windows::WindowEntry::getDynMicroOp(), Windows::WindowEntry↩ ::getExecTime(), DynamicMicroOp::getPeriod(), Windows::longLatencyOperationLatency(), m_cpiLongLatency, m_windows, and Windows::updateCriticalPathTail().

Referenced by dispatchInstruction().

6.163.4 Member Data Documentation

6.163.4.1 m_branch_misprediction_penalty

```
const uint32_t IntervalTimer::m_branch_misprediction_penalty  [private]
```

Definition at line 70 of file interval_timer.h.

Referenced by dispatchInstruction().

6.163.4.2 m_core

```
Core* IntervalTimer::m_core  [private]
```

Definition at line 65 of file interval_timer.h.

Referenced by dispatchWindow(), and issueMemOp().

6.163.4.3 m_core_model

```
const  CoreModel* IntervalTimer::m_core_model  [private]
```

Definition at line 66 of file interval_timer.h.

Referenced by dispatchInstruction().

6.163.4.4 m_cpContrByType

```
uint64_t IntervalTimer::m_cpContrByType[ CPCONTR_TYPE_SIZE]  [private]
```

Definition at line 131 of file interval_timer.h.

Referenced by dispatchWindow(), and IntervalTimer().

6.163.4.5 m_cpiBase

```
SubsecondTime IntervalTimer::m_cpiBase  [private]
```

Definition at line 122 of file interval_timer.h.

Referenced by dispatchWindow(), and IntervalTimer().

6.163.4.6 m_cpiBaseStopDispatch

```
uint64_t IntervalTimer::m_cpiBaseStopDispatch[ STOP_DISPATCH_SIZE]  [private]
```

Definition at line 130 of file interval_timer.h.

Referenced by dispatchWindow(), and IntervalTimer().

6.163.4.7 m_cpiBranchPredictor

```
SubsecondTime IntervalTimer::m_cpiBranchPredictor  [private]
```

Definition at line 123 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.8 m_cpiDataCache

```
std::vector< SubsecondTime> IntervalTimer::m_cpiDataCache  [private]
```

Definition at line 128 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.9 m_cpiInstructionCache

```
std::vector< SubsecondTime> IntervalTimer::m_cpiInstructionCache  [private]
```

Definition at line 127 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.10 m_cpiLongLatency

```
SubsecondTime IntervalTimer::m_cpiLongLatency  [private]
```

Definition at line 125 of file interval_timer.h.

Referenced by IntervalTimer(), and updateCriticalPath().

6.163.4.11 m_cpiSerialization

SubsecondTime IntervalTimer::m_cpiSerialization [private]

Definition at line 124 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.12 m_dispatch_width

const uint32_t IntervalTimer::m_dispatch_width [private]

Definition at line 69 of file interval_timer.h.

Referenced by calculateCurrentDispatchRate(), dispatchInstruction(), and dispatchWindow().

6.163.4.13 m_frequency_domain

const **ComponentPeriod*** IntervalTimer::m_frequency_domain [private]

Definition at line 88 of file interval_timer.h.

Referenced by dispatchWindow().

6.163.4.14 m_lastAccountedMemoryCycle

uint64_t IntervalTimer::m_lastAccountedMemoryCycle [private]

Definition at line 116 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.15 m_lll_dep_mask

UInt64 IntervalTimer::m_lll_dep_mask [private]

Definition at line 72 of file interval_timer.h.

Referenced by blockWindow(), and IntervalTimer().

6.163.4.16 m_loadstore_contention

ContentionModel IntervalTimer::m_loadstore_contention [private]

Definition at line 83 of file interval_timer.h.

Referenced by dispatchInstruction().

6.163.4.17 m_max_load_completion_time

uint64_t IntervalTimer::m_max_load_completion_time [private]

Definition at line 81 of file interval_timer.h.

Referenced by dispatchInstruction().

6.163.4.18 m_max_store_completion_time

uint64_t IntervalTimer::m_max_store_completion_time [private]

Definition at line 80 of file interval_timer.h.

Referenced by dispatchInstruction().

6.163.4.19 m_mem_dep_mask

UInt64 IntervalTimer::m_mem_dep_mask [private]

Definition at line 71 of file interval_timer.h.

Referenced by IntervalTimer(), and simulate().

6.163.4.20 m_numBPredOverlapped

uint64_t IntervalTimer::m_numBPredOverlapped [private]

Definition at line 101 of file interval_timer.h.

Referenced by blockWindow(), and IntervalTimer().

6.163.4.21 m_numDCacheOverlapped

```
uint64_t IntervalTimer::m_numDCacheOverlapped  [private]
```

Definition at line 102 of file interval_timer.h.

Referenced by blockWindow(), and IntervalTimer().

6.163.4.22 m_numHiddenLongerDCacheLatency

```
uint64_t IntervalTimer::m_numHiddenLongerDCacheLatency  [private]
```

Definition at line 112 of file interval_timer.h.

Referenced by blockWindow(), and IntervalTimer().

6.163.4.23 m_numICacheOverlapped

```
uint64_t IntervalTimer::m_numICacheOverlapped  [private]
```

Definition at line 100 of file interval_timer.h.

Referenced by blockWindow(), and IntervalTimer().

6.163.4.24 m_numLongLatencyLoads

```
uint64_t IntervalTimer::m_numLongLatencyLoads  [private]
```

Definition at line 104 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.25 m_numMfenceInsns

```
uint64_t IntervalTimer::m_numMfenceInsns  [private]
```

Definition at line 118 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.26 m_numSerializationInsns

```
uint64_t IntervalTimer::m_numSerializationInsns  [private]
```

Definition at line 107 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.27 m_numTotalLongLatencyLoadLatency

```
uint64_t IntervalTimer::m_numTotalLongLatencyLoadLatency  [private]
```

Definition at line 105 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.28 m_outstandingLongLatencyCycles

```
uint64_t IntervalTimer::m_outstandingLongLatencyCycles  [private]
```

Definition at line 115 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.29 m_outstandingLongLatencyInsns

```
uint64_t IntervalTimer::m_outstandingLongLatencyInsns  [private]
```

Definition at line 114 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.30 m_perf_model

```
PerformanceModel* IntervalTimer::m_perf_model  [private]
```

Definition at line 87 of file interval_timer.h.

Referenced by dispatchInstruction(), and dispatchWindow().

6.163.4.31 m_remaining_dispatch_bandwidth

```
FixedPoint IntervalTimer::m_remaining_dispatch_bandwidth  [private]
```

Definition at line 75 of file interval_timer.h.

Referenced by calculateCurrentDispatchRate().

6.163.4.32 m_totalHiddenDCacheLatency

```
uint64_t IntervalTimer::m_totalHiddenDCacheLatency  [private]
```

Definition at line 110 of file interval_timer.h.

Referenced by blockWindow(), and IntervalTimer().

6.163.4.33 m_totalHiddenLongerDCacheLatency

```
uint64_t IntervalTimer::m_totalHiddenLongerDCacheLatency  [private]
```

Definition at line 111 of file interval_timer.h.

Referenced by blockWindow(), and IntervalTimer().

6.163.4.34 m_totalMfenceLatency

```
uint64_t IntervalTimer::m_totalMfenceLatency  [private]
```

Definition at line 119 of file interval_timer.h.

Referenced by IntervalTimer().

6.163.4.35 m_totalSerializationLatency

```
uint64_t IntervalTimer::m_totalSerializationLatency  [private]
```

Definition at line 108 of file interval_timer.h.

Referenced by dispatchInstruction(), and IntervalTimer().

6.163.4.36 m_uop_type_count

```
UInt64 IntervalTimer::m_uop_type_count[ MicroOp::UOP_SUBTYPE_SIZE] [private]
```

Definition at line 95 of file interval_timer.h.

Referenced by IntervalTimer(), and simulate().

6.163.4.37 m_uops_pause

```
UInt64 IntervalTimer::m_uops_pause [private]
```

Definition at line 98 of file interval_timer.h.

Referenced by IntervalTimer(), and simulate().

6.163.4.38 m_uops_total

```
UInt64 IntervalTimer::m_uops_total [private]
```

Definition at line 96 of file interval_timer.h.

Referenced by IntervalTimer(), and simulate().

6.163.4.39 m_uops_x87

```
UInt64 IntervalTimer::m_uops_x87 [private]
```

Definition at line 97 of file interval_timer.h.

Referenced by IntervalTimer(), and simulate().

6.163.4.40 m_windows

```
Windows* IntervalTimer::m_windows [private]
```

Definition at line 86 of file interval_timer.h.

Referenced by blockWindow(), calculateCurrentDispatchRate(), dispatchInstruction(), dispatchWindow(), free(), getMaxProducerExecTime(), simulate(), and updateCriticalPath().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/interval_performance_model/ **interval_timer.h**
- common/performance_model/performance_models/interval_performance_model/ **interval_timer.cc**

6.164 Windows::Iterator Class Reference

`#include <windows.h>`

Public Member Functions

- **Iterator** (const **Windows** *const **windows**, int start, int **stop**)
- bool **hasNext** ()
- **WindowEntry** & **next** ()

Private Attributes

- int **index**
- const int **stop**
- const **Windows** *const **windows**

6.164.1 Detailed Description

Definition at line 145 of file windows.h.

6.164.2 Constructor & Destructor Documentation

6.164.2.1 Iterator()

```
Windows::Iterator::Iterator (
            const  Windows *const windows,
            int start,
            int stop )
```

Definition at line 327 of file windows.cc.

6.164.3 Member Function Documentation

6.164.3.1 hasNext()

```
bool Windows::Iterator::hasNext ( )
```

Definition at line 329 of file windows.cc.

Referenced by IntervalTimer::blockWindow().

6.164.3.2 next()

```
Windows::WindowEntry & Windows::Iterator::next ( )
```

Definition at line 334 of file windows.cc.

Referenced by IntervalTimer::blockWindow().

6.164.4 Member Data Documentation

6.164.4.1 index

```
int Windows::Iterator::index  [private]
```

Definition at line 147 of file windows.h.

6.164.4.2 stop

```
const int Windows::Iterator::stop  [private]
```

Definition at line 148 of file windows.h.

6.164.4.3 windows

```
const  Windows* const Windows::Iterator::windows  [private]
```

Definition at line 149 of file windows.h.

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/interval_performance_model/ **windows.h**
- common/performance_model/performance_models/interval_performance_model/ **windows.cc**

6.165 MTCircularQueue< T >::iterator Class Reference

```
#include <mt_circular_queue.h>
```

Public Member Functions

- **iterator** (const **MTCircularQueue** &queue, **UInt32** idx)=0

6.165.1 Detailed Description

template<class T>
class MTCircularQueue< T >::iterator

Definition at line 15 of file mt_circular_queue.h.

6.165.2 Constructor & Destructor Documentation

6.165.2.1 iterator()

```
template<class T >
MTCircularQueue< T >::iterator::iterator (
            const  MTCircularQueue & queue,
             UInt32 idx )  [pure virtual]
```

The documentation for this class was generated from the following file:

- common/misc/ **mt_circular_queue.h**

6.166 CircularQueue< T >::iterator Class Reference

`#include <circular_queue.h>`

Inheritance diagram for CircularQueue< T >::iterator:

Public Member Functions

- **iterator** (**CircularQueue** &queue, **UInt32** idx)
- T & **operator*** () const
- T * **operator->** () const
- **iterator** & **operator++** ()
- bool **operator==** (**iterator** const &rhs) const
- bool **operator!=** (**iterator** const &rhs) const

Private Attributes

- **CircularQueue** & **_queue**
- **UInt32** **_idx**

6.166.1 Detailed Description

template<class T>
class CircularQueue< T >::iterator

Definition at line 19 of file circular_queue.h.

6.166.2 Constructor & Destructor Documentation

6.166.2.1 iterator()

```
template<class T >
CircularQueue< T >::iterator::iterator (
              CircularQueue & queue,
              UInt32 idx )  [inline]
```

Definition at line 25 of file circular_queue.h.

6.166.3 Member Function Documentation

6.166.3.1 operator"!=()

```
template<class T >
bool  CircularQueue< T >::iterator::operator!= (
              iterator const & rhs ) const  [inline]
```

Definition at line 30 of file circular_queue.h.

6.166.3.2 operator*()

```
template<class T >
T&  CircularQueue< T >::iterator::operator* ( ) const  [inline]
```

Definition at line 26 of file circular_queue.h.

References CircularQueue< T >::iterator::_idx, CircularQueue< T >::iterator::_queue, and CircularQueue< T >::at().

6.166.3.3 operator++()

```
template<class T >
iterator&  CircularQueue< T >::iterator::operator++ ( )  [inline]
```

Definition at line 28 of file circular_queue.h.

References CircularQueue< T >::iterator::_idx.

6.166.3.4 operator->()

```
template<class T >
T*  CircularQueue< T >::iterator::operator-> ( ) const  [inline]
```

Definition at line 27 of file circular_queue.h.

References CircularQueue< T >::iterator::_idx, CircularQueue< T >::iterator::_queue, and CircularQueue< T >::at().

6.166.3.5 operator==()

```
template<class T >
bool  CircularQueue< T >::iterator::operator== (
              iterator const & rhs ) const  [inline]
```

Definition at line 29 of file circular_queue.h.

References CircularQueue< T >::iterator::_idx, and CircularQueue< T >::iterator::_queue.

6.166.4 Member Data Documentation

6.166.4.1 _idx

```
template<class T >
UInt32  CircularQueue< T >::iterator::_idx  [private]
```

Definition at line 23 of file circular_queue.h.

Referenced by CircularQueue< T >::iterator::operator*(), CircularQueue< T >::iterator::operator++(), CircularQueue< T >::iterator::operator->(), and CircularQueue< T >::iterator::operator==().

6.166.4.2 _queue

```
template<class T >
CircularQueue& CircularQueue< T >::iterator::_queue [private]
```

Definition at line 22 of file circular_queue.h.

Referenced by CircularQueue< T >::iterator::operator*(), CircularQueue< T >::iterator::operator->(), and CircularQueue< T >::iterator::operator==().

The documentation for this class was generated from the following file:

- common/misc/ **circular_queue.h**

6.167 JmpInstruction Class Reference

```
#include <instruction.h>
```

Inheritance diagram for JmpInstruction:

Public Member Functions

- **JmpInstruction** (**OperandList** &dest)

Additional Inherited Members

6.167.1 Detailed Description

Definition at line 113 of file instruction.h.

6.167.2 Constructor & Destructor Documentation

6.167.2.1 JmpInstruction()

```
JmpInstruction::JmpInstruction (
              OperandList & dest ) [inline]
```

Definition at line 116 of file instruction.h.

The documentation for this class was generated from the following file:

- common/performance_model/ **instruction.h**

6.168 config::Key Class Reference

Key (p. 669): A configuration setting entry This class is used to hold a given setting from a configuration. It contains the actual data, as well as functions to get the type.

`#include <key.hpp>`

Public Member Functions

- template<class V >
  Key (const String &parentPath, const String &name, const V &value)
  Constructor to create a key.
- const String **getString** () const
- bool **getBool** () const
- **SInt64** **getInt** () const
- double **getFloat** () const
- void **getValue** (bool &bool_val) const
- void **getValue** (**SInt64** &int_val) const
- void **getValue** (String &string_val) const
- void **getValue** (double &double_val) const
- bool **getFloatValid** ()
- bool **getIntValid** ()
- bool **getBoolValid** ()
- bool **getStringValid** ()
- const String **getName** () const
- template<> **Key** (const String &parentPath_, const String &name_, const String &value_)

Private Member Functions

- unsigned short **DetermineType** (String)
- void **throwInvalid** (String) const

Private Attributes

- String **m_name**
- String **m_value**
- double **m_value_f**
- **SInt64** **m_value_i**
- bool **m_value_b**
- String **m_parentPath**
- const unsigned short **m_type**

6.168.1 Detailed Description

Key (p. 669): A configuration setting entry This class is used to hold a given setting from a configuration. It contains the actual data, as well as functions to get the type.

Definition at line 28 of file key.hpp.

6.168.2 Constructor & Destructor Documentation

6.168.2.1 Key() [1/2]

```
template<class V >
config::Key::Key (
            const String & parentPath,
            const String & name,
            const V & value )
```

Constructor to create a key.

Definition at line 28 of file key.cpp.

6.168.2.2 Key() [2/2]

```
template<>
config::Key::Key (
            const String & parentPath_,
            const String & name_,
            const String & value_ )
```

Definition at line 37 of file key.cpp.

6.168.3 Member Function Documentation

6.168.3.1 DetermineType()

```
unsigned short config::Key::DetermineType (
            String value )  [private]
```

Definition at line 50 of file key.cpp.

References config::long_p, m_value, m_value_b, m_value_f, m_value_i, config::TYPE_BOOL_VALID, config::TY↩PE_FLOAT_VALID, config::TYPE_INT_VALID, and config::TYPE_STRING_VALID.

6.168.3.2 getBool()

```
bool config::Key::getBool ( ) const
```

Referenced by config::Config::getBoolArray().

6.168.3.3 getBoolValid()

```
bool config::Key::getBoolValid ( )  [inline]
```

Definition at line 48 of file key.hpp.

References m_type, and config::TYPE_BOOL_VALID.

6.168.3.4 getFloat()

```
double config::Key::getFloat ( ) const
```

Referenced by config::Config::getFloatArray().

6.168.3.5 getFloatValid()

```
bool config::Key::getFloatValid ( )  [inline]
```

Definition at line 46 of file key.hpp.

References m_type, and config::TYPE_FLOAT_VALID.

6.168.3.6 getInt()

```
SInt64 config::Key::getInt ( ) const
```

Referenced by config::Config::getIntArray().

6.168.3.7 getIntValid()

```
bool config::Key::getIntValid ( )  [inline]
```

Definition at line 47 of file key.hpp.

References m_type, and config::TYPE_INT_VALID.

6.168.3.8 getName()

```
const String config::Key::getName ( ) const  [inline]
```

Definition at line 51 of file key.hpp.

References m_name.

6.168.3.9 getString()

```
const String config::Key::getString ( ) const
```

Referenced by config::Config::getStringArray().

6.168.3.10 getStringValid()

```
bool config::Key::getStringValid ( )  [inline]
```

Definition at line 49 of file key.hpp.

6.168.3.11 getValue() [1/4]

```
void config::Key::getValue (
             bool & bool_val ) const
```

6.168.3.12 getValue() [2/4]

```
void config::Key::getValue (
             double & double_val ) const
```

6.168.3.13 getValue() [3/4]

```
void config::Key::getValue (
              SInt64 & int_val ) const
```

6.168.3.14 getValue() [4/4]

```
void config::Key::getValue (
            String & string_val ) const
```

6.168.3.15 throwInvalid()

```
void config::Key::throwInvalid (
            String  ) const  [private]
```

6.168.4 Member Data Documentation

6.168.4.1 m_name

```
String config::Key::m_name  [private]
```

Definition at line 56 of file key.hpp.

Referenced by getName().

6.168.4.2 m_parentPath

```
String config::Key::m_parentPath  [private]
```

Definition at line 63 of file key.hpp.

6.168.4.3 m_type

```
const unsigned short config::Key::m_type  [private]
```

Definition at line 64 of file key.hpp.

Referenced by getBoolValid(), getFloatValid(), and getIntValid().

6.168.4.4 m_value

```
String config::Key::m_value  [private]
```

Definition at line 58 of file key.hpp.

Referenced by DetermineType().

6.168.4.5 m_value_b

```
bool config::Key::m_value_b  [private]
```

Definition at line 61 of file key.hpp.

Referenced by DetermineType().

6.168.4.6 m_value_f

```
double config::Key::m_value_f  [private]
```

Definition at line 59 of file key.hpp.

Referenced by DetermineType().

6.168.4.7 m_value_i

```
SInt64 config::Key::m_value_i  [private]
```

Definition at line 60 of file key.hpp.

Referenced by DetermineType().

The documentation for this class was generated from the following files:

- common/config/ **key.hpp**
- common/config/ **key.cpp**

6.169 config::KeyNotFound Class Reference

```
#include <config_exceptions.hpp>
```

Inheritance diagram for config::KeyNotFound:

Private Member Functions

- virtual const char * **what** () const throw ()

6.169.1 Detailed Description

Definition at line 36 of file config_exceptions.hpp.

6.169.2 Member Function Documentation

6.169.2.1 what()

```
virtual const char* config::KeyNotFound::what ( ) const throw ( )   [inline], [private], [virtual]
```

Definition at line 38 of file config_exceptions.hpp.

The documentation for this class was generated from the following file:

- common/config/ **config_exceptions.hpp**

6.170 LockCreator Class Reference

```
#include <lock.h>
```

Inheritance diagram for LockCreator:

Static Public Member Functions

- static **LockImplementation** * **create** ()

6.170.1 Detailed Description

Definition at line 27 of file lock.h.

6.170.2 Member Function Documentation

6.170.2.1 create()

```
static LockImplementation* LockCreator::create ( ) [static]
```

The documentation for this class was generated from the following file:

- common/misc/ **lock.h**

6.171 LockCreator_Default Class Reference

```
#include <lock.h>
```

Inheritance diagram for LockCreator_Default:

Static Public Member Functions

- static **LockImplementation** * **create** ()

6.171.1 Detailed Description

Definition at line 98 of file lock.h.

6.171.2 Member Function Documentation

6.171.2.1 create()

```
LockImplementation * LockCreator_Default::create ( ) [static]
```

Definition at line 23 of file pin_lock.cc.

The documentation for this class was generated from the following files:

- common/misc/ **lock.h**
- pin/ **pin_lock.cc**

6.172 LockCreator_NullLock Class Reference

`#include <lock.h>`

Inheritance diagram for LockCreator_NullLock:

Static Public Member Functions

- static **LockImplementation** * **create** ()

6.172.1 Detailed Description

Definition at line 113 of file lock.h.

6.172.2 Member Function Documentation

6.172.2.1 create()

`static` **LockImplementation*** `LockCreator_NullLock::create ( )` `[static]`

The documentation for this class was generated from the following file:

- common/misc/ **lock.h**

6.173 LockCreator_RwLock Class Reference

`#include <lock.h>`

Inheritance diagram for LockCreator_RwLock:

Static Public Member Functions

- static **LockImplementation** * **create** ()

6.173.1 Detailed Description

Definition at line 103 of file lock.h.

6.173.2 Member Function Documentation

6.173.2.1 create()

```
LockImplementation * LockCreator_RwLock::create ( )  [static]
```

Definition at line 27 of file pin_lock.cc.

The documentation for this class was generated from the following files:

- common/misc/ **lock.h**
- pin/ **pin_lock.cc**

6.174 LockCreator_Spinlock Class Reference

```
#include <lock.h>
```

Inheritance diagram for LockCreator_Spinlock:

Static Public Member Functions

- static **LockImplementation** * **create** ()

6.174.1 Detailed Description

Definition at line 108 of file lock.h.

6.174.2 Member Function Documentation

6.174.2.1 create()

```
LockImplementation * LockCreator_Spinlock::create ( )  [static]
```

Definition at line 33 of file pin_lock.cc.

The documentation for this class was generated from the following files:

- common/misc/ **lock.h**
- pin/ **pin_lock.cc**

6.175 LockedHash Class Reference

`#include <locked_hash.h>`

Public Member Functions

- **LockedHash** (**UInt64** size)
- ~**LockedHash** ()
- std::pair< bool, **UInt64** > **find** (**UInt64** key)
- bool **insert** (**UInt64** key, **UInt64** value)
- void **remove** (**UInt64** key)

Protected Types

- typedef std::unordered_map< **UInt64**, **UInt64** > **Bucket**

Protected Attributes

- **UInt64** **_size**
- **Bucket** * **_bins**
- **Lock** * **_locks**

6.175.1 Detailed Description

Definition at line 9 of file locked_hash.h.

6.175.2 Member Typedef Documentation

6.175.2.1 Bucket

```
typedef std::unordered_map< UInt64, UInt64> LockedHash::Bucket [protected]
```

Definition at line 12 of file locked_hash.h.

6.175.3 Constructor & Destructor Documentation

6.175.3.1 LockedHash()

```
LockedHash::LockedHash (
            UInt64 size )
```

Definition at line 3 of file locked_hash.cc.

6.175.3.2 ~LockedHash()

```
LockedHash::~LockedHash ( )
```

Definition at line 11 of file locked_hash.cc.

6.175.4 Member Function Documentation

6.175.4.1 find()

```
std::pair< bool, UInt64 > LockedHash::find (
            UInt64 key )
```

Definition at line 20 of file locked_hash.cc.

References _bins, _locks, _size, TLock< T_LockCreator >::acquire(), and TLock< T_LockCreator >::release().

6.175.4.2 insert()

```
bool LockedHash::insert (
            UInt64 key,
            UInt64 value )
```

Definition at line 54 of file locked_hash.cc.

References _bins, _locks, _size, TLock< T_LockCreator >::acquire(), and TLock< T_LockCreator >::release().

6.175.4.3 remove()

```
void LockedHash::remove (
              UInt64 key )
```

Definition at line 40 of file locked_hash.cc.

References _bins, _locks, _size, TLock< T_LockCreator >::acquire(), and TLock< T_LockCreator >::release().

6.175.5 Member Data Documentation

6.175.5.1 _bins

```
Bucket* LockedHash::_bins  [protected]
```

Definition at line 15 of file locked_hash.h.

Referenced by find(), insert(), and remove().

6.175.5.2 _locks

```
Lock* LockedHash::_locks  [protected]
```

Definition at line 16 of file locked_hash.h.

Referenced by find(), insert(), and remove().

6.175.5.3 _size

```
UInt64 LockedHash::_size  [protected]
```

Definition at line 14 of file locked_hash.h.

Referenced by find(), insert(), and remove().

The documentation for this class was generated from the following files:

- common/misc/ **locked_hash.h**
- common/misc/ **locked_hash.cc**

6.176 LockFreeHash Class Reference

`#include <lockfree_hash.h>`

Inheritance diagram for LockFreeHash:

Public Member Functions

- **LockFreeHash** (**UInt64** **size**)
- ~**LockFreeHash** ()
- std::pair< bool, **UInt64** > **find** (**UInt64** key)
- bool **insert** (**UInt64** key, **UInt64** value)

Private Member Functions

- **UInt64** **bucket_size** (**UInt64** key)

Additional Inherited Members

6.176.1 Detailed Description

Definition at line 15 of file lockfree_hash.h.

6.176.2 Constructor & Destructor Documentation

6.176.2.1 LockFreeHash()

```
LockFreeHash::LockFreeHash (
              UInt64 size )
```

Definition at line 3 of file lockfree_hash.cc.

6.176.2.2 ~LockFreeHash()

```
LockFreeHash::~LockFreeHash ( )
```

Definition at line 7 of file lockfree_hash.cc.

6.176.3 Member Function Documentation

6.176.3.1 bucket_size()

```
UInt64 LockFreeHash::bucket_size (
            UInt64 key )  [private]
```

Definition at line 12 of file lockfree_hash.cc.

References BasicHash::array, and BasicHash::size.

Referenced by find(), and insert().

6.176.3.2 find()

```
std::pair< bool,  UInt64 > LockFreeHash::find (
            UInt64 key )
```

Definition at line 19 of file lockfree_hash.cc.

References bucket_size(), and BasicHash::find().

6.176.3.3 insert()

```
bool LockFreeHash::insert (
            UInt64 key,
            UInt64 value )
```

Definition at line 27 of file lockfree_hash.cc.

References bucket_size(), and BasicHash::insert().

The documentation for this class was generated from the following files:

- common/misc/ **lockfree_hash.h**
- common/misc/ **lockfree_hash.cc**

6.177 LockImplementation Class Reference

```
#include <lock.h>
```

Inheritance diagram for LockImplementation:

Public Member Functions

- **LockImplementation** ()
- virtual ~**LockImplementation** ()
- virtual void **acquire** ()=0
- virtual void **release** ()=0
- virtual void **acquire_read** ()
- virtual void **release_read** ()

6.177.1 Detailed Description

Definition at line 15 of file lock.h.

6.177.2 Constructor & Destructor Documentation

6.177.2.1 LockImplementation()

```
LockImplementation::LockImplementation ( )  [inline]
```

Definition at line 18 of file lock.h.

6.177.2.2 ~LockImplementation()

```
virtual LockImplementation::~LockImplementation ( )  [inline], [virtual]
```

Definition at line 19 of file lock.h.

6.177.3 Member Function Documentation

6.177.3.1 acquire()

```
virtual void LockImplementation::acquire ( )  [pure virtual]
```

Implemented in **PthreadLock** (p. 952), and **PinLock** (p. 930).

Referenced by TLock< LockCreator_Default >::acquire(), and acquire_read().

6.177.3.2 acquire_read()

```
virtual void LockImplementation::acquire_read ( )  [inline], [virtual]
```

Definition at line 23 of file lock.h.

References acquire().

Referenced by TLock< LockCreator_Default >::acquire_read().

6.177.3.3 release()

```
virtual void LockImplementation::release ( )  [pure virtual]
```

Implemented in **PthreadLock** (p. 952), and **PinLock** (p. 930).

Referenced by TLock< LockCreator_Default >::release(), and release_read().

6.177.3.4 release_read()

```
virtual void LockImplementation::release_read ( )  [inline], [virtual]
```

Definition at line 24 of file lock.h.

References release().

Referenced by TLock< LockCreator_Default >::release_read().

The documentation for this class was generated from the following file:

- common/misc/ **lock.h**

6.178 Log Class Reference

```
#include <log.h>
```

Public Types

- enum **ErrorState** { **None**, **Warning**, **Error** }

Public Member Functions

- **Log** (**Config** &config)
- ∼**Log** ()
- void **log** (**ErrorState** err, const char ∗source_file, **SInt32** source_line, const char ∗format,...)
- bool **isEnabled** (const char ∗module)
- bool **isLoggingEnabled** () const
- String **getModule** (const char ∗filename)

Static Public Member Functions

- static **Log** ∗ **getSingleton** ()

Private Member Functions

- **UInt64** **getTimestamp** ()
- void **initFileDescriptors** ()
- void **getDisabledModules** ()
- void **getEnabledModules** ()
- bool **initIsLoggingEnabled** ()
- void **discoverCore** (**core_id_t** ∗core_id, bool ∗sim_thread)
- void **getFile** (**core_id_t** core_id, bool sim_thread, FILE ∗∗f, **Lock** ∗∗l)

Static Private Member Functions

- static void **parseModules** (std::set< String > &mods, String list)

Private Attributes

- **ErrorState** **_state**
- FILE ∗∗ **_coreFiles**
- FILE ∗∗ **_simFiles**
- **Lock** ∗ **_coreLocks**
- **Lock** ∗ **_simLocks**
- FILE ∗ **_systemFile**
- **Lock** **_systemLock**
- **core_id_t** **_coreCount**
- **UInt64** **_startTime**
- std::set< String > **_disabledModules**
- std::set< String > **_enabledModules**
- bool **_loggingEnabled**
- bool **_anyLoggingEnabled**

Static Private Attributes

- static const size_t **MODULE_LENGTH** = 20
- static **Log** ∗ **_singleton**

6.178.1 Detailed Description

Definition at line 14 of file log.h.

6.178.2 Member Enumeration Documentation

6.178.2.1 ErrorState

```
enum Log::ErrorState
```

Enumerator

None	
Warning	
Error	

Definition at line 22 of file log.h.

6.178.3 Constructor & Destructor Documentation

6.178.3.1 Log()

```
Log::Log (
            Config & config )
```

Definition at line 28 of file log.cc.

References _anyLoggingEnabled, _enabledModules, _loggingEnabled, _singleton, formatFileName(), getDisabledModules(), getEnabledModules(), CircularLog::init(), initFileDescriptors(), and initIsLoggingEnabled().

6.178.3.2 ~Log()

```
Log::~Log ( )
```

Definition at line 46 of file log.cc.

References _coreCount, _coreFiles, _coreLocks, _simFiles, _simLocks, _singleton, _systemFile, and CircularLog::fini().

6.178.4 Member Function Documentation

6.178.4.1 discoverCore()

```
void Log::discoverCore (
              core_id_t * core_id,
             bool * sim_thread )  [private]
```

Definition at line 189 of file log.cc.

References CoreManager::amiSimThread(), CoreManager::getCurrentCoreID(), and INVALID_CORE_ID.

Referenced by log().

6.178.4.2 getDisabledModules()

```
void Log::getDisabledModules ( )  [private]
```

Definition at line 139 of file log.cc.

References _disabledModules, LOG_PRINT_ERROR, and parseModules().

Referenced by Log().

6.178.4.3 getEnabledModules()

```
void Log::getEnabledModules ( )  [private]
```

Definition at line 153 of file log.cc.

References _enabledModules, LOG_PRINT_ERROR, and parseModules().

Referenced by Log().

6.178.4.4 getFile()

```
void Log::getFile (
              core_id_t core_id,
             bool sim_thread,
             FILE ** f,
              Lock ** l )  [private]
```

Definition at line 207 of file log.cc.

References _coreCount, _coreFiles, _coreLocks, _simFiles, _simLocks, _systemFile, _systemLock, formatFile↩Name(), and INVALID_CORE_ID.

Referenced by log().

6.178.4.5 getModule()

```
String Log::getModule (
            const char * filename )
```

Definition at line 261 of file log.cc.

References MODULE_LENGTH.

6.178.4.6 getSingleton()

```
Log * Log::getSingleton ( )  [static]
```

Definition at line 69 of file log.cc.

References _singleton.

Referenced by instructionCallback().

6.178.4.7 getTimestamp()

```
UInt64 Log::getTimestamp ( )  [private]
```

Definition at line 180 of file log.cc.

References _startTime.

Referenced by log().

6.178.4.8 initFileDescriptors()

```
void Log::initFileDescriptors ( )  [private]
```

Definition at line 85 of file log.cc.

References _coreCount, _coreFiles, _coreLocks, _simFiles, _simLocks, and _systemFile.

Referenced by Log().

6.178.4.9 initIsLoggingEnabled()

```
bool Log::initIsLoggingEnabled ( )  [private]
```

Definition at line 167 of file log.cc.

Referenced by Log().

6.178.4.10 isEnabled()

```
bool Log::isEnabled (
            const char * module )
```

Definition at line 75 of file log.cc.

References _anyLoggingEnabled, _disabledModules, _enabledModules, and _loggingEnabled.

6.178.4.11 isLoggingEnabled()

```
bool Log::isLoggingEnabled ( ) const  [inline]
```

Definition at line 32 of file log.h.

References _anyLoggingEnabled.

6.178.4.12 log()

```
void Log::log (
             ErrorState err,
            const char * source_file,
             SInt32 source_line,
            const char * format,
             ... )
```

Definition at line 295 of file log.cc.

References TLock< T_LockCreator >::acquire(), discoverCore(), Error, CircularLog::fini(), getFile(), getTimestamp(), INVALID_CORE_ID, None, TLock< T_LockCreator >::release(), and Warning.

6.178.4.13 parseModules()

```
void Log::parseModules (
            std::set< String > & mods,
            String list )  [static], [private]
```

Definition at line 102 of file log.cc.

References MODULE_LENGTH.

Referenced by getDisabledModules(), and getEnabledModules().

6.178.5 Member Data Documentation

6.178.5.1 _anyLoggingEnabled

```
bool Log::_anyLoggingEnabled  [private]
```

Definition at line 65 of file log.h.

Referenced by isEnabled(), isLoggingEnabled(), and Log().

6.178.5.2 _coreCount

```
core_id_t Log::_coreCount  [private]
```

Definition at line 60 of file log.h.

Referenced by getFile(), initFileDescriptors(), and ~Log().

6.178.5.3 _coreFiles

```
FILE** Log::_coreFiles  [private]
```

Definition at line 51 of file log.h.

Referenced by getFile(), initFileDescriptors(), and ~Log().

6.178.5.4 _coreLocks

```
Lock* Log::_coreLocks  [private]
```

Definition at line 53 of file log.h.

Referenced by getFile(), initFileDescriptors(), and ~Log().

6.178.5.5 _disabledModules

```
std::set<String> Log::_disabledModules  [private]
```

Definition at line 62 of file log.h.

Referenced by getDisabledModules(), and isEnabled().

6.178.5.6 _enabledModules

```
std::set<String> Log::_enabledModules  [private]
```

Definition at line 63 of file log.h.

Referenced by getEnabledModules(), isEnabled(), and Log().

6.178.5.7 _loggingEnabled

```
bool Log::_loggingEnabled  [private]
```

Definition at line 64 of file log.h.

Referenced by isEnabled(), and Log().

6.178.5.8 _simFiles

```
FILE** Log::_simFiles  [private]
```

Definition at line 52 of file log.h.

Referenced by getFile(), initFileDescriptors(), and ~Log().

6.178.5.9 _simLocks

```
Lock* Log::_simLocks  [private]
```

Definition at line 54 of file log.h.

Referenced by getFile(), initFileDescriptors(), and ~Log().

6.178.5.10 _singleton

```
Log * Log::_singleton  [static], [private]
```

Definition at line 72 of file log.h.

Referenced by getSingleton(), Log(), and ~Log().

6.178.5.11 _startTime

```
UInt64 Log::_startTime  [private]
```

Definition at line 61 of file log.h.

Referenced by getTimestamp().

6.178.5.12 _state

```
ErrorState Log::_state  [private]
```

Definition at line 48 of file log.h.

6.178.5.13 _systemFile

```
FILE* Log::_systemFile  [private]
```

Definition at line 57 of file log.h.

Referenced by getFile(), initFileDescriptors(), and ~Log().

6.178.5.14 _systemLock

```
Lock Log::_systemLock  [private]
```

Definition at line 58 of file log.h.

Referenced by getFile().

6.178.5.15 MODULE_LENGTH

```
const size_t Log::MODULE_LENGTH = 20  [static], [private]
```

Definition at line 70 of file log.h.

Referenced by getModule(), and parseModules().

The documentation for this class was generated from the following files:

- common/misc/ **log.h**
- common/misc/ **log.cc**

6.179 LoopProfiler::Loop Struct Reference

Public Member Functions

- **Loop** (**IntPtr** _eip, **UInt64** _size)
- **UInt64 weight** () const

Static Public Member Functions

- static bool **cmp** (const **Loop** ∗a, const **Loop** ∗b)

Public Attributes

- **IntPtr eip**
- **UInt64 size**
- **UInt64 count**

6.179.1 Detailed Description

Definition at line 12 of file loop_profiler.h.

6.179.2 Constructor & Destructor Documentation

6.179.2.1 Loop()

```
LoopProfiler::Loop::Loop (
              IntPtr _eip,
              UInt64 _size )  [inline]
```

Definition at line 14 of file loop_profiler.h.

6.179.3 Member Function Documentation

6.179.3.1 cmp()

```
static bool LoopProfiler::Loop::cmp (
             const  Loop * a,
             const  Loop * b )  [inline], [static]
```

Definition at line 15 of file loop_profiler.h.

References weight().

Referenced by LoopProfiler::∼LoopProfiler().

6.179.3.2 weight()

```
UInt64 LoopProfiler::Loop::weight ( ) const  [inline]
```

Definition at line 16 of file loop_profiler.h.

References count, and size.

Referenced by cmp().

6.179.4 Member Data Documentation

6.179.4.1 count

```
UInt64 LoopProfiler::Loop::count
```

Definition at line 20 of file loop_profiler.h.

Referenced by LoopProfiler::traceInstruction(), and weight().

6.179.4.2 eip

IntPtr LoopProfiler::Loop::eip

Definition at line 18 of file loop_profiler.h.

6.179.4.3 size

UInt64 LoopProfiler::Loop::size

Definition at line 19 of file loop_profiler.h.

Referenced by weight().

The documentation for this struct was generated from the following file:

- common/performance_model/instruction_tracers/ **loop_profiler.h**

6.180 LoopBranchPredictor Class Reference

#include <lpb.h>

Inheritance diagram for LoopBranchPredictor:

Classes

- class **Way**

Public Member Functions

- **LoopBranchPredictor** (**UInt32** entries, **UInt32** tag_bitwidth, **UInt32** ways)
- bool **predict** (**IntPtr** ip, **IntPtr** target)
- **BranchPredictorReturnValue lookup** (**IntPtr** ip, **IntPtr** target)
- void **update** (bool predicted, bool actual, **IntPtr** ip, **IntPtr** target)

Private Member Functions

- void **gen_index_tag** (**IntPtr** ip, **UInt32** &index, **UInt32** &tag)
- bool **prediction_match** (**UInt32** way, **UInt32** index, bool actual)

Private Attributes

- **UInt64 m_lru_use_count**
- **UInt32 m_num_ways**
- std::vector< **Way** > **m_ways**

6.180.1 Detailed Description

Definition at line 20 of file lpb.h.

6.180.2 Constructor & Destructor Documentation

6.180.2.1 LoopBranchPredictor()

```
LoopBranchPredictor::LoopBranchPredictor (
            UInt32 entries,
            UInt32 tag_bitwidth,
            UInt32 ways )  [inline]
```

Definition at line 25 of file lpb.h.

6.180.3 Member Function Documentation

6.180.3.1 gen_index_tag()

```
void LoopBranchPredictor::gen_index_tag (
            IntPtr ip,
            UInt32 & index,
            UInt32 & tag )  [inline], [private]
```

Definition at line 295 of file lpb.h.

Referenced by lookup(), and update().

6.180.3.2 lookup()

```
BranchPredictorReturnValue LoopBranchPredictor::lookup (
            IntPtr ip,
            IntPtr target )  [inline]
```

Definition at line 41 of file lpb.h.

References gen_index_tag(), BranchPredictorReturnValue::hit, BranchPredictorReturnValue::InvalidBranch, m_↩ lru_use_count, m_num_ways, m_ways, and BranchPredictorReturnValue::prediction.

Referenced by PentiumMBranchPredictor::predict().

6.180.3.3 predict()

```
bool LoopBranchPredictor::predict (
              IntPtr ip,
              IntPtr target )  [inline]
```

Definition at line 35 of file lpb.h.

6.180.3.4 prediction_match()

```
bool LoopBranchPredictor::prediction_match (
              UInt32 way,
              UInt32 index,
             bool actual )  [inline], [private]
```

Definition at line 302 of file lpb.h.

References debug_cout, and m_ways.

Referenced by update().

6.180.3.5 update()

```
void LoopBranchPredictor::update (
             bool predicted,
             bool actual,
              IntPtr ip,
              IntPtr target )  [inline]
```

Definition at line 78 of file lpb.h.

References debug_cout, gen_index_tag(), m_lru_use_count, m_num_ways, m_ways, and prediction_match().

Referenced by PentiumMBranchPredictor::update().

6.180.4 Member Data Documentation

6.180.4.1 m_lru_use_count

```
UInt64 LoopBranchPredictor::m_lru_use_count  [private]
```

Definition at line 332 of file lpb.h.

Referenced by lookup(), and update().

6.180.4.2 m_num_ways

```
UInt32 LoopBranchPredictor::m_num_ways [private]
```

Definition at line 333 of file lpb.h.

Referenced by lookup(), and update().

6.180.4.3 m_ways

```
std::vector< Way> LoopBranchPredictor::m_ways [private]
```

Definition at line 334 of file lpb.h.

Referenced by lookup(), prediction_match(), and update().

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **lpb.h**

6.181 LoopProfiler Class Reference

```
#include <loop_profiler.h>
```

Inheritance diagram for LoopProfiler:

Classes

- struct **Loop**

Public Member Functions

- **LoopProfiler** (const **Core** *core)
- virtual ~**LoopProfiler** ()
- virtual void **traceInstruction** (const **DynamicMicroOp** *uop, **uop_times_t** *times)

Private Types

- typedef std::vector< **Loop** * > **LoopList**
- typedef std::unordered_map< **IntPtr**, **LoopList** > **LoopMap**

Private Member Functions

- **Loop** * **findLoop** (**IntPtr** eip, **UInt64** size)
- **Loop** * **insertLoop** (**IntPtr** eip, **UInt64** size)

Private Attributes

- const **Core** * **m_core**
- **LoopMap m_loops**
- **UInt64 m_total_instructions**
- **IntPtr m_eip_last**

Additional Inherited Members

6.181.1 Detailed Description

Definition at line 9 of file loop_profiler.h.

6.181.2 Member Typedef Documentation

6.181.2.1 LoopList

```
typedef std::vector< Loop*>  LoopProfiler::LoopList  [private]
```

Definition at line 24 of file loop_profiler.h.

6.181.2.2 LoopMap

```
typedef std::unordered_map< IntPtr,  LoopList>  LoopProfiler::LoopMap  [private]
```

Definition at line 25 of file loop_profiler.h.

6.181.3 Constructor & Destructor Documentation

6.181.3.1 LoopProfiler()

```
LoopProfiler::LoopProfiler (
            const  Core * core )
```

Definition at line 7 of file loop_profiler.cc.

6.181.3.2 ~LoopProfiler()

```
LoopProfiler::~LoopProfiler ( )  [virtual]
```

Definition at line 14 of file loop_profiler.cc.

References LoopProfiler::Loop::cmp(), m_loops, and m_total_instructions.

6.181.4 Member Function Documentation

6.181.4.1 findLoop()

```
LoopProfiler::Loop * LoopProfiler::findLoop (
            IntPtr eip,
            UInt64 size )  [private]
```

Definition at line 40 of file loop_profiler.cc.

References m_loops.

Referenced by traceInstruction().

6.181.4.2 insertLoop()

```
LoopProfiler::Loop * LoopProfiler::insertLoop (
            IntPtr eip,
            UInt64 size )  [private]
```

Definition at line 51 of file loop_profiler.cc.

References m_loops.

Referenced by traceInstruction().

6.181.4.3 traceInstruction()

```
void LoopProfiler::traceInstruction (
            const  DynamicMicroOp * uop,
             uop_times_t * times )  [virtual]
```

Implements **InstructionTracer** (p. 636).

Definition at line 59 of file loop_profiler.cc.

References LoopProfiler::Loop::count, findLoop(), Instruction::getAddress(), MicroOp::getInstruction(), DynamicMicroOp::getMicroOp(), insertLoop(), MicroOp::isFirst(), m_eip_last, and m_total_instructions.

6.181.5 Member Data Documentation

6.181.5.1 m_core

```
const Core* LoopProfiler::m_core [private]
```

Definition at line 22 of file loop_profiler.h.

6.181.5.2 m_eip_last

```
IntPtr LoopProfiler::m_eip_last [private]
```

Definition at line 29 of file loop_profiler.h.

Referenced by traceInstruction().

6.181.5.3 m_loops

```
LoopMap LoopProfiler::m_loops [private]
```

Definition at line 26 of file loop_profiler.h.

Referenced by findLoop(), insertLoop(), and ~LoopProfiler().

6.181.5.4 m_total_instructions

```
UInt64 LoopProfiler::m_total_instructions [private]
```

Definition at line 27 of file loop_profiler.h.

Referenced by traceInstruction(), and ~LoopProfiler().

The documentation for this class was generated from the following files:

- common/performance_model/instruction_tracers/ **loop_profiler.h**
- common/performance_model/instruction_tracers/ **loop_profiler.cc**

6.182 LoopTracer Class Reference

`#include <loop_tracer.h>`

Inheritance diagram for LoopTracer:

Classes

- class **Instr**

Public Member Functions

- **LoopTracer** (const **Core** *core)
- virtual ~**LoopTracer** ()
- virtual void **traceInstruction** (const **DynamicMicroOp** *uop, **uop_times_t** *times)

Private Types

- typedef std::map< std::pair< **IntPtr**, **UInt8** >, **Instr** > **Instructions**

Private Attributes

- const **Core** * **m_core**
- const **IntPtr** **m_address_base**
- const **SInt64** **m_iter_start**
- const **SInt64** **m_iter_count**
- bool **m_active**
- **SInt64** **m_iter_current**
- **UInt64** **m_iter_instr**
- **UInt8** **m_instr_uop**
- uint64_t **m_cycle_min**
- uint64_t **m_cycle_max**
- uint64_t **m_disas_max**
- **Instructions** **m_instructions**

Additional Inherited Members

6.182.1 Detailed Description

Definition at line 13 of file loop_tracer.h.

6.182.2 Member Typedef Documentation

6.182.2.1 Instructions

```
typedef std::map<std::pair< IntPtr, UInt8>, Instr> LoopTracer::Instructions [private]
```

Definition at line 38 of file loop_tracer.h.

6.182.3 Constructor & Destructor Documentation

6.182.3.1 LoopTracer()

```
LoopTracer::LoopTracer (
            const Core * core )
```

Definition at line 8 of file loop_tracer.cc.

References LOG_ASSERT_ERROR, and m_iter_count.

6.182.3.2 ~LoopTracer()

```
LoopTracer::~LoopTracer ( ) [virtual]
```

Definition at line 25 of file loop_tracer.cc.

References m_cycle_max, m_cycle_min, m_disas_max, and m_instructions.

6.182.4 Member Function Documentation

6.182.4.1 traceInstruction()

```
void LoopTracer::traceInstruction (
            const DynamicMicroOp * uop,
             uop_times_t * times ) [virtual]
```

Implements **InstructionTracer** (p. 636).

Definition at line 58 of file loop_tracer.cc.

References SubsecondTime::divideRounded(), Instruction::getAddress(), Instruction::getDisassembly(), Core::getDvfsDomain(), MicroOp::getInstruction(), DynamicMicroOp::getMicroOp(), ComponentPeriod::getPeriod(), DynamicMicroOp::isFirst(), MicroOp::isLast(), InstructionTracer::uop_times_t::issue, m_active, m_address_base, m_core, m_cycle_max, m_cycle_min, m_disas_max, m_instr_uop, m_instructions, m_iter_count, m_iter_current, m_iter_instr, and m_iter_start.

6.182.5 Member Data Documentation

6.182.5.1 m_active

```
bool LoopTracer::m_active  [private]
```

Definition at line 22 of file loop_tracer.h.

Referenced by traceInstruction().

6.182.5.2 m_address_base

```
const  IntPtr LoopTracer::m_address_base  [private]
```

Definition at line 18 of file loop_tracer.h.

Referenced by traceInstruction().

6.182.5.3 m_core

```
const  Core* LoopTracer::m_core  [private]
```

Definition at line 16 of file loop_tracer.h.

Referenced by traceInstruction().

6.182.5.4 m_cycle_max

```
uint64_t LoopTracer::m_cycle_max  [private]
```

Definition at line 27 of file loop_tracer.h.

Referenced by traceInstruction(), and ~LoopTracer().

6.182.5.5 m_cycle_min

```
uint64_t LoopTracer::m_cycle_min  [private]
```

Definition at line 26 of file loop_tracer.h.

Referenced by traceInstruction(), and ~LoopTracer().

6.182.5.6 m_disas_max

```
uint64_t LoopTracer::m_disas_max  [private]
```

Definition at line 28 of file loop_tracer.h.

Referenced by traceInstruction(), and ~LoopTracer().

6.182.5.7 m_instr_uop

```
UInt8 LoopTracer::m_instr_uop  [private]
```

Definition at line 25 of file loop_tracer.h.

Referenced by traceInstruction().

6.182.5.8 m_instructions

```
Instructions LoopTracer::m_instructions  [private]
```

Definition at line 39 of file loop_tracer.h.

Referenced by traceInstruction(), and ~LoopTracer().

6.182.5.9 m_iter_count

```
const  SInt64 LoopTracer::m_iter_count  [private]
```

Definition at line 20 of file loop_tracer.h.

Referenced by LoopTracer(), and traceInstruction().

6.182.5.10 m_iter_current

```
SInt64 LoopTracer::m_iter_current  [private]
```

Definition at line 23 of file loop_tracer.h.

Referenced by traceInstruction().

6.182.5.11 m_iter_instr

```
UInt64 LoopTracer::m_iter_instr  [private]
```

Definition at line 24 of file loop_tracer.h.

Referenced by traceInstruction().

6.182.5.12 m_iter_start

```
const  SInt64 LoopTracer::m_iter_start  [private]
```

Definition at line 19 of file loop_tracer.h.

Referenced by traceInstruction().

The documentation for this class was generated from the following files:

- common/performance_model/instruction_tracers/ **loop_tracer.h**
- common/performance_model/instruction_tracers/ **loop_tracer.cc**

6.183 MagicServer::MagicMarkerType Struct Reference

```
#include <magic_server.h>
```

Public Attributes

- **thread_id_t thread_id**
- **core_id_t core_id**
- **UInt64 arg0**
- **UInt64 arg1**
- const char * **str**

6.183.1 Detailed Description

Definition at line 11 of file magic_server.h.

6.183.2 Member Data Documentation

6.183.2.1 arg0

```
UInt64 MagicServer::MagicMarkerType::arg0
```

Definition at line 14 of file magic_server.h.

Referenced by hookCallbackMagicMarkerType().

6.183.2.2 arg1

```
UInt64 MagicServer::MagicMarkerType::arg1
```

Definition at line 14 of file magic_server.h.

Referenced by hookCallbackMagicMarkerType().

6.183.2.3 core_id

```
core_id_t MagicServer::MagicMarkerType::core_id
```

Definition at line 13 of file magic_server.h.

Referenced by hookCallbackMagicMarkerType().

6.183.2.4 str

```
const char* MagicServer::MagicMarkerType::str
```

Definition at line 15 of file magic_server.h.

Referenced by hookCallbackMagicMarkerType().

6.183.2.5 thread_id

```
thread_id_t MagicServer::MagicMarkerType::thread_id
```

Definition at line 12 of file magic_server.h.

Referenced by hookCallbackMagicMarkerType().

The documentation for this struct was generated from the following file:

- common/system/ **magic_server.h**

6.184 MagicServer Class Reference

`#include <magic_server.h>`

Classes

- struct **MagicMarkerType**

Public Member Functions

- **MagicServer** ()
- ~**MagicServer** ()
- **UInt64 Magic** (**thread_id_t** thread_id, **core_id_t** core_id, **UInt64** cmd, **UInt64** arg0, **UInt64** arg1)
- bool **inROI** (void) const
- **UInt64 Magic_unlocked** (**thread_id_t** thread_id, **core_id_t** core_id, **UInt64** cmd, **UInt64** arg0, **UInt64** arg1)
- **UInt64 setFrequency** (**UInt64** core_number, **UInt64** freq_in_mhz)
- **UInt64 getFrequency** (**UInt64** core_number)
- void **enablePerformance** ()
- void **disablePerformance** ()
- **UInt64 setPerformance** (bool enabled)
- **UInt64 setInstrumentationMode** (**UInt64** sim_api_opt)
- void **setProgress** (float progress)

Static Public Member Functions

- static **UInt64 getGlobalInstructionCount** (void)

Private Attributes

- bool **m_performance_enabled**
- **Progress m_progress**

6.184.1 Detailed Description

Definition at line 7 of file magic_server.h.

6.184.2 Constructor & Destructor Documentation

6.184.2.1 MagicServer()

`MagicServer::MagicServer ( )`

Definition at line 15 of file magic_server.cc.

6.184.2.2 ~MagicServer()

```
MagicServer::~MagicServer ( )
```

Definition at line 20 of file magic_server.cc.

6.184.3 Member Function Documentation

6.184.3.1 disablePerformance()

```
void MagicServer::disablePerformance ( )
```

Definition at line 130 of file magic_server.cc.

References Simulator::disablePerformanceModels(), SubsecondTime::divideRounded(), Simulator::getClockSkewMinimizationServer(), Simulator::getConfig(), Simulator::getCoreManager(), FastforwardPerformanceModel::getFastforwardedTime(), PerformanceModel::getFastforwardPerformanceModel(), getGlobalInstructionCount(), PerformanceModel::getNonIdleElapsedTime(), SubsecondTime::getNS(), Timer::getTime(), InstMode::inst_mode_end, ninstrs_start, PinDetach(), t_start, and SubsecondTime::Zero().

Referenced by setPerformance().

6.184.3.2 enablePerformance()

```
void MagicServer::enablePerformance ( )
```

Definition at line 120 of file magic_server.cc.

References Simulator::enablePerformanceModels(), getGlobalInstructionCount(), InstMode::inst_mode_roi, ninstrs_start, Timer::start(), and t_start.

Referenced by setPerformance().

6.184.3.3 getFrequency()

```
UInt64 MagicServer::getFrequency (
            UInt64 core_number )
```

Definition at line 227 of file magic_server.cc.

References ComponentPeriod::getPeriodInFreqMHz().

Referenced by Magic_unlocked().

6.184.3.4 getGlobalInstructionCount()

```
UInt64 MagicServer::getGlobalInstructionCount (
            void  )  [static]
```

Definition at line 108 of file magic_server.cc.

Referenced by disablePerformance(), enablePerformance(), getIcount(), and Progress::record().

6.184.3.5 inROI()

```
bool MagicServer::inROI (
            void  ) const  [inline]
```

Definition at line 22 of file magic_server.h.

References m_performance_enabled.

6.184.3.6 Magic()

```
UInt64 MagicServer::Magic (
            thread_id_t thread_id,
            core_id_t core_id,
            UInt64 cmd,
            UInt64 arg0,
            UInt64 arg1 )
```

Definition at line 23 of file magic_server.cc.

References Simulator::getThreadManager(), and Magic_unlocked().

6.184.3.7 Magic_unlocked()

```
UInt64 MagicServer::Magic_unlocked (
            thread_id_t thread_id,
            core_id_t core_id,
            UInt64 cmd,
            UInt64 arg0,
            UInt64 arg1 )
```

Definition at line 30 of file magic_server.cc.

References Core::accessMemory(), StatsManager::EVENT_THREAD_NAME, Simulator::getConfig(), getFrequency(), HookType::HOOK_APPLICATION_ROI_BEGIN, HookType::HOOK_APPLICATION_ROI_END, HookType::HOOK_MAGIC_MARKER, HookType::HOOK_MAGIC_USER, LOG_ASSERT_ERROR, m_performance_enabled, SubsecondTime::MaxTime(), Core::MEM_MODELED_NONE, Core::NONE, Core::READ, Config::ROI_MAGIC, setFrequency(), setInstrumentationMode(), and setPerformance().

Referenced by Magic().

6.184.3.8 setFrequency()

```
UInt64 MagicServer::setFrequency (
            UInt64 core_number,
            UInt64 freq_in_mhz )
```

Definition at line 204 of file magic_server.cc.

References Core::BROKEN, ComponentPeriod::fromFreqHz(), Simulator::getDvfsManager(), HookType::HOOK_↩CPUFREQ_CHANGE, SubsecondTime::MaxTime(), and ThreadManager::STALL_BROKEN.

Referenced by Magic_unlocked().

6.184.3.9 setInstrumentationMode()

```
UInt64 MagicServer::setInstrumentationMode (
            UInt64 sim_api_opt )
```

Definition at line 237 of file magic_server.cc.

References InstMode::CACHE_ONLY, InstMode::DETAILED, InstMode::FAST_FORWARD, and LOG_PRINT_E↩RROR.

Referenced by Magic_unlocked().

6.184.3.10 setPerformance()

```
UInt64 MagicServer::setPerformance (
            bool enabled )
```

Definition at line 166 of file magic_server.cc.

References disablePerformance(), enablePerformance(), Timer::getTime(), HookType::HOOK_ROI_BEGIN, HookType::HOOK_ROI_END, logmem_enable(), logmem_write_allocations(), m_performance_enabled, Timer↩::start(), and t_start.

Referenced by Magic_unlocked(), and Simulator::∼Simulator().

6.184.3.11 setProgress()

```
void MagicServer::setProgress (
            float progress )  [inline]
```

Definition at line 36 of file magic_server.h.

References m_progress, and Progress::setProgress().

6.184.4 Member Data Documentation

6.184.4.1 m_performance_enabled

```
bool MagicServer::m_performance_enabled  [private]
```

Definition at line 39 of file magic_server.h.

Referenced by inROI(), Magic_unlocked(), and setPerformance().

6.184.4.2 m_progress

```
Progress MagicServer::m_progress  [private]
```

Definition at line 40 of file magic_server.h.

Referenced by setProgress().

The documentation for this class was generated from the following files:

- common/system/ **magic_server.h**
- common/system/ **magic_server.cc**

6.185 MemAccessInstruction Class Reference

```
#include <instruction.h>
```

Inheritance diagram for MemAccessInstruction:

Public Member Functions

- **MemAccessInstruction** (**SubsecondTime** cost, **IntPtr** address, **UInt32** data_size, bool is_fence)
- **IntPtr getDataAddress** () const
- **UInt32 getDataSize** () const
- bool **isFence** () const

Private Attributes

- **IntPtr m_address**
- **UInt32 m_data_size**
- bool **m_is_fence**

Additional Inherited Members

6.185.1 Detailed Description

Definition at line 202 of file instruction.h.

6.185.2 Constructor & Destructor Documentation

6.185.2.1 MemAccessInstruction()

```
MemAccessInstruction::MemAccessInstruction (
              SubsecondTime cost,
              IntPtr address,
              UInt32 data_size,
             bool is_fence )  [inline]
```

Definition at line 209 of file instruction.h.

6.185.3 Member Function Documentation

6.185.3.1 getDataAddress()

```
IntPtr MemAccessInstruction::getDataAddress ( ) const  [inline]
```

Definition at line 215 of file instruction.h.

References m_address.

Referenced by MicroOpPerformanceModel::handleInstruction().

6.185.3.2 getDataSize()

```
UInt32 MemAccessInstruction::getDataSize ( ) const  [inline]
```

Definition at line 216 of file instruction.h.

References m_data_size.

6.185.3.3 isFence()

```
bool MemAccessInstruction::isFence ( ) const  [inline]
```

Definition at line 217 of file instruction.h.

References m_is_fence.

Referenced by MicroOpPerformanceModel::handleInstruction().

6.185.4 Member Data Documentation

6.185.4.1 m_address

```
IntPtr MemAccessInstruction::m_address  [private]
```

Definition at line 205 of file instruction.h.

Referenced by getDataAddress().

6.185.4.2 m_data_size

```
UInt32 MemAccessInstruction::m_data_size  [private]
```

Definition at line 206 of file instruction.h.

Referenced by getDataSize().

6.185.4.3 m_is_fence

```
bool MemAccessInstruction::m_is_fence  [private]
```

Definition at line 207 of file instruction.h.

Referenced by isFence().

The documentation for this class was generated from the following file:

- common/performance_model/ **instruction.h**

6.186 FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock Class Reference

Public Member Functions

- **MemBlock** ()
- bool **isFull** () const
- void **clear** ()
- void * **allocate** (**Data_t** vectorIndex)
- void **deallocate** (**Data_t** *ptr)

Private Attributes

- **Data_t** * **block**
- **Data_t** **firstFreeUnitIndex**
- **Data_t** **allocatedElementsAmount**
- **Data_t** **endIndex**

6.186.1 Detailed Description

template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
class FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock

Definition at line 51 of file FSBAllocator.hh.

6.186.2 Constructor & Destructor Documentation

6.186.2.1 MemBlock()

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::MemBlock ( )  [inline]
```

Definition at line 57 of file FSBAllocator.hh.

6.186.3 Member Function Documentation

6.186.3.1 allocate()

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
void*  FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::allocate (
            Data_t vectorIndex )  [inline]
```

Definition at line 73 of file FSBAllocator.hh.

References FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::allocatedElements↩ Amount, FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::block, FSBAllocator_↩ ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::ElemSizeInDSize, FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::endIndex, FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::firstFreeUnitIndex, and FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::Unit↩ SizeInDSize.

6.186.3.2 clear()

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
void  FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::clear ( )  [inline]
```

Definition at line 68 of file FSBAllocator.hh.

References FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::firstFreeUnitIndex.

Referenced by FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::deallocate().

6.186.3.3 deallocate()

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
void  FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::deallocate (
            Data_t * ptr )  [inline]
```

Definition at line 97 of file FSBAllocator.hh.

References FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::allocatedElements↩ Amount, FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::block, FSBAllocator↩ _ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::clear(), and FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::firstFreeUnitIndex.

6.186.3.4 isFull()

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
bool  FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::isFull ( ) const
[inline]
```

Definition at line 63 of file FSBAllocator.hh.

References FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::allocatedElements↩ Amount, and FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlockElements.

6.186.4 Member Data Documentation

6.186.4.1 allocatedElementsAmount

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
Data_t FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::allocated↩
ElementsAmount [private]
```

Definition at line 54 of file FSBAllocator.hh.

Referenced by FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::allocate(), FSB↩ Allocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::deallocate(), and FSBAllocator_Elem↩ Allocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::isFull().

6.186.4.2 block

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
Data_t* FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::block [private]
```

Definition at line 53 of file FSBAllocator.hh.

Referenced by FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::allocate(), and F↩ SBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::deallocate().

6.186.4.3 endIndex

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
Data_t FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::endIndex
[private]
```

Definition at line 54 of file FSBAllocator.hh.

Referenced by FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::allocate().

6.186.4.4 firstFreeUnitIndex

```
template<unsigned ElemSize, unsigned MaxElem = 0, typename TypeToAlloc = UnspecifiedType>
Data_t FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::firstFree↩
UnitIndex [private]
```

Definition at line 54 of file FSBAllocator.hh.

Referenced by FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::allocate(), FS↩ BAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::clear(), and FSBAllocator_Elem↩ Allocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock::deallocate().

The documentation for this class was generated from the following file:

- common/misc/ **FSBAllocator.hh**

6.187 MemComponent Class Reference

`#include <mem_component.h>`

Public Types

- enum **component_t** {
  INVALID_MEM_COMPONENT = 0, **MIN_MEM_COMPONENT**, **CORE** = MIN_MEM_COMPONENT, **FI↩**
  RST_LEVEL_CACHE,
  L1_ICACHE = FIRST_LEVEL_CACHE, **L1_DCACHE**, **L2_CACHE**, **L3_CACHE**,
  L4_CACHE, **LAST_LEVEL_CACHE** = 20, **TAG_DIR**, **NUCA_CACHE**,
  DRAM_CACHE, **DRAM**, **MAX_MEM_COMPONENT** = DRAM, **NUM_MEM_COMPONENTS** = MAX_M↩
  EM_COMPONENT - MIN_MEM_COMPONENT + 1 }

6.187.1 Detailed Description

Definition at line 4 of file mem_component.h.

6.187.2 Member Enumeration Documentation

6.187.2.1 component_t

enum **MemComponent::component_t**

Enumerator

INVALID_MEM_COMPONENT	
MIN_MEM_COMPONENT	
CORE	
FIRST_LEVEL_CACHE	
L1_ICACHE	
L1_DCACHE	
L2_CACHE	
L3_CACHE	
L4_CACHE	
LAST_LEVEL_CACHE	
TAG_DIR	
NUCA_CACHE	
DRAM_CACHE	
DRAM	
MAX_MEM_COMPONENT	
NUM_MEM_COMPONENTS	

Definition at line 7 of file mem_component.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/ **mem_component.h**

6.188 MemGuard Class Reference

`#include <memguard.h>`

Public Member Functions

- **MemGuard** ()
- **MemGuard** (const **MemGuard** &m)
- **MemGuard** & **operator=** (const **MemGuard** &m)
- ∼**MemGuard** ()

Private Member Functions

- void **protect** ()
- void **unprotect** ()
- bool **isAligned** ()
- void ∗ **pagePointer** ()

Private Attributes

- char **m_guard** [2 ∗ **PAGE_SIZE**]

6.188.1 Detailed Description

Definition at line 10 of file memguard.h.

6.188.2 Constructor & Destructor Documentation

6.188.2.1 MemGuard() [1/2]

`MemGuard::MemGuard ( )`

Definition at line 5 of file memguard.cc.

References protect().

6.188.2.2 MemGuard() [2/2]

```
MemGuard::MemGuard (
            const  MemGuard & m )
```

Definition at line 10 of file memguard.cc.

References protect().

6.188.2.3 ~MemGuard()

```
MemGuard::~MemGuard ( )
```

Definition at line 22 of file memguard.cc.

References unprotect().

6.188.3 Member Function Documentation

6.188.3.1 isAligned()

```
bool MemGuard::isAligned ( )  [private]
```

Definition at line 37 of file memguard.cc.

References m_guard, and PAGE_SIZE.

Referenced by protect(), and unprotect().

6.188.3.2 operator=()

```
MemGuard & MemGuard::operator= (
            const  MemGuard & m )
```

Definition at line 16 of file memguard.cc.

6.188.3.3 pagePointer()

```
void * MemGuard::pagePointer ( )  [private]
```

Definition at line 42 of file memguard.cc.

References m_guard, and PAGE_SIZE.

Referenced by protect(), and unprotect().

6.188.3.4 protect()

```
void MemGuard::protect ( )  [private]
```

Definition at line 27 of file memguard.cc.

References isAligned(), PAGE_SIZE, and pagePointer().

Referenced by MemGuard().

6.188.3.5 unprotect()

```
void MemGuard::unprotect ( )  [private]
```

Definition at line 32 of file memguard.cc.

References isAligned(), PAGE_SIZE, and pagePointer().

Referenced by ~MemGuard().

6.188.4 Member Data Documentation

6.188.4.1 m_guard

```
char MemGuard::m_guard[2 * PAGE_SIZE]  [private]
```

Definition at line 12 of file memguard.h.

Referenced by isAligned(), and pagePointer().

The documentation for this class was generated from the following files:

- common/misc/ **memguard.h**
- common/misc/ **memguard.cc**

6.189 MemoryDependencies Class Reference

`#include <memory_dependencies.h>`

Classes

- struct **Producer**

Public Member Functions

- **MemoryDependencies** ()
- ~**MemoryDependencies** ()
- void **setDependencies** (**DynamicMicroOp** &microOp, uint64_t lowestValidSequenceNumber)
- void **clear** ()

Private Member Functions

- void **add** (uint64_t sequenceNumber, uint64_t address)
- uint64_t **find** (uint64_t address)
- void **clean** (uint64_t lowestValidSequenceNumber)

Private Attributes

- **CircularQueue**< **Producer** > **producers**
- uint64_t **membar**

6.189.1 Detailed Description

Definition at line 8 of file memory_dependencies.h.

6.189.2 Constructor & Destructor Documentation

6.189.2.1 MemoryDependencies()

`MemoryDependencies::MemoryDependencies ( )`

Definition at line 3 of file memory_dependencies.cc.

References clear().

6.189.2.2 ~MemoryDependencies()

```
MemoryDependencies::~MemoryDependencies ( )
```

Definition at line 9 of file memory_dependencies.cc.

6.189.3 Member Function Documentation

6.189.3.1 add()

```
void MemoryDependencies::add (
            uint64_t sequenceNumber,
            uint64_t address )  [private]
```

Definition at line 59 of file memory_dependencies.cc.

References producers.

Referenced by setDependencies().

6.189.3.2 clean()

```
void MemoryDependencies::clean (
            uint64_t lowestValidSequenceNumber )  [private]
```

Definition at line 74 of file memory_dependencies.cc.

References producers.

Referenced by setDependencies().

6.189.3.3 clear()

```
void MemoryDependencies::clear ( )
```

Definition at line 80 of file memory_dependencies.cc.

References INVALID_SEQNR, membar, and producers.

Referenced by Windows::clear(), and MemoryDependencies().

6.189.3.4 find()

```
uint64_t MemoryDependencies::find (
            uint64_t address )  [private]
```

Definition at line 65 of file memory_dependencies.cc.

References INVALID_SEQNR, and producers.

Referenced by setDependencies().

6.189.3.5 setDependencies()

```
void MemoryDependencies::setDependencies (
             DynamicMicroOp & microOp,
            uint64_t lowestValidSequenceNumber )
```

Definition at line 13 of file memory_dependencies.cc.

References add(), DynamicMicroOp::addDependency(), clean(), find(), DynamicMicroOp::getLoadAccess(), DynamicMicroOp::getMicroOp(), DynamicMicroOp::getSequenceNumber(), DynamicMicroOp::getStoreAccess(), INVALID_SEQNR, MicroOp::isLoad(), MicroOp::isMemBarrier(), MicroOp::isStore(), membar, and Memory::↩ Access::phys.

Referenced by Windows::add(), RobSmtTimer::setDependencies(), and RobTimer::simulate().

6.189.4 Member Data Documentation

6.189.4.1 membar

```
uint64_t MemoryDependencies::membar  [private]
```

Definition at line 25 of file memory_dependencies.h.

Referenced by clear(), and setDependencies().

6.189.4.2 producers

```
CircularQueue< Producer> MemoryDependencies::producers  [private]
```

Definition at line 24 of file memory_dependencies.h.

Referenced by add(), clean(), clear(), and find().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/micro_op/ **memory_dependencies.h**
- common/performance_model/performance_models/micro_op/ **memory_dependencies.cc**

6.190 DynamicInstruction::MemoryInfo Struct Reference

`#include <dynamic_instruction.h>`

Public Attributes

- bool **executed**
- **Operand::Direction dir**
- **IntPtr addr**
- **UInt32 size**
- **UInt32 num_misses**
- **SubsecondTime latency**
- **HitWhere::where_t hit_where**

6.190.1 Detailed Description

Definition at line 30 of file dynamic_instruction.h.

6.190.2 Member Data Documentation

6.190.2.1 addr

IntPtr DynamicInstruction::MemoryInfo::addr

Definition at line 33 of file dynamic_instruction.h.

Referenced by DynamicInstruction::accessMemory(), DynamicInstruction::addMemory(), and MicroOp↩ PerformanceModel::handleInstruction().

6.190.2.2 dir

Operand::Direction DynamicInstruction::MemoryInfo::dir

Definition at line 32 of file dynamic_instruction.h.

Referenced by DynamicInstruction::addMemory(), OneIPCPerformanceModel::handleInstruction(), and MicroOp↩ PerformanceModel::handleInstruction().

6.190.2.3 executed

```
bool DynamicInstruction::MemoryInfo::executed
```

Definition at line 31 of file dynamic_instruction.h.

Referenced by DynamicInstruction::addMemory().

6.190.2.4 hit_where

```
HitWhere::where_t DynamicInstruction::MemoryInfo::hit_where
```

Definition at line 37 of file dynamic_instruction.h.

Referenced by DynamicInstruction::accessMemory(), DynamicInstruction::addMemory(), OneIPCPerformance↩ Model::handleInstruction(), and MicroOpPerformanceModel::handleInstruction().

6.190.2.5 latency

```
SubsecondTime DynamicInstruction::MemoryInfo::latency
```

Definition at line 36 of file dynamic_instruction.h.

Referenced by DynamicInstruction::accessMemory(), DynamicInstruction::addMemory(), OneIPCPerformance↩ Model::handleInstruction(), and MicroOpPerformanceModel::handleInstruction().

6.190.2.6 num_misses

```
UInt32 DynamicInstruction::MemoryInfo::num_misses
```

Definition at line 35 of file dynamic_instruction.h.

Referenced by DynamicInstruction::addMemory().

6.190.2.7 size

```
UInt32 DynamicInstruction::MemoryInfo::size
```

Definition at line 34 of file dynamic_instruction.h.

Referenced by DynamicInstruction::accessMemory(), and DynamicInstruction::addMemory().

The documentation for this struct was generated from the following file:

- common/performance_model/ **dynamic_instruction.h**

6.191 ParametricDramDirectoryMSI::MemoryManager Class Reference

`#include <memory_manager.h>`

Inheritance diagram for ParametricDramDirectoryMSI::MemoryManager:

Public Member Functions

- **MemoryManager** (**Core** ∗core, **Network** ∗network, **ShmemPerfModel** ∗shmem_perf_model)
- ∼**MemoryManager** ()
- **UInt64 getCacheBlockSize** () const
- **Cache** ∗ **getCache** (**MemComponent::component_t** mem_component)
- **Cache** ∗ **getL1ICache** ()
- **Cache** ∗ **getL1DCache** ()
- **Cache** ∗ **getLastLevelCache** ()
- **PrL1PrL2DramDirectoryMSI::DramDirectoryCache** ∗ **getDramDirectoryCache** ()
- **PrL1PrL2DramDirectoryMSI::DramCntlr** ∗ **getDramCntlr** ()
- **AddressHomeLookup** ∗ **getTagDirectoryHomeLookup** ()
- **AddressHomeLookup** ∗ **getDramControllerHomeLookup** ()
- **CacheCntlr** ∗ **getCacheCntlrAt** (**core_id_t** core_id, **MemComponent::component_t** mem_component)
- void **setCacheCntlrAt** (**core_id_t** core_id, **MemComponent::component_t** mem_component, **Cache**↩ **Cntlr** ∗cache_cntlr)
- **HitWhere::where_t coreInitiateMemoryAccess** (**MemComponent::component_t** mem_component, **Core::lock_signal_t** lock_signal, **Core::mem_op_t** mem_op_type, **IntPtr** address, **UInt32** offset, **Byte** ∗data_buf, **UInt32** data_length, **Core::MemModeled** modeled)
- void **handleMsgFromNetwork** (**NetPacket** &packet)
- void **sendMsg** (**PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t** msg_type, **MemComponent**↩ **::component_t** sender_mem_component, **MemComponent::component_t** receiver_mem_component, **core_id_t** requester, **core_id_t** receiver, **IntPtr** address, **Byte** ∗data_buf=NULL, **UInt32** data_length=0, **HitWhere::where_t** where= **HitWhere::UNKNOWN**, **ShmemPerf** ∗perf=NULL, **ShmemPerfModel::**↩ **Thread_t** thread_num= **ShmemPerfModel::NUM_CORE_THREADS**)
- void **broadcastMsg** (**PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t** msg_type, **MemComponent**↩ **::component_t** sender_mem_component, **MemComponent::component_t** receiver_mem_component, **core_id_t** requester, **IntPtr** address, **Byte** ∗data_buf=NULL, **UInt32** data_length=0, **ShmemPerf** ∗perf=NULL, **ShmemPerfModel::Thread_t** thread_num= **ShmemPerfModel::NUM_CORE_THREADS**)
- **SubsecondTime getL1HitLatency** (void)
- void **addL1Hits** (bool icache, **Core::mem_op_t** mem_op_type, **UInt64** hits)
- void **enableModels** ()
- void **disableModels** ()
- **core_id_t getShmemRequester** (const void ∗pkt_data)
- **UInt32 getModeledLength** (const void ∗pkt_data)
- **SubsecondTime getCost** (**MemComponent::component_t** mem_component, **CachePerfModel::**↩ **CacheAccess_t** access_type)
- void **incrElapsedTime** (**SubsecondTime** latency, **ShmemPerfModel::Thread_t** thread_num= **Shmem**↩ **PerfModel::NUM_CORE_THREADS**)
- void **incrElapsedTime** (**MemComponent::component_t** mem_component, **CachePerfModel::Cache**↩ **Access_t** access_type, **ShmemPerfModel::Thread_t** thread_num= **ShmemPerfModel::NUM_CORE_T**↩ **HREADS**)

Private Member Functions

- void **accessTLB** (**TLB** *tlb, **IntPtr** address, bool isIfetch, **Core::MemModeled** modeled)

Private Attributes

- **CacheCntlr** * **m_cache_cntlrs** [**MemComponent::LAST_LEVEL_CACHE**+1]
- **NucaCache** * **m_nuca_cache**
- **DramCache** * **m_dram_cache**
- **PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr** * **m_dram_directory_cntlr**
- **PrL1PrL2DramDirectoryMSI::DramCntlr** * **m_dram_cntlr**
- **AddressHomeLookup** * **m_tag_directory_home_lookup**
- **AddressHomeLookup** * **m_dram_controller_home_lookup**
- **TLB** * **m_itlb**
- **TLB** * **m_dtlb**
- **TLB** * **m_stlb**
- **ComponentLatency** **m_tlb_miss_penalty**
- bool **m_tlb_miss_parallel**
- **core_id_t** **m_core_id_master**
- bool **m_tag_directory_present**
- bool **m_dram_cntlr_present**
- **Semaphore** * **m_user_thread_sem**
- **Semaphore** * **m_network_thread_sem**
- **UInt32** **m_cache_block_size**
- **MemComponent::component_t** **m_last_level_cache**
- bool **m_enabled**
- **ShmemPerf** **m_dummy_shmem_perf**
- **CachePerfModel** * **m_cache_perf_models** [**MemComponent::LAST_LEVEL_CACHE**+1]

Static Private Attributes

- static **CacheCntlrMap** **m_all_cache_cntlrs**

Additional Inherited Members

6.191.1 Detailed Description

Definition at line 29 of file memory_manager.h.

6.191.2 Constructor & Destructor Documentation

6.191.2.1 MemoryManager()

```
ParametricDramDirectoryMSI::MemoryManager::MemoryManager (
              Core * core,
              Network * network,
              ShmemPerfModel * shmem_perf_model )
```

Definition at line 33 of file memory_manager.cc.

References CachePerfModel::create(), MemComponent::FIRST_LEVEL_CACHE, floorLog2(), getCacheBlock↩Size(), MemoryManagerBase::getCore(), MemoryManagerBase::getCoreListWithMemoryControllers(), Core↩::getDvfsDomain(), Core::getId(), MemoryManagerBase::getNetwork(), MemoryManagerBase::getShmemPerf↩Model(), Core::getTopologyInfo(), itostr(), MemComponent::L1_DCACHE, MemComponent::L1_ICACHE, Mem↩Component::L2_CACHE, LOG_ASSERT_ERROR, LOG_PRINT_ERROR, m_cache_block_size, m_cache_cntlrs, m_cache_perf_models, m_core_id_master, m_dram_cache, m_dram_cntlr, m_dram_cntlr_present, m_dram_↩controller_home_lookup, m_dram_directory_cntlr, m_dtlb, m_itlb, m_last_level_cache, m_network_thread_sem, m_nuca_cache, m_stlb, m_tag_directory_home_lookup, m_tag_directory_present, m_tlb_miss_parallel, m_tlb_↩miss_penalty, m_user_thread_sem, MemoryManagerNetworkCallback(), Network::registerCallback(), setCache↩CntlrAt(), ParametricDramDirectoryMSI::CacheCntlr::setNextCacheCntlr(), ParametricDramDirectoryMSI::Cache↩Cntlr::setPrevCacheCntlrs(), TopologyInfo::setup(), ParametricDramDirectoryMSI::CacheParameters::shared_↩cores, and SHARED_MEM_1.

6.191.2.2 ~MemoryManager()

```
ParametricDramDirectoryMSI::MemoryManager::~MemoryManager ( )
```

Definition at line 378 of file memory_manager.cc.

References MemComponent::FIRST_LEVEL_CACHE, MemoryManagerBase::getNetwork(), m_cache_cntlrs, m↩_cache_perf_models, m_dram_cache, m_dram_cntlr, m_dram_controller_home_lookup, m_dram_directory_cntlr, m_dtlb, m_itlb, m_last_level_cache, m_network_thread_sem, m_nuca_cache, m_stlb, m_tag_directory_home_↩lookup, m_user_thread_sem, SHARED_MEM_1, and Network::unregisterCallback().

6.191.3 Member Function Documentation

6.191.3.1 accessTLB()

```
void ParametricDramDirectoryMSI::MemoryManager::accessTLB (
              TLB * tlb,
              IntPtr address,
             bool isIfetch,
              Core::MemModeled modeled )  [private]
```

Definition at line 590 of file memory_manager.cc.

References ShmemPerfModel::_USER_THREAD, MemoryManagerBase::getCore(), ComponentLatency::get↩Latency(), Core::getPerformanceModel(), MemoryManagerBase::getShmemPerfModel(), incrElapsedTime(), ParametricDramDirectoryMSI::TLB::lookup(), m_tlb_miss_parallel, m_tlb_miss_penalty, Core::MEM_MODEL↩ED_COUNT, Core::MEM_MODELED_NONE, PerformanceModel::queuePseudoInstruction(), and Subsecond↩Time::Zero().

Referenced by coreInitiateMemoryAccess().

6.191.3.2 addL1Hits()

```
void ParametricDramDirectoryMSI::MemoryManager::addL1Hits (
            bool icache,
             Core::mem_op_t mem_op_type,
             UInt64 hits ) [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 747).

Definition at line 100 of file memory_manager.h.

References MemComponent::L1_DCACHE, MemComponent::L1_ICACHE, and m_cache_cntlrs.

6.191.3.3 broadcastMsg()

```
void ParametricDramDirectoryMSI::MemoryManager::broadcastMsg (
             PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t msg_type,
             MemComponent::component_t sender_mem_component,
             MemComponent::component_t receiver_mem_component,
             core_id_t requester,
             IntPtr address,
             Byte * data_buf = NULL,
             UInt32 data_length = 0,
             ShmemPerf * perf = NULL,
             ShmemPerfModel::Thread_t thread_num = ShmemPerfModel::NUM_CORE_THREADS ) [virtual]
```

Implements **MemoryManagerBase** (p. 747).

Definition at line 565 of file memory_manager.cc.

References NetPacket::BROADCAST, MemoryManagerBase::getCore(), ShmemPerfModel::getElapsedTime(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgLen(), MemoryManagerBase::getNetwork(), MemoryManager↩Base::getShmemPerfModel(), LOG_PRINT, m_core_id_master, m_enabled, PrL1PrL2DramDirectoryMSI::↩ShmemMsg::makeMsgBuf(), MYLOG, Network::netSend(), SHARED_MEM_1, and ShmemPerf::updateTime().

6.191.3.4 coreInitiateMemoryAccess()

```
HitWhere::where_t ParametricDramDirectoryMSI::MemoryManager::coreInitiateMemoryAccess (
             MemComponent::component_t mem_component,
             Core::lock_signal_t lock_signal,
             Core::mem_op_t mem_op_type,
             IntPtr address,
             UInt32 offset,
             Byte * data_buf,
             UInt32 data_length,
             Core::MemModeled modeled ) [virtual]
```

Implements **MemoryManagerBase** (p. 748).

Definition at line 418 of file memory_manager.cc.

References accessTLB(), MemComponent::L1_DCACHE, MemComponent::L1_ICACHE, LOG_ASSERT_ERR↩OR, m_cache_cntlrs, m_dtlb, m_itlb, m_last_level_cache, Core::MEM_MODELED_COUNT, Core::MEM_MODE↩LED_NONE, and ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore().

6.191.3.5 disableModels()

```
void ParametricDramDirectoryMSI::MemoryManager::disableModels ( )  [virtual]
```

Implements **MemoryManagerBase** (p. 749).

Definition at line 648 of file memory_manager.cc.

References DramPerfModel::disable(), MemComponent::FIRST_LEVEL_CACHE, PrL1PrL2DramDirectoryMSI::↩ DramCntlr::getDramPerfModel(), m_cache_cntlrs, m_cache_perf_models, m_dram_cntlr, m_dram_cntlr_present, m_enabled, and m_last_level_cache.

6.191.3.6 enableModels()

```
void ParametricDramDirectoryMSI::MemoryManager::enableModels ( )  [virtual]
```

Implements **MemoryManagerBase** (p. 749).

Definition at line 633 of file memory_manager.cc.

References DramPerfModel::enable(), MemComponent::FIRST_LEVEL_CACHE, PrL1PrL2DramDirectoryMSI::↩ DramCntlr::getDramPerfModel(), m_cache_cntlrs, m_cache_perf_models, m_dram_cntlr, m_dram_cntlr_present, m_enabled, and m_last_level_cache.

6.191.3.7 getCache()

```
Cache* ParametricDramDirectoryMSI::MemoryManager::getCache (
            MemComponent::component_t mem_component )  [inline]
```

Definition at line 71 of file memory_manager.h.

References ParametricDramDirectoryMSI::CacheCntlr::getCache(), MemComponent::LAST_LEVEL_CACHE, m↩ _cache_cntlrs, and m_last_level_cache.

Referenced by getL1DCache(), getL1ICache(), and getLastLevelCache().

6.191.3.8 getCacheBlockSize()

```
UInt64 ParametricDramDirectoryMSI::MemoryManager::getCacheBlockSize ( ) const  [inline],
[virtual]
```

Implements **MemoryManagerBase** (p. 749).

Definition at line 69 of file memory_manager.h.

References m_cache_block_size.

Referenced by MemoryManager().

6.191.3.9 getCacheCntlrAt()

```
CacheCntlr* ParametricDramDirectoryMSI::MemoryManager::getCacheCntlrAt (
            core_id_t core_id,
            MemComponent::component_t mem_component ) [inline]
```

Definition at line 82 of file memory_manager.h.

References m_all_cache_cntlrs.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr().

6.191.3.10 getCost()

```
SubsecondTime ParametricDramDirectoryMSI::MemoryManager::getCost (
            MemComponent::component_t mem_component,
            CachePerfModel::CacheAccess_t access_type )
```

Definition at line 611 of file memory_manager.cc.

References CachePerfModel::getLatency(), MemComponent::INVALID_MEM_COMPONENT, m_cache_perf_↩ models, and SubsecondTime::Zero().

Referenced by incrElapsedTime().

6.191.3.11 getDramCntlr()

```
PrL1PrL2DramDirectoryMSI::DramCntlr* ParametricDramDirectoryMSI::MemoryManager::getDramCntlr
( ) [inline]
```

Definition at line 78 of file memory_manager.h.

References m_dram_cntlr.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::flushBlock().

6.191.3.12 getDramControllerHomeLookup()

```
AddressHomeLookup* ParametricDramDirectoryMSI::MemoryManager::getDramControllerHomeLookup ( )
[inline]
```

Definition at line 80 of file memory_manager.h.

References m_dram_controller_home_lookup.

6.191.3.13 getDramDirectoryCache()

```
PrL1PrL2DramDirectoryMSI::DramDirectoryCache* ParametricDramDirectoryMSI::MemoryManager::get↩
DramDirectoryCache ( )  [inline]
```

Definition at line 77 of file memory_manager.h.

References PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::getDramDirectoryCache(), and m_dram_directory↩
_cntlr.

6.191.3.14 getL1DCache()

```
Cache* ParametricDramDirectoryMSI::MemoryManager::getL1DCache ( )  [inline]
```

Definition at line 75 of file memory_manager.h.

References getCache(), and MemComponent::L1_DCACHE.

6.191.3.15 getL1HitLatency()

```
SubsecondTime ParametricDramDirectoryMSI::MemoryManager::getL1HitLatency (
            void  )  [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 750).

Definition at line 99 of file memory_manager.h.

References CachePerfModel::ACCESS_CACHE_DATA_AND_TAGS, CachePerfModel::getLatency(), Mem↩
Component::L1_ICACHE, and m_cache_perf_models.

6.191.3.16 getL1ICache()

```
Cache* ParametricDramDirectoryMSI::MemoryManager::getL1ICache ( )  [inline]
```

Definition at line 74 of file memory_manager.h.

References getCache(), and MemComponent::L1_ICACHE.

6.191.3.17 getLastLevelCache()

```
Cache* ParametricDramDirectoryMSI::MemoryManager::getLastLevelCache ( )  [inline]
```

Definition at line 76 of file memory_manager.h.

References getCache(), and MemComponent::LAST_LEVEL_CACHE.

6.191.3.18 getModeledLength()

```
UInt32 ParametricDramDirectoryMSI::MemoryManager::getModeledLength (
            const void * pkt_data )  [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 750).

Definition at line 110 of file memory_manager.h.

6.191.3.19 getShmemRequester()

```
core_id_t ParametricDramDirectoryMSI::MemoryManager::getShmemRequester (
            const void * pkt_data )  [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 751).

Definition at line 107 of file memory_manager.h.

6.191.3.20 getTagDirectoryHomeLookup()

```
AddressHomeLookup* ParametricDramDirectoryMSI::MemoryManager::getTagDirectoryHomeLookup ( )
[inline]
```

Definition at line 79 of file memory_manager.h.

References m_tag_directory_home_lookup.

6.191.3.21 handleMsgFromNetwork()

```
void ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork (
             NetPacket & packet )  [virtual]
```

Implements **MemoryManagerBase** (p. 751).

Definition at line 444 of file memory_manager.cc.

References ShmemPerfModel::_SIM_THREAD, NetPacket::data, MemComponent::DRAM, PrL1PrL2Dram↩DirectoryMSI::ShmemMsg::getAddress(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataBuf(), PrL1PrL2↩DramDirectoryMSI::ShmemMsg::getDataLength(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType(), Pr↩L1PrL2DramDirectoryMSI::ShmemMsg::getPerf(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getReceiverMem↩Component(), PrL1PrL2DramDirectoryMSI::ShmemMsg::getSenderMemComponent(), PrL1PrL2DramDirectory↩MSI::ShmemMsg::getShmemMsg(), MemoryManagerBase::getShmemPerfModel(), PrL1PrL2DramDirectoryM↩SI::DramDirectoryCntlr::handleMsgFromDRAM(), ParametricDramDirectoryMSI::CacheCntlr::handleMsgFrom↩DramDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::handleMsgFromL2Cache(), DramCntlr↩Interface::handleMsgFromTagDirectory(), MemComponent::L2_CACHE, MemComponent::LAST_LEVEL_CACHE, LOG_ASSERT_ERROR, LOG_PRINT, LOG_PRINT_ERROR, m_cache_cntlrs, m_dram_cache, m_dram_cntlr, m_dram_cntlr_present, m_dram_directory_cntlr, m_dummy_shmem_perf, m_enabled, m_last_level_cache, m↩_tag_directory_present, MYLOG, NetPacket::receiver, NetPacket::sender, ShmemPerfModel::setElapsedTime(), MemComponent::TAG_DIR, NetPacket::time, and ShmemPerf::updatePacket().

6.191.3.22 incrElapsedTime() [1/2]

```
void ParametricDramDirectoryMSI::MemoryManager::incrElapsedTime (
              MemComponent::component_t mem_component,
              CachePerfModel::CacheAccess_t access_type,
              ShmemPerfModel::Thread_t thread_num =  ShmemPerfModel::NUM_CORE_THREADS )
```

Definition at line 627 of file memory_manager.cc.

References getCost(), and incrElapsedTime().

6.191.3.23 incrElapsedTime() [2/2]

```
void ParametricDramDirectoryMSI::MemoryManager::incrElapsedTime (
              SubsecondTime latency,
              ShmemPerfModel::Thread_t thread_num =  ShmemPerfModel::NUM_CORE_THREADS )
```

Definition at line 620 of file memory_manager.cc.

References MemoryManagerBase::getShmemPerfModel(), ShmemPerfModel::incrElapsedTime(), itostr(), and M↩YLOG.

Referenced by accessTLB(), ParametricDramDirectoryMSI::CacheCntlr::copyDataFromNextLevel(), incrElapsed↩Time(), ParametricDramDirectoryMSI::CacheCntlr::incrementQBSLookupCost(), ParametricDramDirectoryM↩SI::CacheCntlr::insertCacheBlock(), ParametricDramDirectoryMSI::CacheCntlr::processFlushReqFromDram↩Directory(), ParametricDramDirectoryMSI::CacheCntlr::processInvReqFromDramDirectory(), ParametricDram↩DirectoryMSI::CacheCntlr::processMemOpFromCore(), ParametricDramDirectoryMSI::CacheCntlr::process↩ShmemReqFromPrevCache(), and ParametricDramDirectoryMSI::CacheCntlr::processWbReqFromDram↩Directory().

6.191.3.24 sendMsg()

```
void ParametricDramDirectoryMSI::MemoryManager::sendMsg (
              PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t msg_type,
              MemComponent::component_t sender_mem_component,
              MemComponent::component_t receiver_mem_component,
              core_id_t requester,
              core_id_t receiver,
              IntPtr address,
              Byte * data_buf = NULL,
              UInt32 data_length = 0,
              HitWhere::where_t where =  HitWhere::UNKNOWN,
              ShmemPerf * perf = NULL,
              ShmemPerfModel::Thread_t thread_num =  ShmemPerfModel::NUM_CORE_THREADS )  [virtual]
```

Implements **MemoryManagerBase** (p. 752).

Definition at line 539 of file memory_manager.cc.

References MemoryManagerBase::getCore(), ShmemPerfModel::getElapsedTime(), PrL1PrL2DramDirectoryMS↩I::ShmemMsg::getMsgLen(), MemoryManagerBase::getNetwork(), MemoryManagerBase::getShmemPerfModel(),

LOG_PRINT, m_core_id_master, m_enabled, PrL1PrL2DramDirectoryMSI::ShmemMsg::makeMsgBuf(), MYLOG, Network::netSend(), PrL1PrL2DramDirectoryMSI::ShmemMsg::setWhere(), SHARED_MEM_1, and ShmemPerf↩ ::updateTime().

Referenced by ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), ParametricDramDirectoryMSI↩ ::CacheCntlr::processExReqToDirectory(), ParametricDramDirectoryMSI::CacheCntlr::processFlushReqFrom↩ DramDirectory(), ParametricDramDirectoryMSI::CacheCntlr::processInvReqFromDramDirectory(), Parametric↩ DramDirectoryMSI::CacheCntlr::processShReqToDirectory(), ParametricDramDirectoryMSI::CacheCntlr::process↩ UpgradeReqToDirectory(), and ParametricDramDirectoryMSI::CacheCntlr::processWbReqFromDramDirectory().

6.191.3.25 setCacheCntlrAt()

```
void ParametricDramDirectoryMSI::MemoryManager::setCacheCntlrAt (
              core_id_t core_id,
              MemComponent::component_t mem_component,
              CacheCntlr * cache_cntlr )  [inline]
```

Definition at line 83 of file memory_manager.h.

References m_all_cache_cntlrs.

Referenced by MemoryManager().

6.191.4 Member Data Documentation

6.191.4.1 m_all_cache_cntlrs

```
std::map< CoreComponentType, CacheCntlr * > ParametricDramDirectoryMSI::MemoryManager::m_↩
all_cache_cntlrs  [static], [private]
```

Definition at line 61 of file memory_manager.h.

Referenced by getCacheCntlrAt(), and setCacheCntlrAt().

6.191.4.2 m_cache_block_size

```
UInt32 ParametricDramDirectoryMSI::MemoryManager::m_cache_block_size  [private]
```

Definition at line 51 of file memory_manager.h.

Referenced by getCacheBlockSize(), and MemoryManager().

6.191.4.3 m_cache_cntlrs

CacheCntlr* ParametricDramDirectoryMSI::MemoryManager::m_cache_cntlrs[**MemComponent::LAST_LEVEL_CACHE**+1] [private]

Definition at line 32 of file memory_manager.h.

Referenced by addL1Hits(), coreInitiateMemoryAccess(), disableModels(), enableModels(), getCache(), handleMsgFromNetwork(), MemoryManager(), and ~MemoryManager().

6.191.4.4 m_cache_perf_models

CachePerfModel* ParametricDramDirectoryMSI::MemoryManager::m_cache_perf_models[**MemComponent::LAST_LEVEL_CACHE**+1] [private]

Definition at line 58 of file memory_manager.h.

Referenced by disableModels(), enableModels(), getCost(), getL1HitLatency(), MemoryManager(), and ~MemoryManager().

6.191.4.5 m_core_id_master

core_id_t ParametricDramDirectoryMSI::MemoryManager::m_core_id_master [private]

Definition at line 43 of file memory_manager.h.

Referenced by broadcastMsg(), MemoryManager(), and sendMsg().

6.191.4.6 m_dram_cache

DramCache* ParametricDramDirectoryMSI::MemoryManager::m_dram_cache [private]

Definition at line 34 of file memory_manager.h.

Referenced by handleMsgFromNetwork(), MemoryManager(), and ~MemoryManager().

6.191.4.7 m_dram_cntlr

PrL1PrL2DramDirectoryMSI::DramCntlr* ParametricDramDirectoryMSI::MemoryManager::m_dram_cntlr [private]

Definition at line 36 of file memory_manager.h.

Referenced by disableModels(), enableModels(), getDramCntlr(), handleMsgFromNetwork(), MemoryManager(), and ~MemoryManager().

6.191.4.8 m_dram_cntlr_present

```
bool ParametricDramDirectoryMSI::MemoryManager::m_dram_cntlr_present  [private]
```

Definition at line 46 of file memory_manager.h.

Referenced by disableModels(), enableModels(), handleMsgFromNetwork(), and MemoryManager().

6.191.4.9 m_dram_controller_home_lookup

```
AddressHomeLookup* ParametricDramDirectoryMSI::MemoryManager::m_dram_controller_home_lookup
[private]
```

Definition at line 38 of file memory_manager.h.

Referenced by getDramControllerHomeLookup(), MemoryManager(), and ~MemoryManager().

6.191.4.10 m_dram_directory_cntlr

```
PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr* ParametricDramDirectoryMSI::MemoryManager::m_↩
dram_directory_cntlr  [private]
```

Definition at line 35 of file memory_manager.h.

Referenced by getDramDirectoryCache(), handleMsgFromNetwork(), MemoryManager(), and ~MemoryManager().

6.191.4.11 m_dtlb

```
TLB * ParametricDramDirectoryMSI::MemoryManager::m_dtlb  [private]
```

Definition at line 39 of file memory_manager.h.

Referenced by coreInitiateMemoryAccess(), MemoryManager(), and ~MemoryManager().

6.191.4.12 m_dummy_shmem_perf

```
ShmemPerf ParametricDramDirectoryMSI::MemoryManager::m_dummy_shmem_perf  [private]
```

Definition at line 55 of file memory_manager.h.

Referenced by handleMsgFromNetwork().

6.191.4.13 m_enabled

```
bool ParametricDramDirectoryMSI::MemoryManager::m_enabled  [private]
```

Definition at line 53 of file memory_manager.h.

Referenced by broadcastMsg(), disableModels(), enableModels(), handleMsgFromNetwork(), and sendMsg().

6.191.4.14 m_itlb

```
TLB* ParametricDramDirectoryMSI::MemoryManager::m_itlb  [private]
```

Definition at line 39 of file memory_manager.h.

Referenced by coreInitiateMemoryAccess(), MemoryManager(), and ~MemoryManager().

6.191.4.15 m_last_level_cache

```
MemComponent::component_t ParametricDramDirectoryMSI::MemoryManager::m_last_level_cache  [private]
```

Definition at line 52 of file memory_manager.h.

Referenced by coreInitiateMemoryAccess(), disableModels(), enableModels(), getCache(), handleMsgFrom↩Network(), MemoryManager(), and ~MemoryManager().

6.191.4.16 m_network_thread_sem

```
Semaphore* ParametricDramDirectoryMSI::MemoryManager::m_network_thread_sem  [private]
```

Definition at line 49 of file memory_manager.h.

Referenced by MemoryManager(), and ~MemoryManager().

6.191.4.17 m_nuca_cache

```
NucaCache* ParametricDramDirectoryMSI::MemoryManager::m_nuca_cache  [private]
```

Definition at line 33 of file memory_manager.h.

Referenced by MemoryManager(), and ~MemoryManager().

6.191.4.18 m_stlb

```
TLB * ParametricDramDirectoryMSI::MemoryManager::m_stlb  [private]
```

Definition at line 39 of file memory_manager.h.

Referenced by MemoryManager(), and ~MemoryManager().

6.191.4.19 m_tag_directory_home_lookup

```
AddressHomeLookup* ParametricDramDirectoryMSI::MemoryManager::m_tag_directory_home_lookup
[private]
```

Definition at line 37 of file memory_manager.h.

Referenced by getTagDirectoryHomeLookup(), MemoryManager(), and ~MemoryManager().

6.191.4.20 m_tag_directory_present

```
bool ParametricDramDirectoryMSI::MemoryManager::m_tag_directory_present  [private]
```

Definition at line 45 of file memory_manager.h.

Referenced by handleMsgFromNetwork(), and MemoryManager().

6.191.4.21 m_tlb_miss_parallel

```
bool ParametricDramDirectoryMSI::MemoryManager::m_tlb_miss_parallel  [private]
```

Definition at line 41 of file memory_manager.h.

Referenced by accessTLB(), and MemoryManager().

6.191.4.22 m_tlb_miss_penalty

```
ComponentLatency ParametricDramDirectoryMSI::MemoryManager::m_tlb_miss_penalty  [private]
```

Definition at line 40 of file memory_manager.h.

Referenced by accessTLB(), and MemoryManager().

6.191.4.23 m_user_thread_sem

```
Semaphore* ParametricDramDirectoryMSI::MemoryManager::m_user_thread_sem  [private]
```

Definition at line 48 of file memory_manager.h.

Referenced by MemoryManager(), and ~MemoryManager().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **memory_manager.h**
- common/core/memory_subsystem/parametric_dram_directory_msi/ **memory_manager.cc**

6.192 FastNehalem::MemoryManager Class Reference

```
#include <memory_manager.h>
```

Inheritance diagram for FastNehalem::MemoryManager:

Public Member Functions

- **MemoryManager** (**Core** *core, **Network** *network, **ShmemPerfModel** *shmem_perf_model)
- ~**MemoryManager** ()
- **SubsecondTime** **coreInitiateMemoryAccessFast** (bool use_icache, **Core::mem_op_t** mem_op_type, **IntPtr** address)

Private Attributes

- **CacheBase** * **icache**
- **CacheBase** * **dcache**
- **CacheBase** * **l2cache**

Static Private Attributes

- static **CacheBase** * **l3cache** = NULL
- static **CacheBase** * **dram** = NULL

Additional Inherited Members

6.192.1 Detailed Description

Definition at line 15 of file memory_manager.h.

6.192.2 Constructor & Destructor Documentation

6.192.2.1 MemoryManager()

```
FastNehalem::MemoryManager::MemoryManager (
              Core * core,
              Network * network,
              ShmemPerfModel * shmem_perf_model )
```

Definition at line 12 of file memory_manager.cc.

References dcache, dram, icache, MemComponent::L1_DCACHE, MemComponent::L1_ICACHE, MemComponent::L2_CACHE, l2cache, MemComponent::L3_CACHE, and l3cache.

6.192.2.2 ~MemoryManager()

```
FastNehalem::MemoryManager::~MemoryManager ( )
```

Definition at line 24 of file memory_manager.cc.

References dcache, icache, and l2cache.

6.192.3 Member Function Documentation

6.192.3.1 coreInitiateMemoryAccessFast()

```
SubsecondTime FastNehalem::MemoryManager::coreInitiateMemoryAccessFast (
            bool use_icache,
             Core::mem_op_t mem_op_type,
             IntPtr address )  [inline], [virtual]
```

Implements **MemoryManagerFast** (p. 756).

Definition at line 25 of file memory_manager.h.

References MemoryManagerFast::CACHE_LINE_BITS, dcache, and icache.

6.192.4 Member Data Documentation

6.192.4.1 dcache

```
CacheBase * FastNehalem::MemoryManager::dcache  [private]
```

Definition at line 18 of file memory_manager.h.

Referenced by coreInitiateMemoryAccessFast(), MemoryManager(), and ~MemoryManager().

6.192.4.2 dram

```
CacheBase * FastNehalem::MemoryManager::dram = NULL  [static], [private]
```

Definition at line 19 of file memory_manager.h.

Referenced by MemoryManager().

6.192.4.3 icache

```
CacheBase* FastNehalem::MemoryManager::icache  [private]
```

Definition at line 18 of file memory_manager.h.

Referenced by coreInitiateMemoryAccessFast(), MemoryManager(), and ~MemoryManager().

6.192.4.4 l2cache

```
CacheBase * FastNehalem::MemoryManager::l2cache  [private]
```

Definition at line 18 of file memory_manager.h.

Referenced by MemoryManager(), and ~MemoryManager().

6.192.4.5 l3cache

```
CacheBase * FastNehalem::MemoryManager::l3cache = NULL  [static], [private]
```

Definition at line 19 of file memory_manager.h.

Referenced by MemoryManager().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/fast_nehalem/ **memory_manager.h**
- common/core/memory_subsystem/fast_nehalem/ **memory_manager.cc**

6.193 MemoryManagerBase Class Reference

```
#include <memory_manager_base.h>
```

Inheritance diagram for MemoryManagerBase:

Public Types

- enum **CachingProtocol_t** { **PARAMETRIC_DRAM_DIRECTORY_MSI**, **FAST_NEHALEM**, **NUM_CAC↩ HING_PROTOCOL_TYPES** }

Public Member Functions

- **MemoryManagerBase** (**Core** ∗core, **Network** ∗network, **ShmemPerfModel** ∗shmem_perf_model)
- virtual ∼**MemoryManagerBase** ()
- virtual **HitWhere::where_t** **coreInitiateMemoryAccess** (**MemComponent::component_t** mem_↩ component, **Core::lock_signal_t** lock_signal, **Core::mem_op_t** mem_op_type, **IntPtr** address, **UInt32** offset, **Byte** ∗data_buf, **UInt32** data_length, **Core::MemModeled** modeled)=0
- virtual **SubsecondTime** **coreInitiateMemoryAccessFast** (bool icache, **Core::mem_op_t** mem_op_type, **IntPtr** address)
- virtual void **handleMsgFromNetwork** (**NetPacket** &packet)=0
- virtual **UInt64** **getCacheBlockSize** () const =0
- virtual **SubsecondTime** **getL1HitLatency** (void)=0
- virtual void **addL1Hits** (bool icache, **Core::mem_op_t** mem_op_type, **UInt64** hits)=0
- virtual **core_id_t** **getShmemRequester** (const void ∗pkt_data)=0
- virtual void **enableModels** ()=0
- virtual void **disableModels** ()=0
- virtual **UInt32** **getModeledLength** (const void ∗pkt_data)=0
- **Core** ∗ **getCore** ()

- virtual void **sendMsg** (**PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t** msg_type, **MemComponent::component_t** sender_mem_component, **MemComponent::component_t** receiver_mem_component, **core_id_t** requester, **core_id_t** receiver, **IntPtr** address, **Byte** *data_buf=NULL, **UInt32** data_length=0, **HitWhere::where_t** where= **HitWhere::UNKNOWN**, **ShmemPerf** *perf=NULL, **ShmemPerfModel::Thread_t** thread_num= **ShmemPerfModel::NUM_CORE_THREADS**)=0
- virtual void **broadcastMsg** (**PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t** msg_type, **MemComponent::component_t** sender_mem_component, **MemComponent::component_t** receiver_mem_component, **core_id_t** requester, **IntPtr** address, **Byte** *data_buf=NULL, **UInt32** data_length=0, **ShmemPerf** *perf=NULL, **ShmemPerfModel::Thread_t** thread_num= **ShmemPerfModel::NUM_CORE_THREADS**)=0

Static Public Member Functions

- static **CachingProtocol_t** **parseProtocolType** (String &protocol_type)
- static **MemoryManagerBase** * **createMMU** (String protocol_type, **Core** *core, **Network** *network, **ShmemPerfModel** *shmem_perf_model)

Protected Member Functions

- **Network** * **getNetwork** ()
- **ShmemPerfModel** * **getShmemPerfModel** ()
- std::vector< **core_id_t** > **getCoreListWithMemoryControllers** (void)
- void **printCoreListWithMemoryControllers** (std::vector< **core_id_t** > &core_list_with_memory_controllers)

Private Member Functions

- void **parseMemoryControllerList** (String &memory_controller_positions, std::vector< **core_id_t** > &core_list_from_cfg_file, **SInt32** application_core_count)

Private Attributes

- **Core** * **m_core**
- **Network** * **m_network**
- **ShmemPerfModel** * **m_shmem_perf_model**

6.193.1 Detailed Description

Definition at line 13 of file memory_manager_base.h.

6.193.2 Member Enumeration Documentation

6.193.2.1 CachingProtocol_t

enum **MemoryManagerBase::CachingProtocol_t**

Enumerator

PARAMETRIC_DRAM_DIRECTORY_MSI	
FAST_NEHALEM	
NUM_CACHING_PROTOCOL_TYPES	

Definition at line 16 of file memory_manager_base.h.

6.193.3 Constructor & Destructor Documentation

6.193.3.1 MemoryManagerBase()

```
MemoryManagerBase::MemoryManagerBase (
              Core * core,
              Network * network,
              ShmemPerfModel * shmem_perf_model )  [inline]
```

Definition at line 38 of file memory_manager_base.h.

6.193.3.2 ~MemoryManagerBase()

```
virtual MemoryManagerBase::~MemoryManagerBase ( )  [inline], [virtual]
```

Definition at line 43 of file memory_manager_base.h.

6.193.4 Member Function Documentation

6.193.4.1 addL1Hits()

```
virtual void MemoryManagerBase::addL1Hits (
             bool icache,
              Core::mem_op_t mem_op_type,
              UInt64 hits )  [pure virtual]
```

Implemented in **ParametricDramDirectoryMSI::MemoryManager** (p. 730), and **MemoryManagerFast** (p. 755).

Referenced by Core::logMemoryHit().

6.193.4.2 broadcastMsg()

```
virtual void MemoryManagerBase::broadcastMsg (
              PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t msg_type,
              MemComponent::component_t sender_mem_component,
              MemComponent::component_t receiver_mem_component,
              core_id_t requester,
              IntPtr address,
              Byte * data_buf = NULL,
              UInt32 data_length = 0,
              ShmemPerf * perf = NULL,
              ShmemPerfModel::Thread_t thread_num = ShmemPerfModel::NUM_CORE_THREADS ) [pure
virtual]
```

Implemented in **ParametricDramDirectoryMSI::MemoryManager** (p. 731), and **MemoryManagerFast** (p. 755).

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2Dram↩ DirectoryMSI::DramDirectoryCntlr::processNullifyReq(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr↩ ::processUpgradeReqFromL2Cache().

6.193.4.3 coreInitiateMemoryAccess()

```
virtual HitWhere::where_t MemoryManagerBase::coreInitiateMemoryAccess (
              MemComponent::component_t mem_component,
              Core::lock_signal_t lock_signal,
              Core::mem_op_t mem_op_type,
              IntPtr address,
              UInt32 offset,
              Byte * data_buf,
              UInt32 data_length,
              Core::MemModeled modeled ) [pure virtual]
```

Implemented in **ParametricDramDirectoryMSI::MemoryManager** (p. 731), and **MemoryManagerFast** (p. 755).

Referenced by coreInitiateMemoryAccessFast(), and Core::initiateMemoryAccess().

6.193.4.4 coreInitiateMemoryAccessFast()

```
virtual SubsecondTime MemoryManagerBase::coreInitiateMemoryAccessFast (
            bool icache,
              Core::mem_op_t mem_op_type,
              IntPtr address ) [inline], [virtual]
```

Reimplemented in **FastNehalem::MemoryManager** (p. 743), and **MemoryManagerFast** (p. 756).

Definition at line 52 of file memory_manager_base.h.

References ShmemPerfModel::_USER_THREAD, coreInitiateMemoryAccess(), getCacheBlockSize(), getCore(), ShmemPerfModel::getElapsedTime(), PerformanceModel::getElapsedTime(), Core::getPerformanceModel(), get↩ ShmemPerfModel(), MemComponent::L1_DCACHE, MemComponent::L1_ICACHE, Core::MEM_MODELED_C↩ OUNT_TLBTIME, Core::NONE, and ShmemPerfModel::setElapsedTime().

Referenced by Core::accessMemoryFast().

6.193.4.5 createMMU()

```
MemoryManagerBase * MemoryManagerBase::createMMU (
            String protocol_type,
             Core * core,
             Network * network,
             ShmemPerfModel * shmem_perf_model )  [static]
```

Definition at line 10 of file memory_manager_base.cc.

References FAST_NEHALEM, LOG_PRINT_ERROR, PARAMETRIC_DRAM_DIRECTORY_MSI, and parse↩ ProtocolType().

Referenced by Core::Core().

6.193.4.6 disableModels()

```
virtual void MemoryManagerBase::disableModels ( )  [pure virtual]
```

Implemented in **ParametricDramDirectoryMSI::MemoryManager** (p. 731), and **MemoryManagerFast** (p. 756).

Referenced by Core::disablePerformanceModels().

6.193.4.7 enableModels()

```
virtual void MemoryManagerBase::enableModels ( )  [pure virtual]
```

Implemented in **ParametricDramDirectoryMSI::MemoryManager** (p. 732), and **MemoryManagerFast** (p. 756).

Referenced by Core::enablePerformanceModels().

6.193.4.8 getCacheBlockSize()

```
virtual  UInt64 MemoryManagerBase::getCacheBlockSize ( ) const  [pure virtual]
```

Implemented in **ParametricDramDirectoryMSI::MemoryManager** (p. 732), and **MemoryManagerFast** (p. 756).

Referenced by coreInitiateMemoryAccessFast(), Core::initiateMemoryAccess(), and Core::readInstruction↩ Memory().

6.193.4.9 getCore()

```
Core* MemoryManagerBase::getCore ( )  [inline]
```

Definition at line 91 of file memory_manager_base.h.

References m_core.

Referenced by ParametricDramDirectoryMSI::MemoryManager::accessTLB(), ParametricDramDirectoryMSI::↩ MemoryManager::broadcastMsg(), coreInitiateMemoryAccessFast(), PrL1PrL2DramDirectoryMSI::DramCntlr::↩ DramCntlr(), ParametricDramDirectoryMSI::MemoryManager::MemoryManager(), PrL1PrL2DramDirectoryMSI::↩ DramCntlr::printDramAccessCount(), and ParametricDramDirectoryMSI::MemoryManager::sendMsg().

6.193.4.10 getCoreListWithMemoryControllers()

```
std::vector< core_id_t > MemoryManagerBase::getCoreListWithMemoryControllers (
            void ) [protected]
```

Definition at line 58 of file memory_manager_base.cc.

References cfg, NetworkModel::computeMemoryControllerPositions(), Config::getApplicationCores(), Config::get↩ Singleton(), config::Config::getString(), LOG_ASSERT_ERROR, LOG_PRINT_ERROR, parseMemoryController↩ List(), and NetworkModel::parseNetworkType().

Referenced by ParametricDramDirectoryMSI::MemoryManager::MemoryManager().

6.193.4.11 getL1HitLatency()

```
virtual SubsecondTime MemoryManagerBase::getL1HitLatency (
            void ) [pure virtual]
```

Implemented in **ParametricDramDirectoryMSI::MemoryManager** (p. 734), and **MemoryManagerFast** (p. 757).

6.193.4.12 getModeledLength()

```
virtual UInt32 MemoryManagerBase::getModeledLength (
            const void * pkt_data ) [pure virtual]
```

Implemented in **ParametricDramDirectoryMSI::MemoryManager** (p. 734), and **MemoryManagerFast** (p. 757).

Referenced by Network::getModeledLength().

6.193.4.13 getNetwork()

```
Network* MemoryManagerBase::getNetwork ( )  [inline], [protected]
```

Definition at line 31 of file memory_manager_base.h.

References m_network.

Referenced by ParametricDramDirectoryMSI::MemoryManager::broadcastMsg(), ParametricDramDirectory↩MSI::MemoryManager::MemoryManager(), ParametricDramDirectoryMSI::MemoryManager::sendMsg(), and ParametricDramDirectoryMSI::MemoryManager::∼MemoryManager().

6.193.4.14 getShmemPerfModel()

```
ShmemPerfModel* MemoryManagerBase::getShmemPerfModel ( )  [inline], [protected]
```

Definition at line 32 of file memory_manager_base.h.

References m_shmem_perf_model.

Referenced by ParametricDramDirectoryMSI::MemoryManager::accessTLB(), ParametricDramDirectoryMSI↩::MemoryManager::broadcastMsg(), MemoryManagerFast::coreInitiateMemoryAccess(), coreInitiateMemory↩AccessFast(), ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork(), ParametricDram↩DirectoryMSI::MemoryManager::incrElapsedTime(), ParametricDramDirectoryMSI::MemoryManager::Memory↩Manager(), and ParametricDramDirectoryMSI::MemoryManager::sendMsg().

6.193.4.15 getShmemRequester()

```
virtual core_id_t MemoryManagerBase::getShmemRequester (
            const void * pkt_data )  [pure virtual]
```

Implemented in **ParametricDramDirectoryMSI::MemoryManager** (p. 735), and **MemoryManagerFast** (p. 757).

Referenced by NetworkModelBus::accountPacket(), NetworkModelEMeshHopCounter::processReceivedPacket(), NetworkModelMagic::processReceivedPacket(), NetworkModelEMeshHopByHop::processReceivedPacket(), and NetworkModelEMeshHopByHop::routePacket().

6.193.4.16 handleMsgFromNetwork()

```
virtual void MemoryManagerBase::handleMsgFromNetwork (
             NetPacket & packet )  [pure virtual]
```

Implemented in **ParametricDramDirectoryMSI::MemoryManager** (p. 735), and **MemoryManagerFast** (p. 757).

Referenced by MemoryManagerNetworkCallback().

6.193.4.17 parseMemoryControllerList()

```
void MemoryManagerBase::parseMemoryControllerList (
            String & memory_controller_positions,
            std::vector< core_id_t > & core_list_from_cfg_file,
             SInt32 application_core_count ) [private]
```

Definition at line 138 of file memory_manager_base.cc.

References LOG_ASSERT_ERROR.

Referenced by getCoreListWithMemoryControllers().

6.193.4.18 parseProtocolType()

```
MemoryManagerBase::CachingProtocol_t MemoryManagerBase::parseProtocolType (
            String & protocol_type ) [static]
```

Definition at line 30 of file memory_manager_base.cc.

References FAST_NEHALEM, NUM_CACHING_PROTOCOL_TYPES, and PARAMETRIC_DRAM_DIRECTOR↩Y_MSI.

Referenced by createMMU().

6.193.4.19 printCoreListWithMemoryControllers()

```
void MemoryManagerBase::printCoreListWithMemoryControllers (
            std::vector< core_id_t > & core_list_with_memory_controllers ) [protected]
```

Definition at line 173 of file memory_manager_base.cc.

6.193.4.20 sendMsg()

```
virtual void MemoryManagerBase::sendMsg (
             PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t msg_type,
             MemComponent::component_t sender_mem_component,
             MemComponent::component_t receiver_mem_component,
             core_id_t requester,
             core_id_t receiver,
             IntPtr address,
             Byte * data_buf = NULL,
             UInt32 data_length = 0,
             HitWhere::where_t where = HitWhere::UNKNOWN,
             ShmemPerf * perf = NULL,
             ShmemPerfModel::Thread_t thread_num = ShmemPerfModel::NUM_CORE_THREADS ) [pure
virtual]
```

Implemented in **ParametricDramDirectoryMSI::MemoryManager** (p. 736), and **MemoryManagerFast** (p. 758).

Referenced by DramCntlrInterface::handleMsgFromTagDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectory↩Cntlr::processDRAMReply(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processNullifyReq(), PrL1PrL2DramDirectoryMSI::Dram↩DirectoryCntlr::processShReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::process↩UpgradeReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSendToL2↩Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::sendDataToDram(), and PrL1PrL2DramDirectory↩MSI::DramDirectoryCntlr::sendDataToNUCA().

6.193.5 Member Data Documentation

6.193.5.1 m_core

```
Core* MemoryManagerBase::m_core  [private]
```

Definition at line 24 of file memory_manager_base.h.

Referenced by getCore().

6.193.5.2 m_network

```
Network* MemoryManagerBase::m_network  [private]
```

Definition at line 25 of file memory_manager_base.h.

Referenced by getNetwork().

6.193.5.3 m_shmem_perf_model

```
ShmemPerfModel* MemoryManagerBase::m_shmem_perf_model  [private]
```

Definition at line 26 of file memory_manager_base.h.

Referenced by getShmemPerfModel().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/ **memory_manager_base.h**
- common/core/memory_subsystem/ **memory_manager_base.cc**

6.194 MemoryManagerFast Class Reference

```
#include <memory_manager_fast.h>
```

Inheritance diagram for MemoryManagerFast:

Public Member Functions

- **MemoryManagerFast** (**Core** ∗core, **Network** ∗network, **ShmemPerfModel** ∗shmem_perf_model)
- virtual ∼**MemoryManagerFast** ()
- **HitWhere::where_t coreInitiateMemoryAccess** (**MemComponent::component_t** mem_component, **Core::lock_signal_t** lock_signal, **Core::mem_op_t** mem_op_type, **IntPtr** address, **UInt32** offset, **Byte** ∗data_buf, **UInt32** data_length, **Core::MemModeled** modeled)
- virtual **SubsecondTime coreInitiateMemoryAccessFast** (bool icache, **Core::mem_op_t** mem_op_type, **IntPtr** address)=0
- void **handleMsgFromNetwork** (**NetPacket** &packet)
- void **enableModels** ()
- void **disableModels** ()
- **core_id_t getShmemRequester** (const void ∗pkt_data)
- **UInt32 getModeledLength** (const void ∗pkt_data)
- **UInt64 getCacheBlockSize** () const
- void **sendMsg** (**PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t** msg_type, **MemComponent::component_t** sender_mem_component, **MemComponent::component_t** receiver_mem_component, **core_id_t** requester, **core_id_t** receiver, **IntPtr** address, **Byte** ∗data_buf=NULL, **UInt32** data_length=0, **HitWhere::where_t** where= **HitWhere::UNKNOWN**, **ShmemPerf** ∗perf=NULL, **ShmemPerfModel::Thread_t** thread_num= **ShmemPerfModel::NUM_CORE_THREADS**)
- void **broadcastMsg** (**PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t** msg_type, **MemComponent::component_t** sender_mem_component, **MemComponent::component_t** receiver_mem_component, **core_id_t** requester, **IntPtr** address, **Byte** ∗data_buf=NULL, **UInt32** data_length=0, **ShmemPerf** ∗perf=NULL, **ShmemPerfModel::Thread_t** thread_num= **ShmemPerfModel::NUM_CORE_THREADS**)
- **SubsecondTime getL1HitLatency** (void)
- void **addL1Hits** (bool icache, **Core::mem_op_t** mem_op_type, **UInt64** hits)

Static Protected Attributes

- static const **UInt64 CACHE_LINE_BITS** = 6
- static const **UInt64 CACHE_LINE_SIZE** = 1 << **CACHE_LINE_BITS**

Additional Inherited Members

6.194.1 Detailed Description

Definition at line 9 of file memory_manager_fast.h.

6.194.2 Constructor & Destructor Documentation

6.194.2.1 MemoryManagerFast()

```
MemoryManagerFast::MemoryManagerFast (
              Core * core,
              Network * network,
              ShmemPerfModel * shmem_perf_model )  [inline]
```

Definition at line 16 of file memory_manager_fast.h.

6.194.2.2 ~MemoryManagerFast()

```
virtual MemoryManagerFast::~MemoryManagerFast ( )  [inline], [virtual]
```

Definition at line 19 of file memory_manager_fast.h.

6.194.3 Member Function Documentation

6.194.3.1 addL1Hits()

```
void MemoryManagerFast::addL1Hits (
             bool icache,
              Core::mem_op_t mem_op_type,
              UInt64 hits )  [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 747).

Definition at line 60 of file memory_manager_fast.h.

6.194.3.2 broadcastMsg()

```
void MemoryManagerFast::broadcastMsg (
              PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t msg_type,
              MemComponent::component_t sender_mem_component,
              MemComponent::component_t receiver_mem_component,
              core_id_t requester,
              IntPtr address,
              Byte * data_buf = NULL,
              UInt32 data_length = 0,
              ShmemPerf * perf = NULL,
              ShmemPerfModel::Thread_t thread_num =  ShmemPerfModel::NUM_CORE_THREADS )  [inline],
[virtual]
```

Implements **MemoryManagerBase** (p. 747).

Definition at line 57 of file memory_manager_fast.h.

6.194.3.3 coreInitiateMemoryAccess()

```
HitWhere::where_t MemoryManagerFast::coreInitiateMemoryAccess (
            MemComponent::component_t mem_component,
            Core::lock_signal_t lock_signal,
            Core::mem_op_t mem_op_type,
            IntPtr address,
            UInt32 offset,
            Byte * data_buf,
            UInt32 data_length,
            Core::MemModeled modeled ) [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 748).

Definition at line 21 of file memory_manager_fast.h.

References ShmemPerfModel::_USER_THREAD, coreInitiateMemoryAccessFast(), MemoryManagerBase::get↩ ShmemPerfModel(), ShmemPerfModel::incrElapsedTime(), MemComponent::L1_ICACHE, HitWhere::MISS, and SubsecondTime::Zero().

6.194.3.4 coreInitiateMemoryAccessFast()

```
virtual SubsecondTime MemoryManagerFast::coreInitiateMemoryAccessFast (
            bool icache,
            Core::mem_op_t mem_op_type,
            IntPtr address ) [pure virtual]
```

Reimplemented from **MemoryManagerBase** (p. 748).

Implemented in **FastNehalem::MemoryManager** (p. 743).

Referenced by coreInitiateMemoryAccess().

6.194.3.5 disableModels()

```
void MemoryManagerFast::disableModels ( ) [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 749).

Definition at line 47 of file memory_manager_fast.h.

6.194.3.6 enableModels()

```
void MemoryManagerFast::enableModels ( ) [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 749).

Definition at line 46 of file memory_manager_fast.h.

6.194.3.7 getCacheBlockSize()

```
UInt64 MemoryManagerFast::getCacheBlockSize ( ) const  [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 749).

Definition at line 53 of file memory_manager_fast.h.

References CACHE_LINE_SIZE.

6.194.3.8 getL1HitLatency()

```
SubsecondTime MemoryManagerFast::getL1HitLatency (
            void  )  [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 750).

Definition at line 59 of file memory_manager_fast.h.

References SubsecondTime::Zero().

6.194.3.9 getModeledLength()

```
UInt32 MemoryManagerFast::getModeledLength (
            const void * pkt_data )  [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 750).

Definition at line 50 of file memory_manager_fast.h.

6.194.3.10 getShmemRequester()

```
core_id_t MemoryManagerFast::getShmemRequester (
            const void * pkt_data )  [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 751).

Definition at line 49 of file memory_manager_fast.h.

6.194.3.11 handleMsgFromNetwork()

```
void MemoryManagerFast::handleMsgFromNetwork (
              NetPacket & packet )  [inline], [virtual]
```

Implements **MemoryManagerBase** (p. 751).

Definition at line 44 of file memory_manager_fast.h.

6.194.3.12 sendMsg()

```
void MemoryManagerFast::sendMsg (
              PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t msg_type,
              MemComponent::component_t sender_mem_component,
              MemComponent::component_t receiver_mem_component,
              core_id_t requester,
              core_id_t receiver,
              IntPtr address,
              Byte * data_buf = NULL,
              UInt32 data_length = 0,
              HitWhere::where_t where =  HitWhere::UNKNOWN,
              ShmemPerf * perf = NULL,
              ShmemPerfModel::Thread_t thread_num =  ShmemPerfModel::NUM_CORE_THREADS )  [inline],
[virtual]
```

Implements **MemoryManagerBase** (p. 752).

Definition at line 56 of file memory_manager_fast.h.

6.194.4 Member Data Documentation

6.194.4.1 CACHE_LINE_BITS

```
const  UInt64 MemoryManagerFast::CACHE_LINE_BITS = 6  [static], [protected]
```

Definition at line 12 of file memory_manager_fast.h.

Referenced by FastNehalem::MemoryManager::coreInitiateMemoryAccessFast().

6.194.4.2 CACHE_LINE_SIZE

```
const  UInt64 MemoryManagerFast::CACHE_LINE_SIZE = 1 <<  CACHE_LINE_BITS  [static], [protected]
```

Definition at line 13 of file memory_manager_fast.h.

Referenced by getCacheBlockSize().

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/ **memory_manager_fast.h**

6.195 MemoryResult Struct Reference

`#include <core.h>`

Public Attributes

- **HitWhere::where_t hit_where**
- **subsecond_time_t latency**

6.195.1 Detailed Description

Definition at line 24 of file core.h.

6.195.2 Member Data Documentation

6.195.2.1 hit_where

HitWhere::where_t MemoryResult::hit_where

Definition at line 25 of file core.h.

Referenced by DynamicInstruction::accessMemory(), MicroOpPerformanceModel::handleInstruction(), lite::handleMemoryReadDetailedIssue(), lite::handleMemoryReadFaultinjection(), lite::handleMemoryWriteDetailedIssue(), lite::handleMemoryWriteFaultinjection(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), IntervalTimer::issueMemOp(), and makeMemoryResult().

6.195.2.2 latency

subsecond_time_t MemoryResult::latency

Definition at line 26 of file core.h.

Referenced by DynamicInstruction::accessMemory(), PthreadEmu::BarrierWait(), PthreadEmu::CondBroadcast(), PthreadEmu::CondSignal(), PthreadEmu::CondWait(), MicroOpPerformanceModel::handleInstruction(), lite::handleMemoryReadDetailedIssue(), lite::handleMemoryReadFaultinjection(), lite::handleMemoryWriteDetailedIssue(), lite::handleMemoryWriteFaultinjection(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), IntervalTimer::issueMemOp(), makeMemoryResult(), PthreadEmu::MutexLock(), PthreadEmu::MutexTrylock(), and PthreadEmu::MutexUnlock().

The documentation for this struct was generated from the following file:

- common/core/ **core.h**

6.196 MemoryTracker Class Reference

`#include <memory_tracker.h>`

Classes

- struct **Allocation**
- struct **AllocationSite**
- class **RoutineTracer**
- class **RoutineTracerThread**

Public Member Functions

- **MemoryTracker** ()
- ~**MemoryTracker** ()
- void **logMalloc** (**thread_id_t** thread_id, **UInt64** eip, **UInt64** address, **UInt64** size)
- void **logFree** (**thread_id_t** thread_id, **UInt64** eip, **UInt64** address)

Private Types

- typedef std::unordered_map< **CallStack**, **AllocationSite** * > **AllocationSites**
- typedef std::map< **UInt64**, **Allocation** > **Allocations**

Private Member Functions

- **UInt64** **ce_get_owner** (**core_id_t** core_id, **UInt64** address)
- void **ce_notify_access** (**UInt64** owner, **Core::mem_op_t** mem_op_type, **HitWhere::where_t** hit_where)
- void **ce_notify_evict** (bool on_roi_end, **UInt64** owner, **UInt64** evictor, **CacheBlockInfo::BitsUsedType** bits_used, **UInt32** bits_total)

Static Private Member Functions

- static **UInt64** **__ce_get_owner** (**UInt64** user, **core_id_t** core_id, **UInt64** address)
- static void **__ce_notify_access** (**UInt64** user, **UInt64** owner, **Core::mem_op_t** mem_op_type, **HitWhere::where_t** hit_where)
- static void **__ce_notify_evict** (**UInt64** user, bool on_roi_end, **UInt64** owner, **UInt64** evictor, **CacheBlockInfo::BitsUsedType** bits_used, **UInt32** bits_total)

Private Attributes

- **Lock** **m_lock**
- **Allocations** **m_allocations**
- **AllocationSites** **m_allocation_sites**

6.196.1 Detailed Description

Definition at line 16 of file memory_tracker.h.

6.196.2 Member Typedef Documentation

6.196.2.1 Allocations

```
typedef std::map< UInt64, Allocation> MemoryTracker::Allocations [private]
```

Definition at line 80 of file memory_tracker.h.

6.196.2.2 AllocationSites

```
typedef std::unordered_map< CallStack, AllocationSite* > MemoryTracker::AllocationSites [private]
```

Definition at line 71 of file memory_tracker.h.

6.196.3 Constructor & Destructor Documentation

6.196.3.1 MemoryTracker()

```
MemoryTracker::MemoryTracker ( )
```

Definition at line 8 of file memory_tracker.cc.

References __ce_get_owner(), __ce_notify_access(), and __ce_notify_evict().

Referenced by MemoryTracker::RoutineTracer::RoutineTracer().

6.196.3.2 ~MemoryTracker()

```
MemoryTracker::~MemoryTracker ( )
```

Definition at line 13 of file memory_tracker.cc.

References HitWhereIsValid(), HitWhereString(), m_allocation_sites, RoutineTracer::Routine::m_location, RoutineTracer::Routine::m_name, HitWhere::NUM_HITWHERES, MemoryTracker::AllocationSite::total_loads, MemoryTracker::AllocationSite::total_stores, and HitWhere::WHERE_FIRST.

6.196.4 Member Function Documentation

6.196.4.1 __ce_get_owner()

```
static  UInt64 MemoryTracker::__ce_get_owner (
              UInt64 user,
              core_id_t core_id,
              UInt64 address )  [inline], [static], [private]
```

Definition at line 94 of file memory_tracker.h.

Referenced by MemoryTracker().

6.196.4.2 __ce_notify_access()

```
static void MemoryTracker::__ce_notify_access (
              UInt64 user,
              UInt64 owner,
              Core::mem_op_t mem_op_type,
              HitWhere::where_t hit_where )  [inline], [static], [private]
```

Definition at line 96 of file memory_tracker.h.

Referenced by MemoryTracker().

6.196.4.3 __ce_notify_evict()

```
static void MemoryTracker::__ce_notify_evict (
              UInt64 user,
             bool on_roi_end,
              UInt64 owner,
              UInt64 evictor,
              CacheBlockInfo::BitsUsedType bits_used,
              UInt32 bits_total )  [inline], [static], [private]
```

Definition at line 98 of file memory_tracker.h.

Referenced by MemoryTracker().

6.196.4.4 ce_get_owner()

```
 UInt64 MemoryTracker::ce_get_owner (
              core_id_t core_id,
              UInt64 address )  [private]
```

Definition at line 127 of file memory_tracker.cc.

References LOG_ASSERT_WARNING, m_allocations, and m_lock.

6.196.4.5 ce_notify_access()

```
void MemoryTracker::ce_notify_access (
              UInt64 owner,
              Core::mem_op_t mem_op_type,
              HitWhere::where_t hit_where )  [private]
```

Definition at line 147 of file memory_tracker.cc.

References MemoryTracker::AllocationSite::hit_where_load, MemoryTracker::AllocationSite::hit_where_store, MemoryTracker::AllocationSite::total_loads, MemoryTracker::AllocationSite::total_stores, and Core::WRITE.

6.196.4.6 ce_notify_evict()

```
void MemoryTracker::ce_notify_evict (
             bool on_roi_end,
              UInt64 owner,
              UInt64 evictor,
              CacheBlockInfo::BitsUsedType bits_used,
              UInt32 bits_total )  [private]
```

Definition at line 165 of file memory_tracker.cc.

References MemoryTracker::AllocationSite::evicted_by.

6.196.4.7 logFree()

```
void MemoryTracker::logFree (
              thread_id_t thread_id,
              UInt64 eip,
              UInt64 address )
```

Definition at line 120 of file memory_tracker.cc.

References m_lock.

6.196.4.8 logMalloc()

```
void MemoryTracker::logMalloc (
              thread_id_t thread_id,
              UInt64 eip,
              UInt64 address,
              UInt64 size )
```

Definition at line 66 of file memory_tracker.cc.

References m_allocation_sites, and m_lock.

6.196.5 Member Data Documentation

6.196.5.1 m_allocation_sites

AllocationSites MemoryTracker::m_allocation_sites [private]

Definition at line 84 of file memory_tracker.h.

Referenced by logMalloc(), and ~MemoryTracker().

6.196.5.2 m_allocations

Allocations MemoryTracker::m_allocations [private]

Definition at line 83 of file memory_tracker.h.

Referenced by ce_get_owner().

6.196.5.3 m_lock

Lock MemoryTracker::m_lock [private]

Definition at line 82 of file memory_tracker.h.

Referenced by MemoryTracker::RoutineTracer::addRoutine(), ce_get_owner(), MemoryTracker::RoutineTracer↩ ::hasRoutine(), logFree(), and logMalloc().

The documentation for this class was generated from the following files:

- common/system/ **memory_tracker.h**
- common/system/ **memory_tracker.cc**

6.197 MicroOp Struct Reference

`#include <micro_op.h>`

Public Types

- enum **uop_type_t** { **UOP_INVALID** = 0, **UOP_LOAD**, **UOP_EXECUTE**, **UOP_STORE** }
- enum **uop_subtype_t** {
  UOP_SUBTYPE_FP_ADDSUB, **UOP_SUBTYPE_FP_MULDIV**, **UOP_SUBTYPE_LOAD**, **UOP_SUBT↩ YPE_STORE**,
  UOP_SUBTYPE_GENERIC, **UOP_SUBTYPE_BRANCH**, **UOP_SUBTYPE_SIZE** }

Public Member Functions

- **MicroOp** ()
- void **makeLoad** (uint32_t offset, dl::Decoder::decoder_opcode **instructionOpcode**, const String &instructionOpcodeName, uint16_t mem_size)
- void **makeExecute** (uint32_t offset, uint32_t num_loads, dl::Decoder::decoder_opcode **instructionOpcode**, const String &instructionOpcodeName, bool **isBranch**)
- void **makeStore** (uint32_t offset, uint32_t num_execute, dl::Decoder::decoder_opcode **instructionOpcode**, const String &instructionOpcodeName, uint16_t mem_size)
- void **makeDynamic** (const String &instructionOpcodeName, uint32_t execLatency)
- void **setTypes** ()
- **uop_subtype_t** **getSubtype** (void) const
- bool **isFpLoadStore** () const
- void **setIsX87** (bool _is_x87)
- bool **isX87** (void) const
- void **setOperandSize** (int size)
- uint16_t **getOperandSize** (void) const
- uint16_t **getMemoryAccessSize** (void) const
- void **setInstruction** (**Instruction** *instr)
- **Instruction** * **getInstruction** (void) const
- void **setDecodedInstruction** (const dl::DecodedInst *instr)
- const dl::DecodedInst * **getDecodedInstruction** (void) const
- dl::Decoder::decoder_opcode **getInstructionOpcode** () const
- void **setFirst** (bool **first**)
- bool **isFirst** () const
- void **setLast** (bool **last**)
- bool **isLast** () const
- void **verify** () const
- uint32_t **getSourceRegistersLength** () const
- dl::Decoder::decoder_reg **getSourceRegister** (uint32_t index) const
- void **addSourceRegister** (dl::Decoder::decoder_reg registerId, const String &registerName)
- uint32_t **getAddressRegistersLength** () const
- dl::Decoder::decoder_reg **getAddressRegister** (uint32_t index) const
- void **addAddressRegister** (dl::Decoder::decoder_reg registerId, const String &registerName)
- uint32_t **getDestinationRegistersLength** () const
- dl::Decoder::decoder_reg **getDestinationRegister** (uint32_t index) const
- void **addDestinationRegister** (dl::Decoder::decoder_reg registerId, const String &registerName)
- const **Memory::Access** & **getInstructionPointer** () const
- void **setInstructionPointer** (const **Memory::Access** &ip)
- bool **isBranch** () const
- bool **isInterrupt** () const
- void **setInterrupt** (bool **interrupt**)
- bool **isSerializing** () const
- void **setSerializing** (bool **serializing**)
- bool **isMemBarrier** () const
- void **setMemBarrier** (bool membar)
- bool **isCacheFlush** () const
- bool **isDiv** () const
- bool **isPause** () const
- bool **isLoad** () const
- bool **isStore** () const
- void **setDebugInfo** (String debugInfo)
- String **toString** () const
- String **toShortString** (bool withDisassembly=false) const
- bool **isExecute** () const
- **uop_type_t** **getType** () const

Static Public Member Functions

- static **uop_subtype_t** **getSubtype_Exec** (const **MicroOp** &uop)
- static **uop_subtype_t** **getSubtype** (const **MicroOp** &uop)
- static String **getSubtypeString** (**uop_subtype_t** **uop_subtype**)

Public Attributes

- **uop_type_t** **uop_type**
- **uop_subtype_t** **uop_subtype**
- bool **first**
- bool **last**
- dl::Decoder::decoder_opcode **instructionOpcode**
- const dl::DecodedInst * **decodedInstruction**
- **Instruction** * **instruction**
- uint32_t **microOpTypeOffset**
- uint32_t **intraInstructionDependencies**
- uint32_t **sourceRegistersLength**
- dl::Decoder::decoder_reg **sourceRegisters** [**MAXIMUM_NUMBER_OF_SOURCE_REGISTERS**]
- uint32_t **addressRegistersLength**
- dl::Decoder::decoder_reg **addressRegisters** [**MAXIMUM_NUMBER_OF_ADDRESS_REGISTERS**]
- uint32_t **destinationRegistersLength**
- dl::Decoder::decoder_reg **destinationRegisters** [**MAXIMUM_NUMBER_OF_DESTINATION_REGISTE↩ RS**]
- **Memory::Access** **instructionPointer**
- bool **interrupt**
- bool **serializing**
- bool **branch**
- bool **m_membar**
- bool **is_x87**
- uint16_t **operand_size**
- uint16_t **memoryAccessSize**

6.197.1 Detailed Description

An instruction will be decoded in MicroOperations. There are 3 MicroOperation types: LOAD, EXECUTE AND STORE.

Example below: instruction with 2 loads, 1 execute and 2 stores. +-—+-—+ | | | +-—+-—+-—+-—+-—+ | L1 | L2 | E1 | S1 | S2 | +-—+-—+-—+-—+-—+ | | | +-—+-—+

E1 depend on L1 and L2: intraInstructionDependencies = 2. S1 and S2 depend on E1: intraInstruction↩ Dependencies = 1 (for both S1 and S2).

The typeOffset for L1 = 0, L2 = 1, E1 = 0, S1 = 0, S2 = 1.

Getting the sequenceNumber of the MicroOperation for intra instruction dependencies: this->sequenceNumber - typeOffset (= sequenceNumber of first microOperation with the current type) - intraInstructionDependencies

Definition at line 60 of file micro_op.h.

6.197.2 Member Enumeration Documentation

6.197.2.1 uop_subtype_t

enum **MicroOp::uop_subtype_t**

Enumerator

UOP_SUBTYPE_FP_ADDSUB	
UOP_SUBTYPE_FP_MULDIV	
UOP_SUBTYPE_LOAD	
UOP_SUBTYPE_STORE	
UOP_SUBTYPE_GENERIC	
UOP_SUBTYPE_BRANCH	
UOP_SUBTYPE_SIZE	

Definition at line 68 of file micro_op.h.

6.197.2.2 uop_type_t

`enum` **`MicroOp::uop_type_t`**

Enumerator

UOP_INVALID	
UOP_LOAD	
UOP_EXECUTE	
UOP_STORE	

Definition at line 64 of file micro_op.h.

6.197.3 Constructor & Destructor Documentation

6.197.3.1 MicroOp()

```
MicroOp::MicroOp ( )
```

Definition at line 23 of file micro_op.cc.

References MAXIMUM_NUMBER_OF_ADDRESS_REGISTERS, MAXIMUM_NUMBER_OF_DESTINATION_R↩EGISTERS, and MAXIMUM_NUMBER_OF_SOURCE_REGISTERS.

6.197.4 Member Function Documentation

6.197.4.1 addAddressRegister()

```
void MicroOp::addAddressRegister (
            dl::Decoder::decoder_reg registerId,
            const String & registerName )
```

Definition at line 262 of file micro_op.cc.

References addressRegisters, addressRegistersLength, MAXIMUM_NUMBER_OF_ADDRESS_REGISTERS, and VERIFY_MICROOP.

Referenced by InstructionDecoder::addAddrs().

6.197.4.2 addDestinationRegister()

```
void MicroOp::addDestinationRegister (
            dl::Decoder::decoder_reg registerId,
            const String & registerName )
```

Definition at line 292 of file micro_op.cc.

References destinationRegisters, destinationRegistersLength, MAXIMUM_NUMBER_OF_DESTINATION_REGI↩ STERS, and VERIFY_MICROOP.

Referenced by InstructionDecoder::addDsts().

6.197.4.3 addSourceRegister()

```
void MicroOp::addSourceRegister (
            dl::Decoder::decoder_reg registerId,
            const String & registerName )
```

Definition at line 232 of file micro_op.cc.

References MAXIMUM_NUMBER_OF_SOURCE_REGISTERS, sourceRegisters, sourceRegistersLength, and VERIFY_MICROOP.

Referenced by InstructionDecoder::addSrcs().

6.197.4.4 getAddressRegister()

```
dl::Decoder::decoder_reg MicroOp::getAddressRegister (
            uint32_t index ) const
```

Definition at line 248 of file micro_op.cc.

References addressRegisters, addressRegistersLength, and VERIFY_MICROOP.

Referenced by RobSmtTimer::setStoreAddressProducers(), and RobTimer::simulate().

6.197.4.5 getAddressRegistersLength()

```
uint32_t MicroOp::getAddressRegistersLength ( ) const
```

Definition at line 243 of file micro_op.cc.

References addressRegistersLength, and VERIFY_MICROOP.

6.197.4.6 getDecodedInstruction()

```
const dl::DecodedInst* MicroOp::getDecodedInstruction (
            void  ) const  [inline]
```

Definition at line 159 of file micro_op.h.

References decodedInstruction.

Referenced by getSubtype_Exec(), and isFpLoadStore().

6.197.4.7 getDestinationRegister()

```
dl::Decoder::decoder_reg MicroOp::getDestinationRegister (
            uint32_t index ) const
```

Definition at line 278 of file micro_op.cc.

References destinationRegisters, destinationRegistersLength, and VERIFY_MICROOP.

Referenced by RegisterDependencies::setDependencies().

6.197.4.8 getDestinationRegistersLength()

```
uint32_t MicroOp::getDestinationRegistersLength ( ) const
```

Definition at line 273 of file micro_op.cc.

References destinationRegistersLength, and VERIFY_MICROOP.

Referenced by toString().

6.197.4.9 getInstruction()

```
Instruction* MicroOp::getInstruction (
            void  ) const  [inline]
```

Definition at line 156 of file micro_op.h.

References instruction.

Referenced by RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), IntervalTimer::issueMemOp(), RobTimer::printRob(), RobSmtTimer::printRob(), toShortString(), LoopProfiler::traceInstruction(), and LoopTracer::traceInstruction().

6.197.4.10 getInstructionOpcode()

```
dl::Decoder::decoder_opcode MicroOp::getInstructionOpcode ( ) const  [inline]
```

Definition at line 161 of file micro_op.h.

References instructionOpcode.

Referenced by DynamicMicroOpBoomV1::getAlu(), DynamicMicroOpNehalem::getAlu(), CoreModelBoomV1::getAluLatency(), CoreModelNehalem::getAluLatency(), CoreModelBoomV1::getInstructionLatency(), CoreModelNehalem::getInstructionLatency(), DynamicMicroOpBoomV1::getPort(), DynamicMicroOpNehalem::getPort(), getSubtype_Exec(), isFpLoadStore(), InstructionTracerPrint::traceInstruction(), and InstructionTracerFPStats::traceInstruction().

6.197.4.11 getInstructionPointer()

```
const  Memory::Access& MicroOp::getInstructionPointer ( ) const  [inline]
```

Definition at line 189 of file micro_op.h.

References instructionPointer.

6.197.4.12 getMemoryAccessSize()

```
uint16_t MicroOp::getMemoryAccessSize (
            void  ) const  [inline]
```

Definition at line 153 of file micro_op.h.

References memoryAccessSize.

Referenced by RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), and IntervalTimer::issueMemOp().

6.197.4.13 getOperandSize()

```
uint16_t MicroOp::getOperandSize (
            void  ) const  [inline]
```

Definition at line 152 of file micro_op.h.

References operand_size.

Referenced by CoreModelNehalem::getAluLatency(), and DynamicMicroOpNehalem::getPort().

6.197.4.14 getSourceRegister()

```
dl::Decoder::decoder_reg MicroOp::getSourceRegister (
            uint32_t index ) const
```

Definition at line 218 of file micro_op.cc.

References sourceRegisters, sourceRegistersLength, and VERIFY_MICROOP.

Referenced by RegisterDependencies::setDependencies().

6.197.4.15 getSourceRegistersLength()

```
uint32_t MicroOp::getSourceRegistersLength ( ) const
```

Definition at line 213 of file micro_op.cc.

References sourceRegistersLength, and VERIFY_MICROOP.

Referenced by toString().

6.197.4.16 getSubtype() [1/2]

```
MicroOp::uop_subtype_t MicroOp::getSubtype (
            const  MicroOp & uop )  [static]
```

Definition at line 144 of file micro_op.cc.

References getSubtype_Exec(), isBranch(), isExecute(), isLoad(), isStore(), UOP_SUBTYPE_BRANCH, UOP_S↩UBTYPE_GENERIC, UOP_SUBTYPE_LOAD, and UOP_SUBTYPE_STORE.

Referenced by DynamicMicroOpBoomV1::getBypassType(), DynamicMicroOpNehalem::getBypassType(), and getCpContrType().

6.197.4.17 getSubtype() [2/2]

```
uop_subtype_t MicroOp::getSubtype (
            void  ) const  [inline]
```

Definition at line 146 of file micro_op.h.

References uop_subtype.

Referenced by setTypes(), and verify().

6.197.4.18 getSubtype_Exec()

```
MicroOp::uop_subtype_t MicroOp::getSubtype_Exec (
            const  MicroOp & uop )  [static]
```

Definition at line 123 of file micro_op.cc.

References getDecodedInstruction(), getInstructionOpcode(), UOP_SUBTYPE_BRANCH, UOP_SUBTYPE_FP↩_ADDSUB, UOP_SUBTYPE_FP_MULDIV, and UOP_SUBTYPE_GENERIC.

Referenced by getSubtype(), and isFpLoadStore().

6.197.4.19 getSubtypeString()

```
String MicroOp::getSubtypeString (
             uop_subtype_t uop_subtype )  [static]
```

Definition at line 160 of file micro_op.cc.

References LOG_ASSERT_ERROR, uop_subtype, UOP_SUBTYPE_BRANCH, UOP_SUBTYPE_FP_ADDSUB, UOP_SUBTYPE_FP_MULDIV, UOP_SUBTYPE_GENERIC, UOP_SUBTYPE_LOAD, and UOP_SUBTYPE_ST↩ORE.

Referenced by RobSmtTimer::initializeThread(), IntervalTimer::IntervalTimer(), and RobTimer::RobTimer().

6.197.4.20 getType()

```
uop_type_t MicroOp::getType ( ) const  [inline]
```

Definition at line 215 of file micro_op.h.

References uop_type.

Referenced by MicroOpPerformanceModel::doSquashing().

6.197.4.21 isBranch()

```
bool MicroOp::isBranch ( ) const  [inline]
```

Definition at line 192 of file micro_op.h.

References branch.

Referenced by IntervalTimer::blockWindow(), IntervalTimer::dispatchInstruction(), RobTimer::doDispatch(), DynamicMicroOp::getBranchTarget(), getSubtype(), DynamicMicroOp::isBranchMispredicted(), DynamicMicro↩Op::isBranchTaken(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), makeExecute(), Dynamic↩MicroOp::setBranchTaken(), toString(), and RobSmtTimer::tryDispatch().

6.197.4.22 isCacheFlush()

```
bool MicroOp::isCacheFlush ( ) const  [inline]
```

Definition at line 202 of file micro_op.h.

6.197.4.23 isDiv()

```
bool MicroOp::isDiv ( ) const  [inline]
```

Definition at line 203 of file micro_op.h.

6.197.4.24 isExecute()

```
bool MicroOp::isExecute ( ) const  [inline]
```

Definition at line 213 of file micro_op.h.

References UOP_EXECUTE, and uop_type.

Referenced by IntervalTimer::dispatchInstruction(), and getSubtype().

6.197.4.25 isFirst()

```
bool MicroOp::isFirst ( ) const  [inline]
```

Definition at line 164 of file micro_op.h.

References first.

Referenced by DynamicMicroOp::DynamicMicroOp(), toString(), InstructionTracerFPStats::traceInstruction(), and LoopProfiler::traceInstruction().

6.197.4.26 isFpLoadStore()

```
bool MicroOp::isFpLoadStore ( ) const
```

Definition at line 181 of file micro_op.cc.

References getDecodedInstruction(), getInstructionOpcode(), getSubtype_Exec(), isLoad(), isStore(), UOP_SUB↩ TYPE_FP_ADDSUB, and UOP_SUBTYPE_FP_MULDIV.

Referenced by DynamicMicroOpBoomV1::getBypassType(), and DynamicMicroOpNehalem::getBypassType().

6.197.4.27 isInterrupt()

```
bool MicroOp::isInterrupt ( ) const  [inline]
```

Definition at line 194 of file micro_op.h.

References interrupt.

Referenced by IntervalTimer::dispatchInstruction(), and toString().

6.197.4.28 isLast()

```
bool MicroOp::isLast ( ) const  [inline]
```

Definition at line 167 of file micro_op.h.

References last.

Referenced by DynamicMicroOp::DynamicMicroOp(), toString(), and LoopTracer::traceInstruction().

6.197.4.29 isLoad()

```
bool MicroOp::isLoad ( ) const  [inline]
```

Definition at line 206 of file micro_op.h.

References UOP_LOAD.

Referenced by IntervalTimer::blockWindow(), RobTimer::countOutstandingMemop(), RobSmtTimer::count↩ OutstandingMemop(), IntervalTimer::dispatchInstruction(), RobTimer::doIssue(), RobTimer::findCpiComponent(), RobSmtTimer::findCpiComponent(), getSubtype(), isFpLoadStore(), RobTimer::issueInstruction(), RobSmtTimer↩ ::issueInstruction(), IntervalTimer::issueMemOp(), RobTimer::printRob(), RobSmtTimer::printRob(), Memory↩ Dependencies::setDependencies(), RobSmtTimer::setDependencies(), RobTimer::simulate(), and RobSmtTimer↩ ::tryIssue().

6.197.4.30 isMemBarrier()

```
bool MicroOp::isMemBarrier ( ) const  [inline]
```

Definition at line 200 of file micro_op.h.

References m_membar.

Referenced by IntervalTimer::blockWindow(), IntervalTimer::dispatchInstruction(), RobTimer::doIssue(), RobTimer::findCpiComponent(), RobSmtTimer::findCpiComponent(), MemoryDependencies::setDependencies(), and RobSmtTimer::tryIssue().

6.197.4.31 isPause()

```
bool MicroOp::isPause ( ) const  [inline]
```

Definition at line 204 of file micro_op.h.

6.197.4.32 isSerializing()

```
bool MicroOp::isSerializing ( ) const  [inline]
```

Definition at line 197 of file micro_op.h.

References serializing.

Referenced by IntervalTimer::blockWindow(), IntervalTimer::dispatchInstruction(), RobTimer::doIssue(), RobTimer::findCpiComponent(), RobSmtTimer::findCpiComponent(), toString(), and RobSmtTimer::tryIssue().

6.197.4.33 isStore()

```
bool MicroOp::isStore ( ) const  [inline]
```

Definition at line 207 of file micro_op.h.

References UOP_STORE.

Referenced by IntervalTimer::dispatchInstruction(), RobTimer::doIssue(), RobTimer::findCpiComponent(), RobSmtTimer::findCpiComponent(), getSubtype(), isFpLoadStore(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), IntervalTimer::issueMemOp(), RobTimer::printRob(), RobSmtTimer::printRob(), MemoryDependencies::setDependencies(), RobSmtTimer::setDependencies(), RobTimer::simulate(), and RobSmtTimer::tryIssue().

6.197.4.34 isX87()

```
bool MicroOp::isX87 (
            void  ) const  [inline]
```

Definition at line 150 of file micro_op.h.

References is_x87.

6.197.4.35 makeDynamic()

```
void MicroOp::makeDynamic (
            const String & instructionOpcodeName,
            uint32_t execLatency )
```

Definition at line 110 of file micro_op.cc.

References branch, instructionOpcode, intraInstructionDependencies, microOpTypeOffset, setTypes(), UOP_EX↩ECUTE, and uop_type.

Referenced by MicroOpPerformanceModel::MicroOpPerformanceModel().

6.197.4.36 makeExecute()

```
void MicroOp::makeExecute (
            uint32_t offset,
            uint32_t num_loads,
            dl::Decoder::decoder_opcode instructionOpcode,
            const String & instructionOpcodeName,
            bool isBranch )
```

Definition at line 86 of file micro_op.cc.

References branch, instructionOpcode, intraInstructionDependencies, isBranch(), microOpTypeOffset, setTypes(), UOP_EXECUTE, and uop_type.

Referenced by InstructionDecoder::decode().

6.197.4.37 makeLoad()

```
void MicroOp::makeLoad (
            uint32_t offset,
            dl::Decoder::decoder_opcode instructionOpcode,
            const String & instructionOpcodeName,
            uint16_t mem_size )
```

Definition at line 74 of file micro_op.cc.

References instructionOpcode, intraInstructionDependencies, memoryAccessSize, microOpTypeOffset, setTypes(), UOP_LOAD, and uop_type.

Referenced by InstructionDecoder::decode(), and MicroOpPerformanceModel::MicroOpPerformanceModel().

6.197.4.38 makeStore()

```
void MicroOp::makeStore (
            uint32_t offset,
            uint32_t num_execute,
            dl::Decoder::decoder_opcode instructionOpcode,
            const String & instructionOpcodeName,
            uint16_t mem_size )
```

Definition at line 98 of file micro_op.cc.

References instructionOpcode, intraInstructionDependencies, memoryAccessSize, microOpTypeOffset, setTypes(), UOP_STORE, and uop_type.

Referenced by InstructionDecoder::decode().

6.197.4.39 setDebugInfo()

```
void MicroOp::setDebugInfo (
            String debugInfo )
```

6.197.4.40 setDecodedInstruction()

```
void MicroOp::setDecodedInstruction (
            const dl::DecodedInst * instr )  [inline]
```

Definition at line 158 of file micro_op.h.

References decodedInstruction.

Referenced by InstructionDecoder::decode().

6.197.4.41 setFirst()

```
void MicroOp::setFirst (
            bool first )  [inline]
```

Definition at line 163 of file micro_op.h.

References first.

Referenced by InstructionDecoder::decode(), and MicroOpPerformanceModel::MicroOpPerformanceModel().

6.197.4.42 setInstruction()

```
void MicroOp::setInstruction (
              Instruction * instr )  [inline]
```

Definition at line 155 of file micro_op.h.

References instruction.

Referenced by InstructionDecoder::decode().

6.197.4.43 setInstructionPointer()

```
void MicroOp::setInstructionPointer (
             const  Memory::Access & ip )  [inline]
```

Definition at line 190 of file micro_op.h.

Referenced by InstructionDecoder::decode().

6.197.4.44 setInterrupt()

```
void MicroOp::setInterrupt (
             bool interrupt )  [inline]
```

Definition at line 195 of file micro_op.h.

References interrupt.

Referenced by InstructionDecoder::decode().

6.197.4.45 setIsX87()

```
void MicroOp::setIsX87 (
             bool _is_x87 )  [inline]
```

Definition at line 149 of file micro_op.h.

References is_x87.

Referenced by InstructionDecoder::decode().

6.197.4.46 setLast()

```
void MicroOp::setLast (
            bool last )  [inline]
```

Definition at line 166 of file micro_op.h.

References last.

Referenced by InstructionDecoder::decode(), and MicroOpPerformanceModel::MicroOpPerformanceModel().

6.197.4.47 setMemBarrier()

```
void MicroOp::setMemBarrier (
            bool membar )  [inline]
```

Definition at line 201 of file micro_op.h.

Referenced by InstructionDecoder::decode(), and MicroOpPerformanceModel::MicroOpPerformanceModel().

6.197.4.48 setOperandSize()

```
void MicroOp::setOperandSize (
            int size )  [inline]
```

Definition at line 151 of file micro_op.h.

References operand_size.

Referenced by InstructionDecoder::decode().

6.197.4.49 setSerializing()

```
void MicroOp::setSerializing (
            bool serializing )  [inline]
```

Definition at line 198 of file micro_op.h.

References serializing.

Referenced by InstructionDecoder::decode(), and MicroOpPerformanceModel::MicroOpPerformanceModel().

6.197.4.50 setTypes()

```
void MicroOp::setTypes ( )  [inline]
```

Definition at line 145 of file micro_op.h.

References getSubtype(), and uop_subtype.

Referenced by makeDynamic(), makeExecute(), makeLoad(), and makeStore().

6.197.4.51 toShortString()

```
String MicroOp::toShortString (
            bool withDisassembly = false ) const
```

Definition at line 362 of file micro_op.cc.

References Instruction::getDisassembly(), getInstruction(), instructionOpcode, UOP_EXECUTE, UOP_LOAD, U↩OP_STORE, and uop_type.

Referenced by RobTimer::doCommit(), RobSmtTimer::doCommit(), RobTimer::doDispatch(), RobTimer::issue↩Instruction(), RobSmtTimer::issueInstruction(), RobSmtTimer::pushInstructions(), RobTimer::simulate(), and Rob↩SmtTimer::tryDispatch().

6.197.4.52 toString()

```
String MicroOp::toString ( ) const
```

Definition at line 303 of file micro_op.cc.

References getDestinationRegistersLength(), getSourceRegistersLength(), instructionOpcode, isBranch(), isFirst(), isInterrupt(), isLast(), isSerializing(), UOP_EXECUTE, UOP_LOAD, UOP_STORE, and uop_type.

Referenced by Windows::toString().

6.197.4.53 verify()

```
void MicroOp::verify ( ) const
```

Definition at line 203 of file micro_op.cc.

References destinationRegisters, destinationRegistersLength, getSubtype(), LOG_ASSERT_ERROR, MAXIM↩UM_NUMBER_OF_DESTINATION_REGISTERS, MAXIMUM_NUMBER_OF_SOURCE_REGISTERS, source↩Registers, sourceRegistersLength, and uop_subtype.

6.197.5 Member Data Documentation

6.197.5.1 addressRegisters

```
dl::Decoder::decoder_reg MicroOp::addressRegisters[ MAXIMUM_NUMBER_OF_ADDRESS_REGISTERS]
```

Definition at line 104 of file micro_op.h.

Referenced by addAddressRegister(), and getAddressRegister().

6.197.5.2 addressRegistersLength

```
uint32_t MicroOp::addressRegistersLength
```

This field contains the length of the destinationRegisters array.

Definition at line 103 of file micro_op.h.

Referenced by addAddressRegister(), getAddressRegister(), and getAddressRegistersLength().

6.197.5.3 branch

```
bool MicroOp::branch
```

Is this instruction a branch ?

Definition at line 124 of file micro_op.h.

Referenced by isBranch(), makeDynamic(), and makeExecute().

6.197.5.4 decodedInstruction

```
const dl::DecodedInst* MicroOp::decodedInstruction
```

Definition at line 85 of file micro_op.h.

Referenced by getDecodedInstruction(), and setDecodedInstruction().

6.197.5.5 destinationRegisters

```
dl::Decoder::decoder_reg MicroOp::destinationRegisters[ MAXIMUM_NUMBER_OF_DESTINATION_REGISTERS]
```

This array contains the registers written by this MicroOperation, the integer is an id given by libdisasm64. Only valid for UOP_EXECUTE.

Definition at line 107 of file micro_op.h.

Referenced by addDestinationRegister(), getDestinationRegister(), and verify().

6.197.5.6 destinationRegistersLength

```
uint32_t MicroOp::destinationRegistersLength
```

Definition at line 105 of file micro_op.h.

Referenced by addDestinationRegister(), getDestinationRegister(), getDestinationRegistersLength(), and verify().

6.197.5.7 first

```
bool MicroOp::first
```

This microOp is the first microOp of the instruction.

Definition at line 80 of file micro_op.h.

Referenced by isFirst(), and setFirst().

6.197.5.8 instruction

```
Instruction* MicroOp::instruction
```

Definition at line 87 of file micro_op.h.

Referenced by getInstruction(), and setInstruction().

6.197.5.9 instructionOpcode

```
dl::Decoder::decoder_opcode MicroOp::instructionOpcode
```

Definition at line 84 of file micro_op.h.

Referenced by getInstructionOpcode(), makeDynamic(), makeExecute(), makeLoad(), makeStore(), toShort↩ String(), and toString().

6.197.5.10 instructionPointer

```
Memory::Access MicroOp::instructionPointer
```

The instruction pointer.

Definition at line 116 of file micro_op.h.

Referenced by getInstructionPointer().

6.197.5.11 interrupt

```
bool MicroOp::interrupt
```

Is this microOp an interrupt ?

Definition at line 119 of file micro_op.h.

Referenced by isInterrupt(), and setInterrupt().

6.197.5.12 intraInstructionDependencies

```
uint32_t MicroOp::intraInstructionDependencies
```

The intraInstructionDependencies variable gives the number of preceding microOp on which this micrOp depends. This number is counted from the first microOp with that type.

Definition at line 96 of file micro_op.h.

Referenced by DynamicMicroOp::DynamicMicroOp(), makeDynamic(), makeExecute(), makeLoad(), and make↩ Store().

6.197.5.13 is_x87

```
bool MicroOp::is_x87
```

Definition at line 132 of file micro_op.h.

Referenced by isX87(), and setIsX87().

6.197.5.14 last

```
bool MicroOp::last
```

This microOp is the last microOp of the instruction.

Definition at line 82 of file micro_op.h.

Referenced by isLast(), and setLast().

6.197.5.15 m_membar

```
bool MicroOp::m_membar
```

Debug info about the microOperation.

Definition at line 131 of file micro_op.h.

Referenced by isMemBarrier().

6.197.5.16 memoryAccessSize

```
uint16_t MicroOp::memoryAccessSize
```

Definition at line 134 of file micro_op.h.

Referenced by getMemoryAccessSize(), makeLoad(), and makeStore().

6.197.5.17 microOpTypeOffset

```
uint32_t MicroOp::microOpTypeOffset
```

The typeOffset field contains the offset of the microOp starting from the first microOp with that type.

Definition at line 94 of file micro_op.h.

Referenced by MicroOpPerformanceModel::doSquashing(), DynamicMicroOp::DynamicMicroOp(), make↩ Dynamic(), makeExecute(), makeLoad(), and makeStore().

6.197.5.18 operand_size

```
uint16_t MicroOp::operand_size
```

Definition at line 133 of file micro_op.h.

Referenced by getOperandSize(), and setOperandSize().

6.197.5.19 serializing

```
bool MicroOp::serializing
```

Is this microOp serializing ?

Definition at line 121 of file micro_op.h.

Referenced by isSerializing(), and setSerializing().

6.197.5.20 sourceRegisters

```
dl::Decoder::decoder_reg MicroOp::sourceRegisters[ MAXIMUM_NUMBER_OF_SOURCE_REGISTERS]
```

This array contains the registers read by this MicroOperation, the integer is an id given by libdisasm64. Only valid for UOP_LOAD and UOP_EXECUTE.

Definition at line 101 of file micro_op.h.

Referenced by addSourceRegister(), getSourceRegister(), and verify().

6.197.5.21 sourceRegistersLength

```
uint32_t MicroOp::sourceRegistersLength
```

This field contains the length of the sourceRegisters array.

Definition at line 99 of file micro_op.h.

Referenced by addSourceRegister(), getSourceRegister(), getSourceRegistersLength(), and verify().

6.197.5.22 uop_subtype

uop_subtype_t MicroOp::uop_subtype

Definition at line 77 of file micro_op.h.

Referenced by DynamicMicroOpNehalem::getPort(), getSubtype(), getSubtypeString(), setTypes(), and verify().

6.197.5.23 uop_type

uop_type_t MicroOp::uop_type

The microOperation type.

Definition at line 66 of file micro_op.h.

Referenced by DynamicMicroOpBoomV1::getAlu(), DynamicMicroOpNehalem::getAlu(), getType(), isExecute(), makeDynamic(), makeExecute(), makeLoad(), makeStore(), toShortString(), and toString().

The documentation for this struct was generated from the following files:

- common/performance_model/performance_models/micro_op/ **micro_op.h**
- common/performance_model/performance_models/micro_op/ **micro_op.cc**

6.198 MicroOpPerformanceModel Class Reference

`#include <micro_op_performance_model.h>`

Inheritance diagram for MicroOpPerformanceModel:

Public Member Functions

- **MicroOpPerformanceModel** (**Core** *core, bool issue_memops)
- ~**MicroOpPerformanceModel** ()

Protected Member Functions

- virtual boost::tuple< uint64_t, uint64_t > **simulate** (const std::vector< **DynamicMicroOp** * > &insts)=0
- virtual void **notifyElapsedTimeUpdate** ()=0
- void **doSquashing** (std::vector< **DynamicMicroOp** * > &current_uops, uint32_t first_squashed=0)

Protected Attributes

- const **CoreModel** * **m_core_model**

Private Member Functions

- void **handleInstruction** (**DynamicInstruction** *instruction)

Private Attributes

- **Allocator** * **m_allocator**
- const bool **m_issue_memops**
- std::vector< **DynamicMicroOp** * > **m_current_uops**
- std::vector< **IntPtr** > **m_cache_lines_read**
- std::vector< **IntPtr** > **m_cache_lines_written**
- **UInt64** **m_dyninsn_count**
- **UInt64** **m_dyninsn_cost**
- **UInt64** **m_dyninsn_zero_count**
- **SubsecondTime** **m_cpiITLBMiss**
- **SubsecondTime** **m_cpiDTLBMiss**
- **SubsecondTime** **m_cpiUnknown**
- **SubsecondTime** **m_cpiMemAccess**

Static Private Attributes

- static **MicroOp** * **m_serialize_uop** = NULL
- static **MicroOp** * **m_mfence_uop** = NULL
- static **MicroOp** * **m_memaccess_uop** = NULL

Additional Inherited Members

6.198.1 Detailed Description

Definition at line 18 of file micro_op_performance_model.h.

6.198.2 Constructor & Destructor Documentation

6.198.2.1 MicroOpPerformanceModel()

```
MicroOpPerformanceModel::MicroOpPerformanceModel (
              Core * core,
             bool issue_memops )
```

Definition at line 22 of file micro_op_performance_model.cc.

References Core::getId(), itostr(), m_cpiDTLBMiss, m_cpiITLBMiss, m_cpiMemAccess, m_cpiUnknown, m_↩ dyninsn_cost, m_dyninsn_count, m_dyninsn_zero_count, m_memaccess_uop, m_mfence_uop, m_serialize_↩ uop, MicroOp::makeDynamic(), MicroOp::makeLoad(), registerStatsMetric(), MicroOp::setFirst(), MicroOp::set↩ Last(), MicroOp::setMemBarrier(), MicroOp::setSerializing(), and SubsecondTime::Zero().

6.198.2.2 ~MicroOpPerformanceModel()

```
MicroOpPerformanceModel::~MicroOpPerformanceModel ( )
```

Definition at line 95 of file micro_op_performance_model.cc.

References m_allocator.

6.198.3 Member Function Documentation

6.198.3.1 doSquashing()

```
void MicroOpPerformanceModel::doSquashing (
            std::vector< DynamicMicroOp * > & current_uops,
            uint32_t first_squashed = 0 ) [protected]
```

Definition at line 109 of file micro_op_performance_model.cc.

References DynamicMicroOp::getIntraInstrDependenciesLength(), DynamicMicroOp::getMicroOp(), MicroOp::getType(), DynamicMicroOp::isSquashed(), LOG_ASSERT_ERROR, MicroOp::microOpTypeOffset, DynamicMicroOp::setIntraInstrDependenciesLength(), DynamicMicroOp::setMicroOpTypeOffset(), DynamicMicroOp::setSquashedCount(), and MicroOp::UOP_INVALID.

Referenced by handleInstruction().

6.198.3.2 handleInstruction()

```
void MicroOpPerformanceModel::handleInstruction (
             DynamicInstruction * instruction ) [private], [virtual]
```

Implements **PerformanceModel** (p. 911).

Definition at line 153 of file micro_op_performance_model.cc.

References DynamicInstruction::accessMemory(), ComponentTime::addCycleLatency(), ComponentTime::addLatency(), DynamicInstruction::MemoryInfo::addr, DynamicInstruction::branch_info, CoreModel::createDynamicMicroOp(), DynamicInstruction::MemoryInfo::dir, SubsecondTime::divideRounded(), doSquashing(), DynamicInstruction::eip, Instruction::getAddress(), DynamicInstruction::getBranchCost(), CoreModel::getBypassLatency(), PerformanceModel::getCore(), DynamicInstruction::getCost(), MemAccessInstruction::getDataAddress(), Instruction::getDisassembly(), ComponentTime::getElapsedTime(), ComponentTime::getLatencyGenerator(), CoreModel::getLongLatencyCutoff(), Instruction::getMicroOps(), Instruction::getOperands(), ComponentPeriod::getPeriod(), Instruction::getSize(), Instruction::getType(), Instruction::getTypeName(), MemoryResult::hit_where, DynamicInstruction::MemoryInfo::hit_where, INST_BRANCH, INST_MEM_ACCESS, INST_RECV, INST_SYNC, INST_TLB_MISS, INST_UNKNOWN, DynamicInstruction::instruction, MemAccessInstruction::isFence(), TLBMissInstruction::isIfetch(), Instruction::isPseudo(), itostr(), MemoryResult::latency, DynamicInstruction::MemoryInfo::latency, LOG_ASSERT_ERROR, LOG_PRINT_ERROR, m_allocator, m_cache_lines_read, m_cache_lines_written, m_core_model, m_cpiDTLBMiss, m_cpiITLBMiss, m_cpiMemAccess, m_cpiUnknown, m_current_uops, Operand::m_direction, m_dyninsn_count, m_dyninsn_zero_count, PerformanceModel::m_elapsed_time, PerformanceModel::m_instruction_count, m_issue_memops, m_memaccess_uop, m_mfence_uop, m_serialize_uop, Operand::m_type, Operand::MEMORY, DynamicInstruction::memory_info, notifyElapsedTimeUpdate(), DynamicInstruction::num_memory, Operand::READ, Core::readInstructionMemory(), Memory::Access::set(), DynamicMicroOp::setExecLatency(), DynamicMicroOp::setForceLongLatencyLoad(), simulate(), DynamicInstruction::BranchInfo::taken, DynamicInstruction::BranchInfo::target, and SubsecondTime::Zero().

6.198.3.3 notifyElapsedTimeUpdate()

```
virtual void MicroOpPerformanceModel::notifyElapsedTimeUpdate ( )  [protected], [pure virtual]
```

Reimplemented from **PerformanceModel** (p. 913).

Implemented in **RobSmtPerformanceModel** (p. 1014), **IntervalPerformanceModel** (p. 648), and **Rob↩ PerformanceModel** (p. 1011).

Referenced by handleInstruction().

6.198.3.4 simulate()

```
virtual boost::tuple<uint64_t,uint64_t> MicroOpPerformanceModel::simulate (
            const std::vector<  DynamicMicroOp * > & insts )  [protected], [pure virtual]
```

Implemented in **RobSmtPerformanceModel** (p. 1015), **IntervalPerformanceModel** (p. 648), and **Rob↩ PerformanceModel** (p. 1012).

Referenced by handleInstruction().

6.198.4 Member Data Documentation

6.198.4.1 m_allocator

```
Allocator* MicroOpPerformanceModel::m_allocator  [private]
```

Definition at line 38 of file micro_op_performance_model.h.

Referenced by handleInstruction(), and ∼MicroOpPerformanceModel().

6.198.4.2 m_cache_lines_read

```
std::vector< IntPtr> MicroOpPerformanceModel::m_cache_lines_read  [private]
```

Definition at line 45 of file micro_op_performance_model.h.

Referenced by handleInstruction().

6.198.4.3 m_cache_lines_written

```
std::vector< IntPtr> MicroOpPerformanceModel::m_cache_lines_written  [private]
```

Definition at line 46 of file micro_op_performance_model.h.

Referenced by handleInstruction().

6.198.4.4 m_core_model

```
const  CoreModel* MicroOpPerformanceModel::m_core_model  [protected]
```

Definition at line 25 of file micro_op_performance_model.h.

Referenced by handleInstruction().

6.198.4.5 m_cpiDTLBMiss

```
SubsecondTime MicroOpPerformanceModel::m_cpiDTLBMiss  [private]
```

Definition at line 63 of file micro_op_performance_model.h.

Referenced by handleInstruction(), and MicroOpPerformanceModel().

6.198.4.6 m_cpiITLBMiss

```
SubsecondTime MicroOpPerformanceModel::m_cpiITLBMiss  [private]
```

Definition at line 62 of file micro_op_performance_model.h.

Referenced by handleInstruction(), and MicroOpPerformanceModel().

6.198.4.7 m_cpiMemAccess

```
SubsecondTime MicroOpPerformanceModel::m_cpiMemAccess  [private]
```

Definition at line 65 of file micro_op_performance_model.h.

Referenced by handleInstruction(), and MicroOpPerformanceModel().

6.198.4.8 m_cpiUnknown

SubsecondTime MicroOpPerformanceModel::m_cpiUnknown [private]

Definition at line 64 of file micro_op_performance_model.h.

Referenced by handleInstruction(), and MicroOpPerformanceModel().

6.198.4.9 m_current_uops

std::vector< **DynamicMicroOp***> MicroOpPerformanceModel::m_current_uops [private]

Definition at line 41 of file micro_op_performance_model.h.

Referenced by handleInstruction().

6.198.4.10 m_dyninsn_cost

UInt64 MicroOpPerformanceModel::m_dyninsn_cost [private]

Definition at line 49 of file micro_op_performance_model.h.

Referenced by MicroOpPerformanceModel().

6.198.4.11 m_dyninsn_count

UInt64 MicroOpPerformanceModel::m_dyninsn_count [private]

Definition at line 48 of file micro_op_performance_model.h.

Referenced by handleInstruction(), and MicroOpPerformanceModel().

6.198.4.12 m_dyninsn_zero_count

UInt64 MicroOpPerformanceModel::m_dyninsn_zero_count [private]

Definition at line 50 of file micro_op_performance_model.h.

Referenced by handleInstruction(), and MicroOpPerformanceModel().

6.198.4.13 m_issue_memops

```
const bool MicroOpPerformanceModel::m_issue_memops  [private]
```

Definition at line 39 of file micro_op_performance_model.h.

Referenced by handleInstruction().

6.198.4.14 m_memaccess_uop

```
MicroOp * MicroOpPerformanceModel::m_memaccess_uop = NULL  [static], [private]
```

Definition at line 36 of file micro_op_performance_model.h.

Referenced by handleInstruction(), and MicroOpPerformanceModel().

6.198.4.15 m_mfence_uop

```
MicroOp * MicroOpPerformanceModel::m_mfence_uop = NULL  [static], [private]
```

Definition at line 35 of file micro_op_performance_model.h.

Referenced by handleInstruction(), and MicroOpPerformanceModel().

6.198.4.16 m_serialize_uop

```
MicroOp * MicroOpPerformanceModel::m_serialize_uop = NULL  [static], [private]
```

Definition at line 34 of file micro_op_performance_model.h.

Referenced by handleInstruction(), and MicroOpPerformanceModel().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/ **micro_op_performance_model.h**
- common/performance_model/performance_models/ **micro_op_performance_model.cc**

6.199 MMUPerfModel Class Reference

```
#include <mmu_perf_model.h>
```

Inheritance diagram for MMUPerfModel:

Public Member Functions

- **MMUPerfModel** ()
- ~**MMUPerfModel** ()
- **UInt32 getLatency** (**MMUActions_t** action)

Additional Inherited Members

6.199.1 Detailed Description

Definition at line 7 of file mmu_perf_model.h.

6.199.2 Constructor & Destructor Documentation

6.199.2.1 MMUPerfModel()

```
MMUPerfModel::MMUPerfModel ( )  [inline]
```

Definition at line 12 of file mmu_perf_model.h.

6.199.2.2 ~MMUPerfModel()

```
MMUPerfModel::~MMUPerfModel ( )  [inline]
```

Definition at line 13 of file mmu_perf_model.h.

6.199.3 Member Function Documentation

6.199.3.1 getLatency()

```
UInt32 MMUPerfModel::getLatency (
            MMUActions_t action )  [inline], [virtual]
```

Reimplemented from **MMUPerfModelBase** (p. 796).

Definition at line 15 of file mmu_perf_model.h.

References LOG_ASSERT_ERROR, MMUPerfModelBase::RECEIVE_MESSAGE, MMUPerfModelBase::SEND↩ _MESSAGE, MMUPerfModelBase::shmem_receive_message_delay, and MMUPerfModelBase::shmem_send_↩ message_delay.

The documentation for this class was generated from the following file:

- common/performance_model/ **mmu_perf_model.h**

6.200 MMUPerfModelBase Class Reference

`#include <mmu_perf_model_base.h>`

Inheritance diagram for MMUPerfModelBase:

Public Types

- enum **MMUActions_t** { **SEND_MESSAGE**, **RECEIVE_MESSAGE**, **NUM_MMU_ACTIONS** }
- enum **MMUPerfModel_t** { **MMU_PERF_MODEL** = 0, **NUM_MMU_PERF_MODELS** }

Public Member Functions

- **MMUPerfModelBase** ()
- virtual ~**MMUPerfModelBase** ()
- virtual **UInt32** **getLatency** (**MMUActions_t** action)

Static Public Member Functions

- static **MMUPerfModelBase** * **createModel** (**UInt32** type)

Protected Attributes

- **UInt32** **shmem_send_message_delay**
- **UInt32** **shmem_receive_message_delay**

6.200.1 Detailed Description

Definition at line 8 of file mmu_perf_model_base.h.

6.200.2 Member Enumeration Documentation

6.200.2.1 MMUActions_t

enum **MMUPerfModelBase::MMUActions_t**

Enumerator

SEND_MESSAGE	
RECEIVE_MESSAGE	
NUM_MMU_ACTIONS	

Definition at line 26 of file mmu_perf_model_base.h.

6.200.2.2 MMUPerfModel_t

`enum` **MMUPerfModelBase::MMUPerfModel_t**

Enumerator

MMU_PERF_MODEL	
NUM_MMU_PERF_MODELS	

Definition at line 33 of file mmu_perf_model_base.h.

6.200.3 Constructor & Destructor Documentation

6.200.3.1 MMUPerfModelBase()

```
MMUPerfModelBase::MMUPerfModelBase ( )  [inline]
```

Definition at line 16 of file mmu_perf_model_base.h.

6.200.3.2 ~MMUPerfModelBase()

```
virtual MMUPerfModelBase::~MMUPerfModelBase ( )  [inline], [virtual]
```

Definition at line 22 of file mmu_perf_model_base.h.

6.200.4 Member Function Documentation

6.200.4.1 createModel()

```
MMUPerfModelBase * MMUPerfModelBase::createModel (
            UInt32 type )  [static]
```

Definition at line 6 of file mmu_perf_model_base.cc.

References LOG_ASSERT_ERROR, and MMU_PERF_MODEL.

6.200.4.2 getLatency()

```
virtual  UInt32 MMUPerfModelBase::getLatency (
            MMUActions_t action )  [inline], [virtual]
```

Reimplemented in **MMUPerfModel** (p. 793).

Definition at line 39 of file mmu_perf_model_base.h.

6.200.5 Member Data Documentation

6.200.5.1 shmem_receive_message_delay

```
UInt32 MMUPerfModelBase::shmem_receive_message_delay  [protected]
```

Definition at line 12 of file mmu_perf_model_base.h.

Referenced by MMUPerfModel::getLatency().

6.200.5.2 shmem_send_message_delay

```
UInt32 MMUPerfModelBase::shmem_send_message_delay  [protected]
```

Definition at line 11 of file mmu_perf_model_base.h.

Referenced by MMUPerfModel::getLatency().

The documentation for this class was generated from the following files:

- common/performance_model/ **mmu_perf_model_base.h**
- common/performance_model/ **mmu_perf_model_base.cc**

6.201 ModuloNum Class Reference

`#include <modulo_num.h>`

Public Member Functions

- **ModuloNum** (**UInt32** max_value=-1, **UInt32** value=0)
- ~**ModuloNum** ()
- **UInt32 getValue** () const
- **UInt32 getMaxValue** () const
- void **setValue** (**UInt32** value)
- void **setMaxValue** (**UInt32** max_value)
- **ModuloNum operator+** (const **ModuloNum** &num) const
- **ModuloNum operator-** (const **ModuloNum** &num) const
- **ModuloNum operator+** (**UInt32** value) const
- **ModuloNum operator-** (**UInt32** value) const
- bool **operator==** (const **ModuloNum** &num) const
- bool **operator!=** (const **ModuloNum** &num) const

Private Attributes

- **UInt32 m_value**
- **UInt32 m_max_value**

6.201.1 Detailed Description

Definition at line 6 of file modulo_num.h.

6.201.2 Constructor & Destructor Documentation

6.201.2.1 ModuloNum()

```
ModuloNum::ModuloNum (
              UInt32 max_value = -1,
              UInt32 value = 0 )
```

Definition at line 5 of file modulo_num.cc.

6.201.2.2 ~ModuloNum()

```
ModuloNum::~ModuloNum ( )
```

Definition at line 11 of file modulo_num.cc.

6.201.3 Member Function Documentation

6.201.3.1 getMaxValue()

```
UInt32 ModuloNum::getMaxValue ( ) const  [inline]
```

Definition at line 17 of file modulo_num.h.

References m_max_value.

Referenced by operator+(), operator-(), and operator==().

6.201.3.2 getValue()

```
UInt32 ModuloNum::getValue ( ) const  [inline]
```

Definition at line 16 of file modulo_num.h.

References m_value.

Referenced by MovingGeometricMean< T >::compute(), operator+(), operator-(), operator==(), MovingAverage< SubsecondTime >::printElements(), and MovingArithmeticMean< T >::update().

6.201.3.3 operator"!=()

```
bool ModuloNum::operator!= (
            const  ModuloNum & num ) const
```

Definition at line 61 of file modulo_num.cc.

6.201.3.4 operator+() [1/2]

```
ModuloNum ModuloNum::operator+ (
            const  ModuloNum & num ) const
```

Definition at line 15 of file modulo_num.cc.

References getMaxValue(), getValue(), m_max_value, m_value, and setValue().

6.201.3.5 operator+() [2/2]

```
ModuloNum ModuloNum::operator+ (
            UInt32 value ) const
```

Definition at line 39 of file modulo_num.cc.

References m_max_value, and setValue().

6.201.3.6 operator-() [1/2]

```
ModuloNum ModuloNum::operator- (
            const  ModuloNum & num ) const
```

Definition at line 25 of file modulo_num.cc.

References getMaxValue(), getValue(), m_max_value, m_value, and setValue().

6.201.3.7 operator-() [2/2]

```
ModuloNum ModuloNum::operator- (
            UInt32 value ) const
```

Definition at line 47 of file modulo_num.cc.

References m_max_value, and setValue().

6.201.3.8 operator==()

```
bool ModuloNum::operator== (
            const  ModuloNum & num ) const
```

Definition at line 55 of file modulo_num.cc.

References getMaxValue(), getValue(), m_max_value, and m_value.

6.201.3.9 setMaxValue()

```
void ModuloNum::setMaxValue (
            UInt32 max_value )  [inline]
```

Definition at line 19 of file modulo_num.h.

References m_max_value.

6.201.3.10 setValue()

```
void ModuloNum::setValue (
              UInt32 value )  [inline]
```

Definition at line 18 of file modulo_num.h.

References m_value.

Referenced by operator+(), and operator-().

6.201.4 Member Data Documentation

6.201.4.1 m_max_value

```
UInt32 ModuloNum::m_max_value  [private]
```

Definition at line 10 of file modulo_num.h.

Referenced by getMaxValue(), operator+(), operator-(), operator==(), and setMaxValue().

6.201.4.2 m_value

```
UInt32 ModuloNum::m_value  [private]
```

Definition at line 9 of file modulo_num.h.

Referenced by getValue(), operator+(), operator-(), operator==(), and setValue().

The documentation for this class was generated from the following files:

- common/misc/ **modulo_num.h**
- common/misc/ **modulo_num.cc**

6.202 TraceManager::Monitor Class Reference

Inheritance diagram for TraceManager::Monitor:

Public Member Functions

- **Monitor** (**TraceManager** ∗manager)
- ∼**Monitor** ()
- void **spawn** ()

Private Member Functions

- void **run** ()

Private Attributes

- **_Thread** ∗ **m_thread**
- **TraceManager** ∗ **m_manager**

Additional Inherited Members

6.202.1 Detailed Description

Definition at line 16 of file trace_manager.h.

6.202.2 Constructor & Destructor Documentation

6.202.2.1 Monitor()

```
TraceManager::Monitor::Monitor (
              TraceManager * manager )
```

Definition at line 333 of file trace_manager.cc.

6.202.2.2 ∼Monitor()

```
TraceManager::Monitor::~Monitor ( )
```

Definition at line 338 of file trace_manager.cc.

6.202.3 Member Function Documentation

6.202.3.1 run()

```
void TraceManager::Monitor::run ( )  [private], [virtual]
```

Implements **Runnable** (p. 1106).

Definition at line 343 of file trace_manager.cc.

References n.

6.202.3.2 spawn()

```
void TraceManager::Monitor::spawn ( )
```

Definition at line 370 of file trace_manager.cc.

References _Thread::create().

Referenced by TraceManager::start().

6.202.4 Member Data Documentation

6.202.4.1 m_manager

```
TraceManager* TraceManager::Monitor::m_manager  [private]
```

Definition at line 21 of file trace_manager.h.

6.202.4.2 m_thread

```
_Thread* TraceManager::Monitor::m_thread  [private]
```

Definition at line 20 of file trace_manager.h.

The documentation for this class was generated from the following files:

- common/trace_frontend/ **trace_manager.h**
- common/trace_frontend/ **trace_manager.cc**

6.203 MovingArithmeticMean< T > Class Template Reference

`#include <moving_average.h>`

Inheritance diagram for MovingArithmeticMean< T >:

Public Member Functions

- **MovingArithmeticMean** (**UInt32** max_window_size)
- void **update** (T next_num)
- T **compute** () const

Private Attributes

- T **sum**

Additional Inherited Members

6.203.1 Detailed Description

template<class T>
class MovingArithmeticMean< T >

Definition at line 90 of file moving_average.h.

6.203.2 Constructor & Destructor Documentation

6.203.2.1 MovingArithmeticMean()

```
template<class T >
MovingArithmeticMean< T >::  MovingArithmeticMean (
              UInt32 max_window_size )  [inline]
```

Definition at line 96 of file moving_average.h.

6.203.3 Member Function Documentation

6.203.3.1 compute()

```
template<class T >
T MovingArithmeticMean< T >::compute ( ) const [inline], [virtual]
```

Implements **MovingAverage< T >** (p. 807).

Definition at line 115 of file moving_average.h.

References MovingAverage< T >::m_curr_window_back, MovingAverage< T >::m_curr_window_front, and MovingArithmeticMean< T >::sum.

6.203.3.2 update()

```
template<class T >
void MovingArithmeticMean< T >::update (
            T next_num ) [inline], [virtual]
```

Implements **MovingAverage< T >** (p. 809).

Definition at line 101 of file moving_average.h.

References MovingAverage< T >::addToWindow(), ModuloNum::getValue(), MovingAverage< T >::m_curr_window_back, MovingAverage< T >::m_curr_window_front, MovingAverage< T >::m_max_window_size, MovingAverage< T >::m_num_list, and MovingArithmeticMean< T >::sum.

6.203.4 Member Data Documentation

6.203.4.1 sum

```
template<class T >
T MovingArithmeticMean< T >::sum [private]
```

Definition at line 93 of file moving_average.h.

Referenced by MovingArithmeticMean< T >::compute(), and MovingArithmeticMean< T >::update().

The documentation for this class was generated from the following file:

- common/misc/ **moving_average.h**

6.204 MovingAverage< T > Class Template Reference

`#include <moving_average.h>`

Inheritance diagram for MovingAverage< T >:

Public Types

- enum **AvgType_t** { **ARITHMETIC_MEAN** = 0, **GEOMETRIC_MEAN**, **MEDIAN**, **NUM_AVG_TYPES** }

Public Member Functions

- **MovingAverage** (**UInt32** max_window_size)
- virtual ~**MovingAverage** ()
- virtual void **update** (T next_num)=0
- virtual T **compute** () const =0
- virtual T **compute** (T next_num)

Static Public Member Functions

- static **AvgType_t** **parseAvgType** (String avg_type)
- static **MovingAverage**< T > * **createAvgType** (**AvgType_t** avg_type, **UInt32** max_window_size)

Protected Member Functions

- void **addToWindow** (T next_num)
- void **printElements** ()

Protected Attributes

- std::vector< T > **m_num_list**
- **UInt32** **m_max_window_size**
- **ModuloNum** **m_curr_window_front**
- **ModuloNum** **m_curr_window_back**

6.204.1 Detailed Description

template<**class T**>
class MovingAverage< T >

Definition at line 12 of file moving_average.h.

6.204.2 Member Enumeration Documentation

6.204.2.1 AvgType_t

```
template<class T >
enum MovingAverage::AvgType_t
```

Enumerator

ARITHMETIC_MEAN	
GEOMETRIC_MEAN	
MEDIAN	
NUM_AVG_TYPES	

Definition at line 15 of file moving_average.h.

6.204.3 Constructor & Destructor Documentation

6.204.3.1 MovingAverage()

```
template<class T >
MovingAverage< T >::  MovingAverage (
              UInt32 max_window_size )
```

Definition at line 66 of file moving_average.h.

6.204.3.2 ~MovingAverage()

```
template<class T >
MovingAverage< T >::~ MovingAverage  [virtual]
```

Definition at line 75 of file moving_average.h.

6.204.4 Member Function Documentation

6.204.4.1 addToWindow()

```
template<class T >
void  MovingAverage< T >::addToWindow (
             T next_num )  [protected]
```

Definition at line 79 of file moving_average.h.

Referenced by MovingArithmeticMean< T >::update(), MovingGeometricMean< T >::update(), and Moving↩ Median< T >::update().

6.204.4.2 compute() [1/2]

```
template<class T >
virtual T  MovingAverage< T >::compute ( ) const  [pure virtual]
```

Implemented in **MovingMedian< T >** (p. 812), **MovingGeometricMean< T >** (p. 811), and **MovingArithmeticMean< T >** (p. 803).

Referenced by QueueModelBasic::computeQueueDelay(), and QueueModelHistoryList::computeUsingAnalyticalModel().

6.204.4.3 compute() [2/2]

```
template<class T >
virtual T  MovingAverage< T >::compute (
            T next_num )  [inline], [virtual]
```

Definition at line 45 of file moving_average.h.

Referenced by MovingAverage< SubsecondTime >::compute().

6.204.4.4 createAvgType()

```
template<class T >
MovingAverage< T > *  MovingAverage< T >::createAvgType (
              AvgType_t avg_type,
              UInt32 max_window_size )  [static]
```

Definition at line 174 of file moving_average.h.

Referenced by QueueModelBasic::QueueModelBasic(), and QueueModelHistoryList::QueueModelHistoryList().

6.204.4.5 parseAvgType()

```
template<class T >
static  AvgType_t  MovingAverage< T >::parseAvgType (
             String avg_type )  [inline], [static]
```

Definition at line 26 of file moving_average.h.

Referenced by QueueModelBasic::QueueModelBasic().

6.204.4.6 printElements()

```
template<class T >
void MovingAverage< T >::printElements ( ) [inline], [protected]
```

Definition at line 55 of file moving_average.h.

6.204.4.7 update()

```
template<class T >
virtual void MovingAverage< T >::update (
            T next_num ) [pure virtual]
```

Implemented in **MovingMedian< T >** (p. 813), **MovingGeometricMean< T >** (p. 811), and **MovingArithmeticMean< T >** (p. 804).

Referenced by MovingAverage< SubsecondTime >::compute(), and QueueModelHistoryList::updateAverageDelay().

6.204.5 Member Data Documentation

6.204.5.1 m_curr_window_back

```
template<class T >
ModuloNum MovingAverage< T >::m_curr_window_back [protected]
```

Definition at line 51 of file moving_average.h.

Referenced by MovingArithmeticMean< T >::compute(), MovingGeometricMean< T >::compute(), MovingMedian< T >::compute(), MovingAverage< SubsecondTime >::MovingAverage(), MovingAverage< SubsecondTime >::printElements(), and MovingArithmeticMean< T >::update().

6.204.5.2 m_curr_window_front

```
template<class T >
ModuloNum MovingAverage< T >::m_curr_window_front [protected]
```

Definition at line 50 of file moving_average.h.

Referenced by MovingArithmeticMean< T >::compute(), MovingGeometricMean< T >::compute(), MovingMedian< T >::compute(), MovingAverage< SubsecondTime >::MovingAverage(), MovingAverage< SubsecondTime >::printElements(), and MovingArithmeticMean< T >::update().

6.204.5.3 m_max_window_size

```
template<class T >
UInt32 MovingAverage< T >::m_max_window_size [protected]
```

Definition at line 49 of file moving_average.h.

Referenced by MovingAverage< SubsecondTime >::MovingAverage(), and MovingArithmeticMean< T >::update().

6.204.5.4 m_num_list

```
template<class T >
std::vector<T> MovingAverage< T >::m_num_list [protected]
```

Definition at line 48 of file moving_average.h.

Referenced by MovingGeometricMean< T >::compute(), MovingMedian< T >::compute(), MovingAverage< SubsecondTime >::MovingAverage(), MovingAverage< SubsecondTime >::printElements(), and MovingArithmeticMean< T >::update().

The documentation for this class was generated from the following file:

- common/misc/ **moving_average.h**

6.205 MovingGeometricMean< T > Class Template Reference

```
#include <moving_average.h>
```

Inheritance diagram for MovingGeometricMean< T >:

Public Member Functions

- **MovingGeometricMean** (**UInt32** max_window_size)
- void **update** (T next_num)
- T **compute** () const

Additional Inherited Members

6.205.1 Detailed Description

template<class T>
class MovingGeometricMean< T >

Definition at line 123 of file moving_average.h.

6.205.2 Constructor & Destructor Documentation

6.205.2.1 MovingGeometricMean()

```
template<class T >
MovingGeometricMean< T >::  MovingGeometricMean (
              UInt32 max_window_size )  [inline]
```

Definition at line 126 of file moving_average.h.

6.205.3 Member Function Documentation

6.205.3.1 compute()

```
template<class T >
T  MovingGeometricMean< T >::compute ( ) const  [inline], [virtual]
```

Implements **MovingAverage< T >** (p. 807).

Definition at line 135 of file moving_average.h.

References ModuloNum::getValue(), MovingAverage< T >::m_curr_window_back, MovingAverage< T >::m_curr_window_front, MovingAverage< T >::m_num_list, SubsecondTime::setInternalDataForced(), and SubsecondTime::Zero().

6.205.3.2 update()

```
template<class T >
void  MovingGeometricMean< T >::update (
             T next_num )  [inline], [virtual]
```

Implements **MovingAverage< T >** (p. 809).

Definition at line 130 of file moving_average.h.

References MovingAverage< T >::addToWindow().

The documentation for this class was generated from the following file:

- common/misc/ **moving_average.h**

6.206 MovingMedian< T > Class Template Reference

`#include <moving_average.h>`

Inheritance diagram for MovingMedian< T >:

Public Member Functions

- **MovingMedian** (**UInt32** max_window_size)
- void **update** (T next_num)
- T **compute** () const

Additional Inherited Members

6.206.1 Detailed Description

template<class T>
class MovingMedian< T >

Definition at line 153 of file moving_average.h.

6.206.2 Constructor & Destructor Documentation

6.206.2.1 MovingMedian()

```
template<class T >
MovingMedian< T >::  MovingMedian (
              UInt32 max_window_size )  [inline]
```

Definition at line 156 of file moving_average.h.

6.206.3 Member Function Documentation

6.206.3.1 compute()

```
template<class T >
T MovingMedian< T >::compute ( ) const [inline], [virtual]
```

Implements **MovingAverage< T >** (p. 807).

Definition at line 165 of file moving_average.h.

References MovingAverage< T >::m_curr_window_back, MovingAverage< T >::m_curr_window_front, and MovingAverage< T >::m_num_list.

6.206.3.2 update()

```
template<class T >
void MovingMedian< T >::update (
            T next_num ) [inline], [virtual]
```

Implements **MovingAverage< T >** (p. 809).

Definition at line 160 of file moving_average.h.

References MovingAverage< T >::addToWindow().

The documentation for this class was generated from the following file:

- common/misc/ **moving_average.h**

6.207 ParametricDramDirectoryMSI::MshrEntry Struct Reference

```
#include <cache_cntlr.h>
```

Public Attributes

- **SubsecondTime t_issue**
- **SubsecondTime t_complete**

6.207.1 Detailed Description

Definition at line 144 of file cache_cntlr.h.

6.207.2 Member Data Documentation

6.207.2.1 t_complete

```
SubsecondTime ParametricDramDirectoryMSI::MshrEntry::t_complete
```

Definition at line 145 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::make_mshr().

6.207.2.2 t_issue

```
SubsecondTime ParametricDramDirectoryMSI::MshrEntry::t_issue
```

Definition at line 145 of file cache_cntlr.h.

Referenced by ParametricDramDirectoryMSI::make_mshr().

The documentation for this struct was generated from the following file:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_cntlr.h**

6.208 MTCircularQueue< T > Class Template Reference

`#include <mt_circular_queue.h>`

Inheritance diagram for MTCircularQueue< T >:

Classes

- class **iterator**

Public Member Functions

- **MTCircularQueue** (**UInt32** **size**=64)
- void **push** (const T &t)
- void **push_wait** (const T &t)
- void **push_locked** (const T &t)
- T **pop** (void)
- T **pop_wait** (void)
- T **pop_locked** (void)
- void **full_wait** (void)
- void **empty_wait** (void)
- void **full_wait_locked** (void)
- void **empty_wait_locked** (void)

Private Attributes

- **Lock m_lock**
- **ConditionVariable m_full**
- **ConditionVariable m_empty**

Additional Inherited Members

6.208.1 Detailed Description

template<class T>
class MTCircularQueue< T >

Definition at line 8 of file mt_circular_queue.h.

6.208.2 Constructor & Destructor Documentation

6.208.2.1 MTCircularQueue()

```
template<class T >
MTCircularQueue< T >::  MTCircularQueue (
              UInt32 size = 64 )  [inline]
```

Definition at line 20 of file mt_circular_queue.h.

6.208.3 Member Function Documentation

6.208.3.1 empty_wait()

```
template<class T >
void  MTCircularQueue< T >::empty_wait (
             void  )
```

Definition at line 51 of file mt_circular_queue.h.

6.208.3.2 empty_wait_locked()

```
template<class T >
void  MTCircularQueue< T >::empty_wait_locked (
             void  )
```

Definition at line 59 of file mt_circular_queue.h.

6.208.3.3 full_wait()

```
template<class T >
void  MTCircularQueue< T >::full_wait (
            void  )
```

Definition at line 35 of file mt_circular_queue.h.

6.208.3.4 full_wait_locked()

```
template<class T >
void  MTCircularQueue< T >::full_wait_locked (
            void  )
```

Definition at line 43 of file mt_circular_queue.h.

6.208.3.5 pop()

```
template<class T >
T  MTCircularQueue< T >::pop (
            void  )
```

Definition at line 114 of file mt_circular_queue.h.

6.208.3.6 pop_locked()

```
template<class T >
T  MTCircularQueue< T >::pop_locked (
            void  )
```

Definition at line 100 of file mt_circular_queue.h.

References CircularQueue< T >::full(), and CircularQueue< T >::pop().

6.208.3.7 pop_wait()

```
template<class T >
T  MTCircularQueue< T >::pop_wait (
            void  )
```

Definition at line 122 of file mt_circular_queue.h.

6.208.3.8 push()

```
template<class T >
void  MTCircularQueue< T >::push (
            const T & t )
```

Definition at line 81 of file mt_circular_queue.h.

6.208.3.9 push_locked()

```
template<class T >
void  MTCircularQueue< T >::push_locked (
            const T & t )
```

Definition at line 69 of file mt_circular_queue.h.

References CircularQueue< T >::empty(), and CircularQueue< T >::push().

6.208.3.10 push_wait()

```
template<class T >
void  MTCircularQueue< T >::push_wait (
            const T & t )
```

Definition at line 89 of file mt_circular_queue.h.

6.208.4 Member Data Documentation

6.208.4.1 m_empty

```
template<class T >
ConditionVariable  MTCircularQueue< T >::m_empty  [private]
```

Definition at line 13 of file mt_circular_queue.h.

6.208.4.2 m_full

```
template<class T >
ConditionVariable  MTCircularQueue< T >::m_full  [private]
```

Definition at line 12 of file mt_circular_queue.h.

6.208.4.3 m_lock

```
template<class T >
Lock MTCircularQueue< T >::m_lock [private]
```

Definition at line 11 of file mt_circular_queue.h.

The documentation for this class was generated from the following file:

- common/misc/ **mt_circular_queue.h**

6.209 config::config_parser::Name Struct Reference

`#include <config_file_grammar.hpp>`

Public Member Functions

- **Name** (**RuleID** e_)
- template<typename Node , typename It >
  void **operator()** (Node & **n**, It, It) const

Public Attributes

- **RuleID e**

6.209.1 Detailed Description

Definition at line 51 of file config_file_grammar.hpp.

6.209.2 Constructor & Destructor Documentation

6.209.2.1 Name()

```
config::config_parser::Name::Name (
            RuleID e_ ) [inline]
```

Definition at line 52 of file config_file_grammar.hpp.

6.209.3 Member Function Documentation

6.209.3.1 operator()()

```
template<typename Node , typename It >
void config::config_parser::Name::operator() (
             Node & n,
             It ,
             It  ) const  [inline]
```

Definition at line 55 of file config_file_grammar.hpp.

References n.

6.209.4 Member Data Documentation

6.209.4.1 e

```
RuleID config::config_parser::Name::e
```

Definition at line 58 of file config_file_grammar.hpp.

The documentation for this struct was generated from the following file:

- common/config/ **config_file_grammar.hpp**

6.210 NetMatch Class Reference

```
#include <network.h>
```

Public Attributes

- std::vector< **SInt32** > **senders**
- std::vector< **PacketType** > **types**

6.210.1 Detailed Description

Definition at line 50 of file network.h.

6.210.2 Member Data Documentation

6.210.2.1 senders

`std::vector< SInt32> NetMatch::senders`

Definition at line 53 of file network.h.

Referenced by Network::netRecv(), and Network::netRecvFrom().

6.210.2.2 types

`std::vector< PacketType> NetMatch::types`

Definition at line 54 of file network.h.

Referenced by Network::netRecv(), and Network::netRecvType().

The documentation for this class was generated from the following file:

- common/network/ **network.h**

6.211 NetPacket Class Reference

`#include <network.h>`

Public Member Functions

- **NetPacket** ()
- **NetPacket** (**Byte** ∗)
- **NetPacket** (**SubsecondTime time**, **PacketType type**, **SInt32 sender**, **SInt32 receiver**, **UInt32 length**, const void ∗ **data**)
- **UInt32 bufferSize** () const
- **Byte** ∗ **makeBuffer** () const

Public Attributes

- **subsecond_time_t start_time**
- **subsecond_time_t time**
- **subsecond_time_t queue_delay**
- **PacketType type**
- **SInt32 sender**
- **SInt32 receiver**
- **UInt32 length**
- const void ∗ **data**

Static Public Attributes

- static const **SInt32 BROADCAST** = 0xDEADBABE

6.211.1 Detailed Description

Definition at line 23 of file network.h.

6.211.2 Constructor & Destructor Documentation

6.211.2.1 NetPacket() [1/3]

```
NetPacket::NetPacket ( )
```

Definition at line 482 of file network.cc.

6.211.2.2 NetPacket() [2/3]

```
NetPacket::NetPacket (
              Byte * buffer )  [explicit]
```

Definition at line 508 of file network.cc.

References data, and length.

6.211.2.3 NetPacket() [3/3]

```
NetPacket::NetPacket (
              SubsecondTime time,
              PacketType type,
              SInt32 sender,
              SInt32 receiver,
              UInt32 length,
             const void * data )
```

Definition at line 494 of file network.cc.

6.211.3 Member Function Documentation

6.211.3.1 bufferSize()

```
UInt32 NetPacket::bufferSize ( ) const
```

Definition at line 526 of file network.cc.

References length.

Referenced by Network::getModeledLength(), makeBuffer(), Network::netSend(), and SimThreadManager::quit↩SimThreads().

6.211.3.2 makeBuffer()

```
Byte * NetPacket::makeBuffer ( ) const
```

Definition at line 531 of file network.cc.

References bufferSize(), data, and length.

Referenced by Network::netSend().

6.211.4 Member Data Documentation

6.211.4.1 BROADCAST

```
const SInt32 NetPacket::BROADCAST = 0xDEADBABE [static]
```

Definition at line 43 of file network.h.

Referenced by ParametricDramDirectoryMSI::MemoryManager::broadcastMsg(), NetworkModel::countPacket(), Network::netBroadcast(), Network::netPullFromTransport(), Network::netRecv(), Network::netSend(), Network↩ModelEMeshHopCounter::routePacket(), NetworkModelMagic::routePacket(), NetworkModelBus::routePacket(), and NetworkModelEMeshHopByHop::routePacket().

6.211.4.2 data

```
const void* NetPacket::data
```

Definition at line 33 of file network.h.

Referenced by NetworkModelBus::accountPacket(), Network::getModeledLength(), ParametricDramDirectory↩MSI::MemoryManager::handleMsgFromNetwork(), makeBuffer(), NetPacket(), Network::netPullFromTransport(), Network::netSend(), NetworkModelEMeshHopCounter::processReceivedPacket(), NetworkModelMagic::process↩ReceivedPacket(), NetworkModelEMeshHopByHop::processReceivedPacket(), and NetworkModelEMeshHopBy↩Hop::routePacket().

6.211.4.3 length

UInt32 NetPacket::length

Definition at line 32 of file network.h.

Referenced by bufferSize(), makeBuffer(), NetPacket(), Network::netPullFromTransport(), Network::netRecv(), Network::netSend(), and NetworkModelBus::routePacket().

6.211.4.4 queue_delay

subsecond_time_t NetPacket::queue_delay

Definition at line 28 of file network.h.

Referenced by NetworkModelEMeshHopByHop::processReceivedPacket(), NetworkModelBus::routePacket(), NetworkModelEMeshHopByHop::routePacket(), and ShmemPerf::updatePacket().

6.211.4.5 receiver

SInt32 NetPacket::receiver

Definition at line 31 of file network.h.

Referenced by NetworkModelBus::accountPacket(), NetworkModel::countPacket(), ParametricDramDirectory↩MSI::MemoryManager::handleMsgFromNetwork(), Network::netPullFromTransport(), Network::netSend(), Sim↩ThreadManager::quitSimThreads(), NetworkModelEMeshHopCounter::routePacket(), NetworkModelMagic↩::routePacket(), NetworkModelBus::routePacket(), and NetworkModelEMeshHopByHop::routePacket().

6.211.4.6 sender

SInt32 NetPacket::sender

Definition at line 30 of file network.h.

Referenced by NetworkModelBus::accountPacket(), ParametricDramDirectoryMSI::MemoryManager::handle↩MsgFromNetwork(), Network::netPullFromTransport(), Network::netRecv(), Network::netSend(), NetworkModel↩EMeshHopCounter::processReceivedPacket(), NetworkModelMagic::processReceivedPacket(), NetworkModel↩EMeshHopByHop::processReceivedPacket(), NetworkModelEMeshHopCounter::routePacket(), and Network↩ModelEMeshHopByHop::routePacket().

6.211.4.7 start_time

subsecond_time_t NetPacket::start_time

Definition at line 26 of file network.h.

Referenced by Network::netSend(), NetworkModelEMeshHopCounter::processReceivedPacket(), and NetworkModelEMeshHopByHop::processReceivedPacket().

6.211.4.8 time

subsecond_time_t NetPacket::time

Definition at line 27 of file network.h.

Referenced by ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork(), Network::netPullFromTransport(), Network::netRecv(), Network::netSend(), NetworkModelEMeshHopCounter::processReceivedPacket(), NetworkModelEMeshHopByHop::processReceivedPacket(), NetworkModelEMeshHopCounter::routePacket(), NetworkModelMagic::routePacket(), NetworkModelBus::routePacket(), NetworkModelEMeshHopByHop::routePacket(), and ShmemPerf::updatePacket().

6.211.4.9 type

PacketType NetPacket::type

Definition at line 29 of file network.h.

Referenced by NetworkModelBus::accountPacket(), Network::getModeledLength(), MemoryManagerNetworkCallback(), Network::netPullFromTransport(), Network::netRecv(), Network::netSend(), NetworkModelEMeshHopCounter::processReceivedPacket(), NetworkModelMagic::processReceivedPacket(), NetworkModelEMeshHopByHop::processReceivedPacket(), and NetworkModelEMeshHopByHop::routePacket().

The documentation for this class was generated from the following files:

- common/network/ **network.h**
- common/network/ **network.cc**

6.212 NetRecvIterator Class Reference

Public Member Functions

- **NetRecvIterator** (**UInt32** i)
- **NetRecvIterator** (const std::vector< **SInt32** > &v)
- **NetRecvIterator** (const std::vector< **PacketType** > &v)
- **UInt32 get** ()
- **Boolean done** ()
- void **next** ()
- void **reset** ()

Private Types

- enum { **INT**, **SENDER_VECTOR**, **TYPE_VECTOR** }

Private Attributes

- enum NetRecvIterator:: { ... } **_mode**
- union {
  UInt32 **_max**
  const std::vector< **SInt32** > * **_senders**
  const std::vector< **PacketType** > * **_types**
  };
- **UInt32** **_i**

6.212.1 Detailed Description

Definition at line 211 of file network.cc.

6.212.2 Member Enumeration Documentation

6.212.2.1 anonymous enum

```
anonymous enum  [private]
```

Enumerator

INT	
SENDER_VECTOR	
TYPE_VECTOR	

Definition at line 276 of file network.cc.

6.212.3 Constructor & Destructor Documentation

6.212.3.1 NetRecvIterator() [1/3]

```
NetRecvIterator::NetRecvIterator (
              UInt32 i )  [inline]
```

Definition at line 214 of file network.cc.

6.212.3.2 NetRecvIterator() [2/3]

```
NetRecvIterator::NetRecvIterator (
            const std::vector<  SInt32 > & v )  [inline]
```

Definition at line 220 of file network.cc.

6.212.3.3 NetRecvIterator() [3/3]

```
NetRecvIterator::NetRecvIterator (
            const std::vector<  PacketType > & v )  [inline]
```

Definition at line 226 of file network.cc.

6.212.4 Member Function Documentation

6.212.4.1 done()

```
Boolean NetRecvIterator::done ( )  [inline]
```

Definition at line 249 of file network.cc.

References _i, _max, _mode, _senders, _types, INT, SENDER_VECTOR, and TYPE_VECTOR.

Referenced by Network::netRecv().

6.212.4.2 get()

```
UInt32 NetRecvIterator::get ( )  [inline]
```

Definition at line 233 of file network.cc.

References _i, _mode, _senders, _types, INT, SENDER_VECTOR, and TYPE_VECTOR.

Referenced by Network::netRecv().

6.212.4.3 next()

```
void NetRecvIterator::next ( )  [inline]
```

Definition at line 265 of file network.cc.

References _i.

Referenced by Network::netRecv().

6.212.4.4 reset()

```
void NetRecvIterator::reset ( )  [inline]
```

Definition at line 270 of file network.cc.

References _i.

Referenced by Network::netRecv().

6.212.5 Member Data Documentation

6.212.5.1 "@4

```
union { ... }  [private]
```

6.212.5.2 _i

```
UInt32 NetRecvIterator::_i  [private]
```

Definition at line 288 of file network.cc.

Referenced by done(), get(), next(), and reset().

6.212.5.3 _max

```
UInt32 NetRecvIterator::_max
```

Definition at line 283 of file network.cc.

Referenced by done().

6.212.5.4 _mode

```
enum { ... } NetRecvIterator::_mode [private]
```

Referenced by done(), and get().

6.212.5.5 _senders

```
const std::vector< SInt32>* NetRecvIterator::_senders
```

Definition at line 284 of file network.cc.

Referenced by done(), and get().

6.212.5.6 _types

```
const std::vector< PacketType>* NetRecvIterator::_types
```

Definition at line 285 of file network.cc.

Referenced by done(), and get().

The documentation for this class was generated from the following file:

- common/network/ **network.cc**

6.213 Network Class Reference

```
#include <network.h>
```

Public Types

- typedef void(* **NetworkCallback**) (void *, **NetPacket**)

Public Member Functions

- **Network** (**Core** ∗core)
- ∼**Network** ()
- **Core** ∗ **getCore** () const
- **Transport::Node** ∗ **getTransport** () const
- void **registerCallback** (**PacketType** type, **NetworkCallback** callback, void ∗obj)
- void **unregisterCallback** (**PacketType** type)
- void **netPullFromTransport** ()
- **SInt32 netSend** (**NetPacket** &packet)
- **NetPacket netRecv** (const **NetMatch** &match, **UInt64** timeout_ns=0)
- **SInt32 netSend** (**SInt32** dest, **PacketType** type, const void ∗buf, **UInt32** len)
- **SInt32 netBroadcast** (**PacketType** type, const void ∗buf, **UInt32** len)
- **NetPacket netRecv** (**SInt32 src**, **PacketType** type, **UInt64** timeout_ns=0)
- **NetPacket netRecvFrom** (**SInt32 src**, **UInt64** timeout_ns=0)
- **NetPacket netRecvType** (**PacketType** type, **UInt64** timeout_ns=0)
- void **enableModels** ()
- void **disableModels** ()
- **NetworkModel** ∗ **getNetworkModelFromPacketType** (**PacketType** packet_type)
- **UInt32 getModeledLength** (const **NetPacket** &pkt)

Private Member Functions

- void **forwardPacket** (**NetPacket** &packet)

Private Attributes

- **NetworkModel** ∗ **_models** [**NUM_STATIC_NETWORKS**]
- **NetworkCallback** ∗ **_callbacks**
- void ∗∗ **_callbackObjs**
- **Core** ∗ **_core**
- **Transport::Node** ∗ **_transport**
- **SInt32 _tid**
- **SInt32 _numMod**
- **NetQueue _netQueue**
- **Lock _netQueueLock**
- **ConditionVariable _netQueueCond**

6.213.1 Detailed Description

Definition at line 62 of file network.h.

6.213.2 Member Typedef Documentation

6.213.2.1 NetworkCallback

```
typedef void(* Network::NetworkCallback) (void *,  NetPacket)
```

Definition at line 73 of file network.h.

6.213.3 Constructor & Destructor Documentation

6.213.3.1 Network()

```
Network::Network (
            Core * core )
```

Definition at line 20 of file network.cc.

References _callbackObjs, _callbacks, _core, _models, _numMod, _tid, _transport, NetworkModel::createModel(), Transport::createNode(), Core::getId(), Config::getNetworkModels(), Transport::getSingleton(), Config::getSingleton(), Config::getTotalCores(), LOG_ASSERT_ERROR, LOG_PRINT, NUM_PACKET_TYPES, and NUM_STATIC_NETWORKS.

6.213.3.2 ~Network()

```
Network::~Network ( )
```

Definition at line 45 of file network.cc.

References _callbackObjs, _callbacks, _models, _transport, LOG_PRINT, and NUM_STATIC_NETWORKS.

6.213.4 Member Function Documentation

6.213.4.1 disableModels()

```
void Network::disableModels ( )
```

Definition at line 453 of file network.cc.

References _models, NetworkModel::disable(), and NUM_STATIC_NETWORKS.

Referenced by Core::disablePerformanceModels().

6.213.4.2 enableModels()

```
void Network::enableModels ( )
```

Definition at line 445 of file network.cc.

References _models, NetworkModel::enable(), and NUM_STATIC_NETWORKS.

Referenced by Core::enablePerformanceModels().

6.213.4.3 forwardPacket()

```
void Network::forwardPacket (
              NetPacket & packet )  [private]
```

Definition at line 140 of file network.cc.

References netSend().

Referenced by netPullFromTransport().

6.213.4.4 getCore()

```
Core* Network::getCore ( ) const  [inline]
```

Definition at line 70 of file network.h.

References _core.

Referenced by NetworkModelBus::accountPacket(), getModeledLength(), NetworkModel::NetworkModel(), NetworkModelEMeshHopCounter::NetworkModelEMeshHopCounter(), NetworkModelEMeshHopCounter↩ ::processReceivedPacket(), NetworkModelMagic::processReceivedPacket(), NetworkModelEMeshHopByHop↩ ::processReceivedPacket(), and NetworkModelEMeshHopByHop::routePacket().

6.213.4.5 getModeledLength()

```
UInt32 Network::getModeledLength (
            const  NetPacket & pkt )
```

Definition at line 462 of file network.cc.

References NetPacket::bufferSize(), NetPacket::data, getCore(), Config::getCoreIDLength(), Core::getMemory↩ Manager(), MemoryManagerBase::getModeledLength(), Config::getSingleton(), SHARED_MEM_1, and Net↩ Packet::type.

Referenced by NetworkModel::countPacket(), NetworkModelEMeshHopCounter::processReceivedPacket(), NetworkModelMagic::processReceivedPacket(), NetworkModelEMeshHopByHop::processReceivedPacket(), NetworkModelEMeshHopCounter::routePacket(), NetworkModelBus::routePacket(), and NetworkModelEMesh↩ HopByHop::routePacket().

6.213.4.6 getNetworkModelFromPacketType()

```
NetworkModel * Network::getNetworkModelFromPacketType (
            PacketType packet_type )
```

Definition at line 145 of file network.cc.

References _models.

Referenced by netSend().

6.213.4.7 getTransport()

```
Transport::Node* Network::getTransport ( ) const  [inline]
```

Definition at line 71 of file network.h.

References _transport.

6.213.4.8 netBroadcast()

```
SInt32 Network::netBroadcast (
            PacketType type,
           const void * buf,
            UInt32 len )
```

Definition at line 418 of file network.cc.

References NetPacket::BROADCAST, and netSend().

6.213.4.9 netPullFromTransport()

```
void Network::netPullFromTransport ( )
```

Definition at line 75 of file network.cc.

References _callbackObjs, _callbacks, _core, _models, _netQueue, _netQueueCond, _netQueueLock, _num↩Mod, _transport, TLock< T_LockCreator >::acquire(), ConditionVariable::broadcast(), NetPacket::BROADCA↩ST, NetPacket::data, forwardPacket(), Core::getId(), itostr(), NetPacket::length, LOG_ASSERT_ERROR, LOG_↩PRINT, NUM_PACKET_TYPES, NetworkModel::processReceivedPacket(), Transport::Node::query(), NetPacket↩::receiver, Transport::Node::recv(), TLock< T_LockCreator >::release(), NetPacket::sender, NetPacket::time, and NetPacket::type.

Referenced by SimThread::run().

6.213.4.10 netRecv() [1/2]

```
NetPacket Network::netRecv (
            const  NetMatch & match,
             UInt64 timeout_ns = 0 )
```

Definition at line 291 of file network.cc.

References _core, _netQueue, _netQueueCond, _netQueueLock, _numMod, TLock< T_LockCreator >::acquire(), NetPacket::BROADCAST, NetRecvIterator::done(), NetRecvIterator::get(), PerformanceModel::getElapsedTime(), Core::getId(), Core::getPerformanceModel(), itostr(), NetPacket::length, LOG_ASSERT_ERROR, LOG_PRINT, NetRecvIterator::next(), SubsecondTime::NS(), NUM_PACKET_TYPES, PerformanceModel::queuePseudoInstruction(), TLock< T_LockCreator >::release(), NetRecvIterator::reset(), NetPacket::sender, NetMatch::senders, NetPacket::time, NetPacket::type, NetMatch::types, and ConditionVariable::wait().

Referenced by netRecv(), netRecvFrom(), and netRecvType().

6.213.4.11 netRecv() [2/2]

```
NetPacket Network::netRecv (
             SInt32 src,
             PacketType type,
             UInt64 timeout_ns = 0 )
```

Definition at line 423 of file network.cc.

References netRecv(), NetMatch::senders, src, and NetMatch::types.

6.213.4.12 netRecvFrom()

```
NetPacket Network::netRecvFrom (
             SInt32 src,
             UInt64 timeout_ns = 0 )
```

Definition at line 431 of file network.cc.

References netRecv(), NetMatch::senders, and src.

6.213.4.13 netRecvType()

```
NetPacket Network::netRecvType (
             PacketType type,
             UInt64 timeout_ns = 0 )
```

Definition at line 438 of file network.cc.

References netRecv(), and NetMatch::types.

6.213.4.14 netSend() [1/2]

```
SInt32 Network::netSend (
            NetPacket & packet )
```

Definition at line 150 of file network.cc.

References _core, _models, _transport, NetPacket::BROADCAST, NetPacket::bufferSize(), NetworkModel::countPacket(), Core::getId(), Core::getNetwork(), getNetworkModelFromPacketType(), itostr(), NetPacket::length, LOG_PRINT, NetPacket::makeBuffer(), NUM_PACKET_TYPES, NetPacket::receiver, NetworkModel::routePacket(), Transport::Node::send(), NetPacket::sender, NetPacket::start_time, NetPacket::time, and NetPacket::type.

Referenced by ParametricDramDirectoryMSI::MemoryManager::broadcastMsg(), forwardPacket(), netBroadcast(), netSend(), and ParametricDramDirectoryMSI::MemoryManager::sendMsg().

6.213.4.15 netSend() [2/2]

```
SInt32 Network::netSend (
            SInt32 dest,
            PacketType type,
            const void * buf,
            UInt32 len )
```

Definition at line 404 of file network.cc.

References _core, NetPacket::data, PerformanceModel::getElapsedTime(), Core::getId(), Core::getPerformanceModel(), NetPacket::length, netSend(), NetPacket::receiver, NetPacket::sender, NetPacket::time, and NetPacket::type.

6.213.4.16 registerCallback()

```
void Network::registerCallback (
            PacketType type,
            NetworkCallback callback,
            void * obj )
```

Definition at line 58 of file network.cc.

References _callbackObjs, _callbacks, and NUM_PACKET_TYPES.

Referenced by ParametricDramDirectoryMSI::MemoryManager::MemoryManager(), SimThread::run(), and CoreThread::run().

6.213.4.17 unregisterCallback()

```
void Network::unregisterCallback (
              PacketType type )
```

Definition at line 65 of file network.cc.

References _callbacks, and NUM_PACKET_TYPES.

Referenced by ParametricDramDirectoryMSI::MemoryManager::~MemoryManager().

6.213.5 Member Data Documentation

6.213.5.1 _callbackObjs

```
void** Network::_callbackObjs  [private]
```

Definition at line 109 of file network.h.

Referenced by netPullFromTransport(), Network(), registerCallback(), and ~Network().

6.213.5.2 _callbacks

```
NetworkCallback* Network::_callbacks  [private]
```

Definition at line 108 of file network.h.

Referenced by netPullFromTransport(), Network(), registerCallback(), unregisterCallback(), and ~Network().

6.213.5.3 _core

```
Core* Network::_core  [private]
```

Definition at line 111 of file network.h.

Referenced by getCore(), netPullFromTransport(), netRecv(), netSend(), and Network().

6.213.5.4 _models

NetworkModel* Network::_models[**NUM_STATIC_NETWORKS**] [private]

Definition at line 106 of file network.h.

Referenced by disableModels(), enableModels(), getNetworkModelFromPacketType(), netPullFromTransport(), netSend(), Network(), and ~Network().

6.213.5.5 _netQueue

NetQueue Network::_netQueue [private]

Definition at line 117 of file network.h.

Referenced by netPullFromTransport(), and netRecv().

6.213.5.6 _netQueueCond

ConditionVariable Network::_netQueueCond [private]

Definition at line 119 of file network.h.

Referenced by netPullFromTransport(), and netRecv().

6.213.5.7 _netQueueLock

Lock Network::_netQueueLock [private]

Definition at line 118 of file network.h.

Referenced by netPullFromTransport(), and netRecv().

6.213.5.8 _numMod

SInt32 Network::_numMod [private]

Definition at line 115 of file network.h.

Referenced by netPullFromTransport(), netRecv(), and Network().

6.213.5.9 _tid

```
SInt32 Network::_tid  [private]
```

Definition at line 114 of file network.h.

Referenced by Network().

6.213.5.10 _transport

```
Transport::Node* Network::_transport  [private]
```

Definition at line 112 of file network.h.

Referenced by getTransport(), netPullFromTransport(), netSend(), Network(), and ~Network().

The documentation for this class was generated from the following files:

- common/network/ **network.h**
- common/network/ **network.cc**

6.214 NetworkModel Class Reference

```
#include <network_model.h>
```

Inheritance diagram for NetworkModel:

Classes

- struct **Hop**

Public Member Functions

- **NetworkModel** (**Network** *network, **EStaticNetwork** net_type)
- virtual ~**NetworkModel** ()
- void **countPacket** (const **NetPacket** &packet)
- virtual void **routePacket** (const **NetPacket** &pkt, std::vector< **Hop** > &nextHops)=0
- virtual void **processReceivedPacket** (**NetPacket** &pkt)=0
- virtual void **enable** ()=0
- virtual void **disable** ()=0

Static Public Member Functions

- static **NetworkModel** * **createModel** (**Network** *network, **UInt32** model_type, **EStaticNetwork** net_type)
- static **UInt32** **parseNetworkType** (String str)
- static std::pair< bool, **SInt32** > **computeCoreCountConstraints** (**UInt32** network_type, **SInt32** core_↩ count)
- static std::pair< bool, std::vector< **core_id_t** > > **computeMemoryControllerPositions** (**UInt32** network_type, **SInt32** num_memory_controllers, **SInt32** total_cores)

Protected Member Functions

- **Network** * **getNetwork** ()

Private Attributes

- **Network** * **_network**
- const bool **m_collect_traffic_matrix**
- std::vector< uint64_t > **m_matrix_packets**
- std::vector< uint64_t > **m_matrix_bytes**

6.214.1 Detailed Description

Definition at line 25 of file network_model.h.

6.214.2 Constructor & Destructor Documentation

6.214.2.1 NetworkModel()

```
NetworkModel::NetworkModel (
              Network * network,
              EStaticNetwork net_type )
```

Definition at line 14 of file network_model.cc.

References _network, EStaticNetworkStrings, Network::getCore(), Core::getId(), itostr(), m_collect_traffic_matrix, m_matrix_bytes, m_matrix_packets, and registerStatsMetric().

6.214.2.2 ~NetworkModel()

```
virtual NetworkModel::~NetworkModel ( )  [inline], [virtual]
```

Definition at line 29 of file network_model.h.

6.214.3 Member Function Documentation

6.214.3.1 computeCoreCountConstraints()

```
std::pair< bool,  SInt32 > NetworkModel::computeCoreCountConstraints (
              UInt32 network_type,
              SInt32 core_count )  [static]
```

Definition at line 83 of file network_model.cc.

References NetworkModelEMeshHopByHop::computeCoreCountConstraints(), LOG_PRINT_ERROR, NETWO↩RK_BUS, NETWORK_EMESH_HOP_BY_HOP, NETWORK_EMESH_HOP_COUNTER, and NETWORK_MAGIC.

Referenced by Config::getNearestAcceptableCoreCount().

6.214.3.2 computeMemoryControllerPositions()

```
std::pair< bool, std::vector<  core_id_t > > NetworkModel::computeMemoryControllerPositions (
              UInt32 network_type,
              SInt32 num_memory_controllers,
              SInt32 total_cores )  [static]
```

Definition at line 102 of file network_model.cc.

References NetworkModelEMeshHopByHop::computeMemoryControllerPositions(), LOG_PRINT_ERROR, NET↩WORK_BUS, NETWORK_EMESH_HOP_BY_HOP, NETWORK_EMESH_HOP_COUNTER, and NETWORK_M↩AGIC.

Referenced by MemoryManagerBase::getCoreListWithMemoryControllers().

6.214.3.3 countPacket()

```
void NetworkModel::countPacket (
             const  NetPacket & packet )
```

Definition at line 34 of file network_model.cc.

References NetPacket::BROADCAST, Network::getModeledLength(), getNetwork(), m_collect_traffic_matrix, m_↩matrix_bytes, m_matrix_packets, and NetPacket::receiver.

Referenced by Network::netSend().

6.214.3.4 createModel()

```
NetworkModel * NetworkModel::createModel (
            Network * network,
            UInt32 model_type,
            EStaticNetwork net_type )  [static]
```

Definition at line 45 of file network_model.cc.

References NETWORK_BUS, NETWORK_EMESH_HOP_BY_HOP, NETWORK_EMESH_HOP_COUNTER, and NETWORK_MAGIC.

Referenced by Network::Network().

6.214.3.5 disable()

```
virtual void NetworkModel::disable ( )  [pure virtual]
```

Implemented in **NetworkModelEMeshHopByHop** (p. 855), **NetworkModelBus** (p. 844), **NetworkModelMagic** (p. 868), and **NetworkModelEMeshHopCounter** (p. 864).

Referenced by Network::disableModels().

6.214.3.6 enable()

```
virtual void NetworkModel::enable ( )  [pure virtual]
```

Implemented in **NetworkModelEMeshHopByHop** (p. 856), **NetworkModelBus** (p. 844), **NetworkModelMagic** (p. 869), and **NetworkModelEMeshHopCounter** (p. 864).

Referenced by Network::enableModels().

6.214.3.7 getNetwork()

```
Network* NetworkModel::getNetwork ( )  [inline], [protected]
```

Definition at line 54 of file network_model.h.

References _network.

Referenced by NetworkModelBus::accountPacket(), countPacket(), NetworkModelEMeshHopCounter::process↩ ReceivedPacket(), NetworkModelMagic::processReceivedPacket(), NetworkModelEMeshHopByHop::process↩ ReceivedPacket(), NetworkModelEMeshHopCounter::routePacket(), NetworkModelBus::routePacket(), and NetworkModelEMeshHopByHop::routePacket().

6.214.3.8 parseNetworkType()

```
UInt32 NetworkModel::parseNetworkType (
            String str )  [static]
```

Definition at line 68 of file network_model.cc.

References NETWORK_BUS, NETWORK_EMESH_HOP_BY_HOP, NETWORK_EMESH_HOP_COUNTER, and NETWORK_MAGIC.

Referenced by MemoryManagerBase::getCoreListWithMemoryControllers(), Config::getNearestAcceptableCore↩Count(), and Config::getNetworkModels().

6.214.3.9 processReceivedPacket()

```
virtual void NetworkModel::processReceivedPacket (
             NetPacket & pkt )  [pure virtual]
```

Implemented in **NetworkModelEMeshHopByHop** (p. 856), **NetworkModelBus** (p. 844), **NetworkModelMagic** (p. 869), and **NetworkModelEMeshHopCounter** (p. 865).

Referenced by Network::netPullFromTransport().

6.214.3.10 routePacket()

```
virtual void NetworkModel::routePacket (
            const  NetPacket & pkt,
            std::vector<  Hop > & nextHops )  [pure virtual]
```

Implemented in **NetworkModelEMeshHopByHop** (p. 857), **NetworkModelBus** (p. 845), **NetworkModelMagic** (p. 869), and **NetworkModelEMeshHopCounter** (p. 865).

Referenced by Network::netSend().

6.214.4 Member Data Documentation

6.214.4.1 _network

```
Network* NetworkModel::_network  [private]
```

Definition at line 57 of file network_model.h.

Referenced by getNetwork(), and NetworkModel().

6.214.4.2 m_collect_traffic_matrix

```
const bool NetworkModel::m_collect_traffic_matrix  [private]
```

Definition at line 59 of file network_model.h.

Referenced by countPacket(), and NetworkModel().

6.214.4.3 m_matrix_bytes

```
std::vector<uint64_t> NetworkModel::m_matrix_bytes  [private]
```

Definition at line 61 of file network_model.h.

Referenced by countPacket(), and NetworkModel().

6.214.4.4 m_matrix_packets

```
std::vector<uint64_t> NetworkModel::m_matrix_packets  [private]
```

Definition at line 60 of file network_model.h.

Referenced by countPacket(), and NetworkModel().

The documentation for this class was generated from the following files:

- common/network/ **network_model.h**
- common/network/ **network_model.cc**

6.215 NetworkModelBus Class Reference

```
#include <network_model_bus.h>
```

Inheritance diagram for NetworkModelBus:

Public Member Functions

- **NetworkModelBus** (**Network** ∗net, **EStaticNetwork** net_type)
- ∼**NetworkModelBus** ()
- void **routePacket** (const **NetPacket** &pkt, std::vector< **Hop** > &nextHops)
- void **processReceivedPacket** (**NetPacket** &pkt)
- void **enable** ()
- void **disable** ()

Private Member Functions

- bool **accountPacket** (const **NetPacket** &pkt)

Private Attributes

- bool **_enabled**
- **NetworkModelBusGlobal** ∗ **_bus**
- bool **_ignore_local**

Static Private Attributes

- static **NetworkModelBusGlobal** ∗ **_bus_global** [**NUM_STATIC_NETWORKS**] = { NULL }

Additional Inherited Members

6.215.1 Detailed Description

Definition at line 29 of file network_model_bus.h.

6.215.2 Constructor & Destructor Documentation

6.215.2.1 NetworkModelBus()

```
NetworkModelBus::NetworkModelBus (
              Network * net,
              EStaticNetwork net_type )
```

Definition at line 53 of file network_model_bus.cc.

References _bus, _bus_global, and EStaticNetworkStrings.

6.215.2.2 ~NetworkModelBus()

```
NetworkModelBus::~NetworkModelBus ( )  [inline]
```

Definition at line 41 of file network_model_bus.h.

6.215.3 Member Function Documentation

6.215.3.1 accountPacket()

```
bool NetworkModelBus::accountPacket (
            const  NetPacket & pkt )  [private]
```

Definition at line 108 of file network_model_bus.cc.

References _enabled, _ignore_local, NetPacket::data, Config::getApplicationCores(), Network::getCore(), Core::getMemoryManager(), NetworkModel::getNetwork(), MemoryManagerBase::getShmemRequester(), Config::getSingleton(), INVALID_CORE_ID, LOG_ASSERT_ERROR, NetPacket::receiver, NetPacket::sender, SHARED_MEM_1, and NetPacket::type.

Referenced by routePacket().

6.215.3.2 disable()

```
void NetworkModelBus::disable ( )  [inline], [virtual]
```

Implements **NetworkModel** (p. 840).

Definition at line 50 of file network_model_bus.h.

References _enabled.

6.215.3.3 enable()

```
void NetworkModelBus::enable ( )  [inline], [virtual]
```

Implements **NetworkModel** (p. 840).

Definition at line 47 of file network_model_bus.h.

References _enabled.

6.215.3.4 processReceivedPacket()

```
void NetworkModelBus::processReceivedPacket (
              NetPacket & pkt )  [virtual]
```

Implements **NetworkModel** (p. 841).

Definition at line 103 of file network_model_bus.cc.

6.215.3.5 routePacket()

```
void NetworkModelBus::routePacket (
             const  NetPacket & pkt,
             std::vector<  Hop > & nextHops )  [virtual]
```

Implements **NetworkModel** (p. 841).

Definition at line 66 of file network_model_bus.cc.

References _bus, NetworkModelBusGlobal::_lock, NetworkModelBusGlobal::_num_bytes, NetworkModelBus↩Global::_num_packets, accountPacket(), NetPacket::BROADCAST, NetworkModel::Hop::final_dest, Network::get↩ModeledLength(), NetworkModel::getNetwork(), Config::getSingleton(), Config::getTotalCores(), NetPacket::length, NetworkModel::Hop::next_dest, NetPacket::queue_delay, NetPacket::receiver, NetPacket::time, NetworkModel::↩Hop::time, and NetworkModelBusGlobal::useBus().

6.215.4 Member Data Documentation

6.215.4.1 _bus

```
NetworkModelBusGlobal* NetworkModelBus::_bus  [private]
```

Definition at line 35 of file network_model_bus.h.

Referenced by NetworkModelBus(), and routePacket().

6.215.4.2 _bus_global

```
NetworkModelBusGlobal * NetworkModelBus::_bus_global = { NULL }  [static], [private]
```

Definition at line 31 of file network_model_bus.h.

Referenced by NetworkModelBus().

6.215.4.3 _enabled

```
bool NetworkModelBus::_enabled  [private]
```

Definition at line 34 of file network_model_bus.h.

Referenced by accountPacket(), disable(), and enable().

6.215.4.4 _ignore_local

```
bool NetworkModelBus::_ignore_local  [private]
```

Definition at line 36 of file network_model_bus.h.

Referenced by accountPacket().

The documentation for this class was generated from the following files:

- common/network/ **network_model_bus.h**
- common/network/ **network_model_bus.cc**

6.216 NetworkModelBusGlobal Class Reference

```
#include <network_model_bus.h>
```

Public Member Functions

- **NetworkModelBusGlobal** (String name)
- ∼**NetworkModelBusGlobal** ()
- **SubsecondTime useBus** (**SubsecondTime t_start**, **UInt32** length, **subsecond_time_t** ∗queue_↩ delay_stats=NULL)

Public Attributes

- **Lock _lock**
- const **ComponentBandwidth _bandwidth**
- **QueueModel** ∗ **_queue_model**
- **UInt64 _num_packets**
- **UInt64 _num_packets_delayed**
- **UInt64 _num_bytes**
- **SubsecondTime _time_used**
- **SubsecondTime _total_delay**

6.216.1 Detailed Description

Definition at line 11 of file network_model_bus.h.

6.216.2 Constructor & Destructor Documentation

6.216.2.1 NetworkModelBusGlobal()

```
NetworkModelBusGlobal::NetworkModelBusGlobal (
            String name )
```

Definition at line 13 of file network_model_bus.cc.

References _num_bytes, _num_packets, _num_packets_delayed, _queue_model, _time_used, _total_delay, QueueModel::create(), ComponentPeriod::fromFreqHz(), and registerStatsMetric().

6.216.2.2 ~NetworkModelBusGlobal()

```
NetworkModelBusGlobal::~NetworkModelBusGlobal ( )
```

Definition at line 33 of file network_model_bus.cc.

References _queue_model.

6.216.3 Member Function Documentation

6.216.3.1 useBus()

```
SubsecondTime NetworkModelBusGlobal::useBus (
            SubsecondTime t_start,
            UInt32 length,
            subsecond_time_t * queue_delay_stats = NULL )
```

Definition at line 40 of file network_model_bus.cc.

References _bandwidth, _num_packets_delayed, _queue_model, _time_used, _total_delay, QueueModel::computeQueueDelay(), ComponentBandwidth::getLatency(), t_start, and SubsecondTime::Zero().

Referenced by NetworkModelBus::routePacket().

6.216.4 Member Data Documentation

6.216.4.1 _bandwidth

```
const  ComponentBandwidth NetworkModelBusGlobal::_bandwidth
```

Definition at line 15 of file network_model_bus.h.

Referenced by useBus().

6.216.4.2 _lock

```
Lock NetworkModelBusGlobal::_lock
```

Definition at line 14 of file network_model_bus.h.

Referenced by NetworkModelBus::routePacket().

6.216.4.3 _num_bytes

```
UInt64 NetworkModelBusGlobal::_num_bytes
```

Definition at line 20 of file network_model_bus.h.

Referenced by NetworkModelBusGlobal(), and NetworkModelBus::routePacket().

6.216.4.4 _num_packets

```
UInt64 NetworkModelBusGlobal::_num_packets
```

Definition at line 18 of file network_model_bus.h.

Referenced by NetworkModelBusGlobal(), and NetworkModelBus::routePacket().

6.216.4.5 _num_packets_delayed

```
UInt64 NetworkModelBusGlobal::_num_packets_delayed
```

Definition at line 19 of file network_model_bus.h.

Referenced by NetworkModelBusGlobal(), and useBus().

6.216.4.6 _queue_model

```
QueueModel* NetworkModelBusGlobal::_queue_model
```

Definition at line 16 of file network_model_bus.h.

Referenced by NetworkModelBusGlobal(), useBus(), and ~NetworkModelBusGlobal().

6.216.4.7 _time_used

```
SubsecondTime NetworkModelBusGlobal::_time_used
```

Definition at line 21 of file network_model_bus.h.

Referenced by NetworkModelBusGlobal(), and useBus().

6.216.4.8 _total_delay

```
SubsecondTime NetworkModelBusGlobal::_total_delay
```

Definition at line 22 of file network_model_bus.h.

Referenced by NetworkModelBusGlobal(), and useBus().

The documentation for this class was generated from the following files:

- common/network/ **network_model_bus.h**
- common/network/ **network_model_bus.cc**

6.217 NetworkModelEMeshHopByHop Class Reference

`#include <network_model_emesh_hop_by_hop.h>`

Inheritance diagram for NetworkModelEMeshHopByHop:

Public Types

- enum **OutputDirection** {
  UP = 0, **DOWN**, **LEFT**, **RIGHT**,
  NUM_OUTPUT_DIRECTIONS, **SELF**, **PEER**, **DESTINATION**,
  MAX_OUTPUT_DIRECTIONS }

Public Member Functions

- **NetworkModelEMeshHopByHop** (**Network** *net, **EStaticNetwork** net_type)
- ~**NetworkModelEMeshHopByHop** ()
- void **routePacket** (const **NetPacket** &pkt, std::vector< **Hop** > &nextHops)
- void **processReceivedPacket** (**NetPacket** &pkt)
- void **enable** ()
- void **disable** ()
- bool **isEnabled** ()

Static Public Member Functions

- static void **computeMeshDimensions** (**SInt32** &mesh_width, **SInt32** &mesh_height)
- static std::pair< bool, std::vector< **core_id_t** > > **computeMemoryControllerPositions** (**SInt32** num_↩ memory_controllers, **SInt32** core_count)
- static std::pair< bool, **SInt32** > **computeCoreCountConstraints** (**SInt32** core_count)

Protected Member Functions

- void **createQueueModels** (String name)

Protected Attributes

- bool **m_fake_node**
- **core_id_t** **m_core_id**
- **SInt32** **m_concentration**
- **SInt32** **m_dimensions**
- bool **m_wrap_around**
- **ComponentBandwidthPerCycle** **m_link_bandwidth**
- **ComponentLatency** **m_hop_latency**
- bool **m_broadcast_tree_enabled**
- bool **m_queue_model_enabled**
- String **m_queue_model_type**

Private Member Functions

- void **computePosition** (**core_id_t** core, **SInt32** &x, **SInt32** &y)
- **core_id_t** **computeCoreId** (**SInt32** x, **SInt32** y)
- **SInt32** **computeDistance** (**core_id_t** sender, **core_id_t** receiver)
- void **addHop** (**OutputDirection** direction, **core_id_t** final_dest, **core_id_t** next_dest, **SubsecondTime** pkt_time, **UInt32** pkt_length, std::vector< **Hop** > &nextHops, **core_id_t** requester, **subsecond_time_t** *queue_delay_stats=NULL)
- **SubsecondTime** **computeLatency** (**OutputDirection** direction, **SubsecondTime** pkt_time, **UInt32** pkt↩ _length, **core_id_t** requester, **subsecond_time_t** *queue_delay_stats)
- **SubsecondTime** **computeProcessingTime** (**UInt32** pkt_length)
- **core_id_t** **getNextDest** (**core_id_t** final_dest, **OutputDirection** &direction)
- **SubsecondTime** **computeInjectionPortQueueDelay** (**core_id_t** pkt_receiver, **SubsecondTime** pkt_↩ time, **UInt32** pkt_length)
- **SubsecondTime** **computeEjectionPortQueueDelay** (**SubsecondTime** pkt_time, **UInt32** pkt_length)

Private Attributes

- **SInt32 m_mesh_width**
- **SInt32 m_mesh_height**
- **QueueModel** * **m_queue_models** [**NUM_OUTPUT_DIRECTIONS**]
- **QueueModel** * **m_injection_port_queue_model**
- **QueueModel** * **m_ejection_port_queue_model**
- bool **m_enabled**
- **Lock m_lock**
- **UInt64 m_total_bytes_sent**
- **UInt64 m_total_packets_sent**
- **UInt64 m_total_bytes_received**
- **UInt64 m_total_packets_received**
- **SubsecondTime m_total_contention_delay**
- **SubsecondTime m_total_packet_latency**

6.217.1 Detailed Description

Definition at line 11 of file network_model_emesh_hop_by_hop.h.

6.217.2 Member Enumeration Documentation

6.217.2.1 OutputDirection

enum **NetworkModelEMeshHopByHop::OutputDirection**

Enumerator

UP	
DOWN	
LEFT	
RIGHT	
NUM_OUTPUT_DIRECTIONS	
SELF	
PEER	
DESTINATION	
MAX_OUTPUT_DIRECTIONS	

Definition at line 14 of file network_model_emesh_hop_by_hop.h.

6.217.3 Constructor & Destructor Documentation

6.217.3.1 NetworkModelEMeshHopByHop()

```
NetworkModelEMeshHopByHop::NetworkModelEMeshHopByHop (
              Network * net,
              EStaticNetwork net_type )
```

Definition at line 28 of file network_model_emesh_hop_by_hop.cc.

References computeMeshDimensions(), createQueueModels(), EStaticNetworkStrings, LOG_PRINT_ERROR, m_broadcast_tree_enabled, m_concentration, m_core_id, m_dimensions, m_fake_node, m_hop_latency, m_link↩_bandwidth, m_mesh_height, m_mesh_width, m_queue_model_enabled, m_queue_model_type, m_total_bytes↩_received, m_total_bytes_sent, m_total_contention_delay, m_total_packet_latency, m_total_packets_received, m_total_packets_sent, m_wrap_around, and registerStatsMetric().

6.217.3.2 ~NetworkModelEMeshHopByHop()

```
NetworkModelEMeshHopByHop::~NetworkModelEMeshHopByHop ( )
```

Definition at line 86 of file network_model_emesh_hop_by_hop.cc.

References m_ejection_port_queue_model, m_fake_node, m_injection_port_queue_model, m_queue_models, and NUM_OUTPUT_DIRECTIONS.

6.217.4 Member Function Documentation

6.217.4.1 addHop()

```
void NetworkModelEMeshHopByHop::addHop (
              OutputDirection direction,
              core_id_t final_dest,
              core_id_t next_dest,
              SubsecondTime pkt_time,
              UInt32 pkt_length,
             std::vector< Hop > & nextHops,
              core_id_t requester,
              subsecond_time_t * queue_delay_stats = NULL )  [private]
```

Definition at line 266 of file network_model_emesh_hop_by_hop.cc.

References computeLatency(), NetworkModel::Hop::final_dest, NetworkModel::Hop::next_dest, NUM_OUTPUT↩_DIRECTIONS, and NetworkModel::Hop::time.

Referenced by routePacket().

6.217.4.2 computeCoreCountConstraints()

```
std::pair< bool,  SInt32 > NetworkModelEMeshHopByHop::computeCoreCountConstraints (
              SInt32 core_count )  [static]
```

Definition at line 494 of file network_model_emesh_hop_by_hop.cc.

References computeMeshDimensions().

Referenced by NetworkModel::computeCoreCountConstraints().

6.217.4.3 computeCoreId()

```
core_id_t NetworkModelEMeshHopByHop::computeCoreId (
              SInt32 x,
              SInt32 y )  [private]
```

Definition at line 307 of file network_model_emesh_hop_by_hop.cc.

References m_concentration, m_mesh_height, and m_mesh_width.

Referenced by getNextDest(), and routePacket().

6.217.4.4 computeDistance()

```
SInt32 NetworkModelEMeshHopByHop::computeDistance (
              core_id_t sender,
              core_id_t receiver )  [private]
```

Definition at line 285 of file network_model_emesh_hop_by_hop.cc.

References computePosition(), m_mesh_height, m_mesh_width, and m_wrap_around.

Referenced by processReceivedPacket().

6.217.4.5 computeEjectionPortQueueDelay()

```
SubsecondTime NetworkModelEMeshHopByHop::computeEjectionPortQueueDelay (
              SubsecondTime pkt_time,
              UInt32 pkt_length )  [private]
```

Definition at line 357 of file network_model_emesh_hop_by_hop.cc.

References computeProcessingTime(), QueueModel::computeQueueDelay(), LOG_ASSERT_ERROR, m_↩ ejection_port_queue_model, m_fake_node, m_queue_model_enabled, and SubsecondTime::Zero().

Referenced by processReceivedPacket().

6.217.4.6 computeInjectionPortQueueDelay()

```
SubsecondTime NetworkModelEMeshHopByHop::computeInjectionPortQueueDelay (
            core_id_t pkt_receiver,
            SubsecondTime pkt_time,
            UInt32 pkt_length ) [private]
```

Definition at line 342 of file network_model_emesh_hop_by_hop.cc.

References computeProcessingTime(), QueueModel::computeQueueDelay(), LOG_ASSERT_ERROR, m_core_id, m_fake_node, m_injection_port_queue_model, m_queue_model_enabled, and SubsecondTime::Zero().

Referenced by routePacket().

6.217.4.7 computeLatency()

```
SubsecondTime NetworkModelEMeshHopByHop::computeLatency (
            OutputDirection direction,
            SubsecondTime pkt_time,
            UInt32 pkt_length,
            core_id_t requester,
            subsecond_time_t * queue_delay_stats ) [private]
```

Definition at line 315 of file network_model_emesh_hop_by_hop.cc.

References computeProcessingTime(), QueueModel::computeQueueDelay(), ComponentLatency::getLatency(), Config::getSingleton(), itostr(), LOG_ASSERT_ERROR, LOG_PRINT, m_enabled, m_fake_node, m_hop_latency, m_queue_model_enabled, m_queue_models, NUM_OUTPUT_DIRECTIONS, and SubsecondTime::Zero().

Referenced by addHop().

6.217.4.8 computeMemoryControllerPositions()

```
std::pair< bool, std::vector< core_id_t > > NetworkModelEMeshHopByHop::computeMemoryController↩
Positions (
            SInt32 num_memory_controllers,
            SInt32 core_count ) [static]
```

Definition at line 507 of file network_model_emesh_hop_by_hop.cc.

References computeMeshDimensions(), and LOG_ASSERT_ERROR.

Referenced by NetworkModel::computeMemoryControllerPositions().

6.217.4.9 computeMeshDimensions()

```
void NetworkModelEMeshHopByHop::computeMeshDimensions (
              SInt32 & mesh_width,
              SInt32 & mesh_height )  [static]
```

Definition at line 457 of file network_model_emesh_hop_by_hop.cc.

References Config::getApplicationCores(), Config::getSingleton(), LOG_ASSERT_ERROR, and LOG_PRINT_E↩ RROR.

Referenced by computeCoreCountConstraints(), computeMemoryControllerPositions(), and NetworkModelE↩ MeshHopByHop().

6.217.4.10 computePosition()

```
void NetworkModelEMeshHopByHop::computePosition (
              core_id_t core,
              SInt32 & x,
              SInt32 & y )  [private]
```

Definition at line 300 of file network_model_emesh_hop_by_hop.cc.

References m_concentration, and m_mesh_width.

Referenced by computeDistance(), getNextDest(), and routePacket().

6.217.4.11 computeProcessingTime()

```
SubsecondTime NetworkModelEMeshHopByHop::computeProcessingTime (
              UInt32 pkt_length )  [private]
```

Definition at line 369 of file network_model_emesh_hop_by_hop.cc.

References ComponentBandwidthPerCycle::getRoundedLatency(), LOG_ASSERT_ERROR, m_fake_node, and m_link_bandwidth.

Referenced by computeEjectionPortQueueDelay(), computeInjectionPortQueueDelay(), computeLatency(), and processReceivedPacket().

6.217.4.12 createQueueModels()

```
void NetworkModelEMeshHopByHop::createQueueModels (
             String name )  [protected]
```

Definition at line 102 of file network_model_emesh_hop_by_hop.cc.

References QueueModel::create(), DOWN, ComponentBandwidthPerCycle::getPeriod(), LEFT, m_core_id, m↩ _ejection_port_queue_model, m_injection_port_queue_model, m_link_bandwidth, m_queue_model_type, m_↩ queue_models, RIGHT, and UP.

Referenced by NetworkModelEMeshHopByHop().

6.217.4.13 disable()

```
void NetworkModelEMeshHopByHop::disable ( )  [virtual]
```

Implements **NetworkModel** (p. 840).

Definition at line 445 of file network_model_emesh_hop_by_hop.cc.

References m_enabled.

6.217.4.14 enable()

```
void NetworkModelEMeshHopByHop::enable ( )  [virtual]
```

Implements **NetworkModel** (p. 840).

Definition at line 439 of file network_model_emesh_hop_by_hop.cc.

References m_enabled.

6.217.4.15 getNextDest()

```
SInt32 NetworkModelEMeshHopByHop::getNextDest (
            core_id_t final_dest,
            OutputDirection & direction )  [private]
```

Definition at line 380 of file network_model_emesh_hop_by_hop.cc.

References computeCoreId(), computePosition(), DESTINATION, DOWN, Config::getSingleton(), LEFT, m_↩ concentration, m_core_id, m_fake_node, m_mesh_height, m_mesh_width, m_wrap_around, PEER, RIGHT, SELF, and UP.

Referenced by routePacket().

6.217.4.16 isEnabled()

```
bool NetworkModelEMeshHopByHop::isEnabled ( )
```

Definition at line 451 of file network_model_emesh_hop_by_hop.cc.

References m_enabled.

6.217.4.17 processReceivedPacket()

```
void NetworkModelEMeshHopByHop::processReceivedPacket (
              NetPacket & pkt )  [virtual]
```

Implements **NetworkModel** (p. 841).

Definition at line 225 of file network_model_emesh_hop_by_hop.cc.

References computeDistance(), computeEjectionPortQueueDelay(), computeProcessingTime(), NetPacket::data, Network::getCore(), ComponentLatency::getLatency(), Core::getMemoryManager(), Network::getModeledLength(), NetworkModel::getNetwork(), MemoryManagerBase::getShmemRequester(), Config::getSingleton(), INVALID_CORE_ID, LOG_ASSERT_ERROR, m_core_id, m_enabled, m_fake_node, m_hop_latency, m_lock, m_total_bytes_received, m_total_contention_delay, m_total_packet_latency, m_total_packets_received, NetPacket::queue_delay, NetPacket::sender, SHARED_MEM_1, NetPacket::start_time, NetPacket::time, and NetPacket::type.

6.217.4.18 routePacket()

```
void NetworkModelEMeshHopByHop::routePacket (
             const  NetPacket & pkt,
             std::vector<  Hop > & nextHops )  [virtual]
```

Implements **NetworkModel** (p. 841).

Definition at line 117 of file network_model_emesh_hop_by_hop.cc.

References addHop(), NetPacket::BROADCAST, computeCoreId(), computeInjectionPortQueueDelay(), computePosition(), NetPacket::data, DESTINATION, DOWN, Network::getCore(), Core::getMemoryManager(), Network::getModeledLength(), NetworkModel::getNetwork(), getNextDest(), MemoryManagerBase::getShmemRequester(), Config::getSingleton(), Config::getTotalCores(), INVALID_CORE_ID, LEFT, LOG_ASSERT_ERROR, LOG_PRINT, m_broadcast_tree_enabled, m_core_id, m_fake_node, m_lock, m_total_bytes_sent, m_total_packets_sent, NetPacket::queue_delay, NetPacket::receiver, RIGHT, SELF, NetPacket::sender, SHARED_MEM_1, NetPacket::time, NetPacket::type, UP, and SubsecondTime::Zero().

6.217.5 Member Data Documentation

6.217.5.1 m_broadcast_tree_enabled

```
bool NetworkModelEMeshHopByHop::m_broadcast_tree_enabled  [protected]
```

Definition at line 73 of file network_model_emesh_hop_by_hop.h.

Referenced by NetworkModelEMeshHopByHop(), and routePacket().

6.217.5.2 m_concentration

```
SInt32 NetworkModelEMeshHopByHop::m_concentration [protected]
```

Definition at line 67 of file network_model_emesh_hop_by_hop.h.

Referenced by computeCoreId(), computePosition(), getNextDest(), and NetworkModelEMeshHopByHop().

6.217.5.3 m_core_id

```
core_id_t NetworkModelEMeshHopByHop::m_core_id [protected]
```

Definition at line 66 of file network_model_emesh_hop_by_hop.h.

Referenced by computeInjectionPortQueueDelay(), createQueueModels(), getNextDest(), NetworkModelEMesh↩HopByHop(), processReceivedPacket(), and routePacket().

6.217.5.4 m_dimensions

```
SInt32 NetworkModelEMeshHopByHop::m_dimensions [protected]
```

Definition at line 68 of file network_model_emesh_hop_by_hop.h.

Referenced by NetworkModelEMeshHopByHop().

6.217.5.5 m_ejection_port_queue_model

```
QueueModel* NetworkModelEMeshHopByHop::m_ejection_port_queue_model [private]
```

Definition at line 35 of file network_model_emesh_hop_by_hop.h.

Referenced by computeEjectionPortQueueDelay(), createQueueModels(), and ∼NetworkModelEMeshHopBy↩Hop().

6.217.5.6 m_enabled

```
bool NetworkModelEMeshHopByHop::m_enabled [private]
```

Definition at line 37 of file network_model_emesh_hop_by_hop.h.

Referenced by computeLatency(), disable(), enable(), isEnabled(), and processReceivedPacket().

6.217.5.7 m_fake_node

```
bool NetworkModelEMeshHopByHop::m_fake_node  [protected]
```

Definition at line 65 of file network_model_emesh_hop_by_hop.h.

Referenced by computeEjectionPortQueueDelay(), computeInjectionPortQueueDelay(), computeLatency(), computeProcessingTime(), getNextDest(), NetworkModelEMeshHopByHop(), processReceivedPacket(), route↩Packet(), and ∼NetworkModelEMeshHopByHop().

6.217.5.8 m_hop_latency

```
ComponentLatency NetworkModelEMeshHopByHop::m_hop_latency  [protected]
```

Definition at line 72 of file network_model_emesh_hop_by_hop.h.

Referenced by computeLatency(), NetworkModelEMeshHopByHop(), and processReceivedPacket().

6.217.5.9 m_injection_port_queue_model

```
QueueModel* NetworkModelEMeshHopByHop::m_injection_port_queue_model  [private]
```

Definition at line 34 of file network_model_emesh_hop_by_hop.h.

Referenced by computeInjectionPortQueueDelay(), createQueueModels(), and ∼NetworkModelEMeshHopBy↩Hop().

6.217.5.10 m_link_bandwidth

```
ComponentBandwidthPerCycle NetworkModelEMeshHopByHop::m_link_bandwidth  [protected]
```

Definition at line 71 of file network_model_emesh_hop_by_hop.h.

Referenced by computeProcessingTime(), createQueueModels(), and NetworkModelEMeshHopByHop().

6.217.5.11 m_lock

```
Lock NetworkModelEMeshHopByHop::m_lock  [private]
```

Definition at line 40 of file network_model_emesh_hop_by_hop.h.

Referenced by processReceivedPacket(), and routePacket().

6.217.5.12 m_mesh_height

SInt32 NetworkModelEMeshHopByHop::m_mesh_height [private]

Definition at line 31 of file network_model_emesh_hop_by_hop.h.

Referenced by computeCoreId(), computeDistance(), getNextDest(), and NetworkModelEMeshHopByHop().

6.217.5.13 m_mesh_width

SInt32 NetworkModelEMeshHopByHop::m_mesh_width [private]

Definition at line 30 of file network_model_emesh_hop_by_hop.h.

Referenced by computeCoreId(), computeDistance(), computePosition(), getNextDest(), and NetworkModelE↩ MeshHopByHop().

6.217.5.14 m_queue_model_enabled

bool NetworkModelEMeshHopByHop::m_queue_model_enabled [protected]

Definition at line 75 of file network_model_emesh_hop_by_hop.h.

Referenced by computeEjectionPortQueueDelay(), computeInjectionPortQueueDelay(), computeLatency(), and NetworkModelEMeshHopByHop().

6.217.5.15 m_queue_model_type

String NetworkModelEMeshHopByHop::m_queue_model_type [protected]

Definition at line 76 of file network_model_emesh_hop_by_hop.h.

Referenced by createQueueModels(), and NetworkModelEMeshHopByHop().

6.217.5.16 m_queue_models

QueueModel* NetworkModelEMeshHopByHop::m_queue_models[**NUM_OUTPUT_DIRECTIONS**] [private]

Definition at line 33 of file network_model_emesh_hop_by_hop.h.

Referenced by computeLatency(), createQueueModels(), and ~NetworkModelEMeshHopByHop().

6.217.5.17 m_total_bytes_received

UInt64 NetworkModelEMeshHopByHop::m_total_bytes_received [private]

Definition at line 45 of file network_model_emesh_hop_by_hop.h.

Referenced by NetworkModelEMeshHopByHop(), and processReceivedPacket().

6.217.5.18 m_total_bytes_sent

UInt64 NetworkModelEMeshHopByHop::m_total_bytes_sent [private]

Definition at line 43 of file network_model_emesh_hop_by_hop.h.

Referenced by NetworkModelEMeshHopByHop(), and routePacket().

6.217.5.19 m_total_contention_delay

SubsecondTime NetworkModelEMeshHopByHop::m_total_contention_delay [private]

Definition at line 47 of file network_model_emesh_hop_by_hop.h.

Referenced by NetworkModelEMeshHopByHop(), and processReceivedPacket().

6.217.5.20 m_total_packet_latency

SubsecondTime NetworkModelEMeshHopByHop::m_total_packet_latency [private]

Definition at line 48 of file network_model_emesh_hop_by_hop.h.

Referenced by NetworkModelEMeshHopByHop(), and processReceivedPacket().

6.217.5.21 m_total_packets_received

UInt64 NetworkModelEMeshHopByHop::m_total_packets_received [private]

Definition at line 46 of file network_model_emesh_hop_by_hop.h.

Referenced by NetworkModelEMeshHopByHop(), and processReceivedPacket().

6.217.5.22 m_total_packets_sent

UInt64 NetworkModelEMeshHopByHop::m_total_packets_sent [private]

Definition at line 44 of file network_model_emesh_hop_by_hop.h.

Referenced by NetworkModelEMeshHopByHop(), and routePacket().

6.217.5.23 m_wrap_around

bool NetworkModelEMeshHopByHop::m_wrap_around [protected]

Definition at line 69 of file network_model_emesh_hop_by_hop.h.

Referenced by computeDistance(), getNextDest(), and NetworkModelEMeshHopByHop().

The documentation for this class was generated from the following files:

- common/network/ **network_model_emesh_hop_by_hop.h**
- common/network/ **network_model_emesh_hop_by_hop.cc**

6.218 NetworkModelEMeshHopCounter Class Reference

#include <network_model_emesh_hop_counter.h>

Inheritance diagram for NetworkModelEMeshHopCounter:

Public Member Functions

- **NetworkModelEMeshHopCounter** (**Network** *net, **EStaticNetwork** net_type)
- ~**NetworkModelEMeshHopCounter** ()
- void **routePacket** (const **NetPacket** &pkt, std::vector< **Hop** > &nextHops)
- void **processReceivedPacket** (**NetPacket** &pkt)
- void **enable** ()
- void **disable** ()

Private Member Functions

- void **computePosition** (**core_id_t** core, **SInt32** &x, **SInt32** &y)
- **SInt32** **computeDistance** (**SInt32** x1, **SInt32** y1, **SInt32** x2, **SInt32** y2)
- **SubsecondTime** **computeSerializationLatency** (**UInt32** pkt_length)

Private Attributes

- **ComponentLatency _hopLatency**
- **ComponentBandwidthPerCycle _linkBandwidth**
- **SInt32 _meshWidth**
- **SInt32 _meshHeight**
- bool **_enabled**
- **Lock _lock**
- **UInt64 _num_packets**
- **UInt64 _num_bytes**
- **SubsecondTime _total_latency**

Additional Inherited Members

6.218.1 Detailed Description

Definition at line 8 of file network_model_emesh_hop_counter.h.

6.218.2 Constructor & Destructor Documentation

6.218.2.1 NetworkModelEMeshHopCounter()

```
NetworkModelEMeshHopCounter::NetworkModelEMeshHopCounter (
              Network * net,
              EStaticNetwork net_type )
```

Definition at line 13 of file network_model_emesh_hop_counter.cc.

References _hopLatency, _linkBandwidth, _meshHeight, _meshWidth, Network::getCore(), Core::getId(), Config↩ ::getSingleton(), Config::getTotalCores(), and LOG_PRINT_ERROR.

6.218.2.2 ~NetworkModelEMeshHopCounter()

```
NetworkModelEMeshHopCounter::~NetworkModelEMeshHopCounter ( )
```

Definition at line 41 of file network_model_emesh_hop_counter.cc.

6.218.3 Member Function Documentation

6.218.3.1 computeDistance()

```
SInt32 NetworkModelEMeshHopCounter::computeDistance (
            SInt32 x1,
            SInt32 y1,
            SInt32 x2,
            SInt32 y2 )  [private]
```

Definition at line 52 of file network_model_emesh_hop_counter.cc.

Referenced by routePacket().

6.218.3.2 computePosition()

```
void NetworkModelEMeshHopCounter::computePosition (
            core_id_t core,
            SInt32 & x,
            SInt32 & y )  [private]
```

Definition at line 45 of file network_model_emesh_hop_counter.cc.

References _meshWidth.

Referenced by routePacket().

6.218.3.3 computeSerializationLatency()

```
SubsecondTime NetworkModelEMeshHopCounter::computeSerializationLatency (
            UInt32 pkt_length )  [private]
```

Definition at line 141 of file network_model_emesh_hop_counter.cc.

References _linkBandwidth, and ComponentBandwidthPerCycle::getRoundedLatency().

Referenced by routePacket().

6.218.3.4 disable()

```
void NetworkModelEMeshHopCounter::disable ( )  [inline], [virtual]
```

Implements **NetworkModel** (p. 840).

Definition at line 19 of file network_model_emesh_hop_counter.h.

References _enabled.

6.218.3.5 enable()

```
void NetworkModelEMeshHopCounter::enable ( )  [inline], [virtual]
```

Implements **NetworkModel** (p. 840).

Definition at line 18 of file network_model_emesh_hop_counter.h.

References _enabled.

6.218.3.6 processReceivedPacket()

```
void NetworkModelEMeshHopCounter::processReceivedPacket (
              NetPacket & pkt )  [virtual]
```

Implements **NetworkModel** (p. 841).

Definition at line 112 of file network_model_emesh_hop_counter.cc.

References _enabled, _lock, _num_bytes, _num_packets, _total_latency, NetPacket::data, Network::getCore(), Core::getMemoryManager(), Network::getModeledLength(), NetworkModel::getNetwork(), MemoryManager↩Base::getShmemRequester(), Config::getSingleton(), INVALID_CORE_ID, LOG_ASSERT_ERROR, NetPacket↩::sender, SHARED_MEM_1, NetPacket::start_time, NetPacket::time, and NetPacket::type.

6.218.3.7 routePacket()

```
void NetworkModelEMeshHopCounter::routePacket (
             const  NetPacket & pkt,
             std::vector<  Hop > & nextHops )  [virtual]
```

Implements **NetworkModel** (p. 841).

Definition at line 57 of file network_model_emesh_hop_counter.cc.

References _hopLatency, NetPacket::BROADCAST, computeDistance(), computePosition(), compute↩SerializationLatency(), NetworkModel::Hop::final_dest, ComponentLatency::getLatency(), Network::getModeled↩Length(), NetworkModel::getNetwork(), Config::getSingleton(), Config::getTotalCores(), NetworkModel::Hop::next↩_dest, NetPacket::receiver, NetPacket::sender, NetPacket::time, and NetworkModel::Hop::time.

6.218.4 Member Data Documentation

6.218.4.1 _enabled

```
bool NetworkModelEMeshHopCounter::_enabled  [private]
```

Definition at line 34 of file network_model_emesh_hop_counter.h.

Referenced by disable(), enable(), and processReceivedPacket().

6.218.4.2 _hopLatency

```
ComponentLatency NetworkModelEMeshHopCounter::_hopLatency  [private]
```

Definition at line 28 of file network_model_emesh_hop_counter.h.

Referenced by NetworkModelEMeshHopCounter(), and routePacket().

6.218.4.3 _linkBandwidth

```
ComponentBandwidthPerCycle NetworkModelEMeshHopCounter::_linkBandwidth  [private]
```

Definition at line 29 of file network_model_emesh_hop_counter.h.

Referenced by computeSerializationLatency(), and NetworkModelEMeshHopCounter().

6.218.4.4 _lock

```
Lock NetworkModelEMeshHopCounter::_lock  [private]
```

Definition at line 36 of file network_model_emesh_hop_counter.h.

Referenced by processReceivedPacket().

6.218.4.5 _meshHeight

```
SInt32 NetworkModelEMeshHopCounter::_meshHeight  [private]
```

Definition at line 32 of file network_model_emesh_hop_counter.h.

Referenced by NetworkModelEMeshHopCounter().

6.218.4.6 _meshWidth

SInt32 NetworkModelEMeshHopCounter::_meshWidth [private]

Definition at line 31 of file network_model_emesh_hop_counter.h.

Referenced by computePosition(), and NetworkModelEMeshHopCounter().

6.218.4.7 _num_bytes

UInt64 NetworkModelEMeshHopCounter::_num_bytes [private]

Definition at line 40 of file network_model_emesh_hop_counter.h.

Referenced by processReceivedPacket().

6.218.4.8 _num_packets

UInt64 NetworkModelEMeshHopCounter::_num_packets [private]

Definition at line 39 of file network_model_emesh_hop_counter.h.

Referenced by processReceivedPacket().

6.218.4.9 _total_latency

SubsecondTime NetworkModelEMeshHopCounter::_total_latency [private]

Definition at line 41 of file network_model_emesh_hop_counter.h.

Referenced by processReceivedPacket().

The documentation for this class was generated from the following files:

- common/network/ **network_model_emesh_hop_counter.h**
- common/network/ **network_model_emesh_hop_counter.cc**

6.219 NetworkModelMagic Class Reference

`#include <network_model_magic.h>`

Inheritance diagram for NetworkModelMagic:

Public Member Functions

- **NetworkModelMagic** (**Network** ∗net, **EStaticNetwork** net_type)
- ∼**NetworkModelMagic** ()
- void **routePacket** (const **NetPacket** &pkt, std::vector< **Hop** > &nextHops)
- void **processReceivedPacket** (**NetPacket** &pkt)
- void **enable** ()
- void **disable** ()

Private Attributes

- bool **_enabled**
- **Lock** **_lock**
- **UInt64** **_num_packets**
- **UInt64** **_num_bytes**
- **ComponentLatency** **_latency**

Additional Inherited Members

6.219.1 Detailed Description

Definition at line 9 of file network_model_magic.h.

6.219.2 Constructor & Destructor Documentation

6.219.2.1 NetworkModelMagic()

```
NetworkModelMagic::NetworkModelMagic (
              Network * net,
              EStaticNetwork net_type )
```

Definition at line 8 of file network_model_magic.cc.

6.219.2.2 ∼NetworkModelMagic()

```
NetworkModelMagic::~NetworkModelMagic ( )  [inline]
```

Definition at line 23 of file network_model_magic.h.

6.219.3 Member Function Documentation

6.219.3.1 disable()

```
void NetworkModelMagic::disable ( )  [inline], [virtual]
```

Implements **NetworkModel** (p. 840).

Definition at line 32 of file network_model_magic.h.

References _enabled.

6.219.3.2 enable()

```
void NetworkModelMagic::enable ( )  [inline], [virtual]
```

Implements **NetworkModel** (p. 840).

Definition at line 29 of file network_model_magic.h.

References _enabled.

6.219.3.3 processReceivedPacket()

```
void NetworkModelMagic::processReceivedPacket (
              NetPacket & pkt )  [virtual]
```

Implements **NetworkModel** (p. 841).

Definition at line 46 of file network_model_magic.cc.

References _enabled, _lock, _num_bytes, _num_packets, NetPacket::data, Network::getCore(), Core::get↩MemoryManager(), Network::getModeledLength(), NetworkModel::getNetwork(), MemoryManagerBase::get↩ShmemRequester(), Config::getSingleton(), INVALID_CORE_ID, LOG_ASSERT_ERROR, NetPacket::sender, SHARED_MEM_1, and NetPacket::type.

6.219.3.4 routePacket()

```
void NetworkModelMagic::routePacket (
             const  NetPacket & pkt,
             std::vector<  Hop > & nextHops )  [virtual]
```

Implements **NetworkModel** (p. 841).

Definition at line 17 of file network_model_magic.cc.

References _latency, NetPacket::BROADCAST, NetworkModel::Hop::final_dest, ComponentLatency::getLatency(), Config::getSingleton(), Config::getTotalCores(), NetworkModel::Hop::next_dest, NetPacket::receiver, NetPacket↩::time, and NetworkModel::Hop::time.

6.219.4 Member Data Documentation

6.219.4.1 _enabled

```
bool NetworkModelMagic::_enabled  [private]
```

Definition at line 12 of file network_model_magic.h.

Referenced by disable(), enable(), and processReceivedPacket().

6.219.4.2 _latency

```
ComponentLatency NetworkModelMagic::_latency  [private]
```

Definition at line 19 of file network_model_magic.h.

Referenced by routePacket().

6.219.4.3 _lock

```
Lock NetworkModelMagic::_lock  [private]
```

Definition at line 14 of file network_model_magic.h.

Referenced by processReceivedPacket().

6.219.4.4 _num_bytes

```
UInt64 NetworkModelMagic::_num_bytes  [private]
```

Definition at line 17 of file network_model_magic.h.

Referenced by processReceivedPacket().

6.219.4.5 _num_packets

```
UInt64 NetworkModelMagic::_num_packets [private]
```

Definition at line 16 of file network_model_magic.h.

Referenced by processReceivedPacket().

The documentation for this class was generated from the following files:

- common/network/ **network_model_magic.h**
- common/network/ **network_model_magic.cc**

6.220 Transport::Node Class Reference

```
#include <transport.h>
```

Inheritance diagram for Transport::Node:

Public Member Functions

- virtual ~**Node** ()
- virtual void **globalSend** (**SInt32** dest_proc, const void *buffer, **UInt32** length)=0
- virtual void **send** (**core_id_t** dest, const void *buffer, **UInt32** length)=0
- virtual **Byte** * **recv** ()=0
- virtual bool **query** ()=0

Protected Member Functions

- **core_id_t** **getCoreId** ()
- **Node** (**core_id_t** core_id)

Private Attributes

- **core_id_t** **m_core_id**

6.220.1 Detailed Description

Definition at line 13 of file transport.h.

6.220.2 Constructor & Destructor Documentation

6.220.2.1 ~Node()

```
virtual Transport::Node::~Node ( )  [inline], [virtual]
```

Definition at line 16 of file transport.h.

6.220.2.2 Node()

```
Transport::Node::Node (
             core_id_t core_id )  [protected]
```

Definition at line 35 of file transport.cc.

6.220.3 Member Function Documentation

6.220.3.1 getCoreId()

```
 core_id_t Transport::Node::getCoreId ( )  [protected]
```

Definition at line 40 of file transport.cc.

6.220.3.2 globalSend()

```
virtual void Transport::Node::globalSend (
             SInt32 dest_proc,
            const void * buffer,
             UInt32 length )  [pure virtual]
```

Implemented in **SmTransport::SmNode** (p. 1245).

6.220.3.3 query()

```
virtual bool Transport::Node::query ( )  [pure virtual]
```

Implemented in **SmTransport::SmNode** (p. 1245).

Referenced by Network::netPullFromTransport().

6.220.3.4 recv()

```
virtual  Byte* Transport::Node::recv ( )  [pure virtual]
```

Implemented in **SmTransport::SmNode** (p. 1245).

Referenced by Network::netPullFromTransport().

6.220.3.5 send()

```
virtual void Transport::Node::send (
              core_id_t dest,
             const void * buffer,
              UInt32 length )  [pure virtual]
```

Implemented in **SmTransport::SmNode** (p. 1246).

Referenced by Network::netSend(), and SimThreadManager::quitSimThreads().

6.220.4 Member Data Documentation

6.220.4.1 m_core_id

```
 core_id_t Transport::Node::m_core_id  [private]
```

Definition at line 28 of file transport.h.

The documentation for this class was generated from the following files:

- common/transport/ **transport.h**
- common/transport/ **transport.cc**

6.221 config::config_parser::NodeValue Struct Reference

```
#include <config_file_grammar.hpp>
```

Public Member Functions

- **NodeValue** ()
- **NodeValue** (**RuleID** e_)

Public Attributes

- **RuleID e**

6.221.1 Detailed Description

Definition at line 63 of file config_file_grammar.hpp.

6.221.2 Constructor & Destructor Documentation

6.221.2.1 NodeValue() [1/2]

```
config::config_parser::NodeValue::NodeValue ( )  [inline]
```

Definition at line 66 of file config_file_grammar.hpp.

6.221.2.2 NodeValue() [2/2]

```
config::config_parser::NodeValue::NodeValue (
            RuleID e_ )  [inline]
```

Definition at line 67 of file config_file_grammar.hpp.

6.221.3 Member Data Documentation

6.221.3.1 e

```
RuleID config::config_parser::NodeValue::e
```

Definition at line 64 of file config_file_grammar.hpp.

The documentation for this struct was generated from the following file:

- common/config/ **config_file_grammar.hpp**

6.222 NormalFloatDistribution Class Reference

`#include <distribution.h>`

Inheritance diagram for NormalFloatDistribution:

Public Member Functions

- **NormalFloatDistribution** (double mean, double stddev, unsigned seed=1000)
- double **next** ()

Private Attributes

- std::tr1::ranlux64_base_01 **generator**
- std::tr1::normal_distribution< double > **distribution**

6.222.1 Detailed Description

Definition at line 15 of file distribution.h.

6.222.2 Constructor & Destructor Documentation

6.222.2.1 NormalFloatDistribution()

```
NormalFloatDistribution::NormalFloatDistribution (
            double mean,
            double stddev,
            unsigned seed = 1000 )  [inline]
```

Definition at line 18 of file distribution.h.

6.222.3 Member Function Documentation

6.222.3.1 next()

```
double NormalFloatDistribution::next ( )  [inline], [virtual]
```

Implements **FloatDistribution** (p. 568).

Definition at line 23 of file distribution.h.

References distribution, and generator.

Referenced by NormalTimeDistribution::next().

6.222.4 Member Data Documentation

6.222.4.1 distribution

```
std::tr1::normal_distribution<double> NormalFloatDistribution::distribution  [private]
```

Definition at line 32 of file distribution.h.

Referenced by next().

6.222.4.2 generator

```
std::tr1::ranlux64_base_01 NormalFloatDistribution::generator  [private]
```

Definition at line 31 of file distribution.h.

Referenced by next().

The documentation for this class was generated from the following file:

- common/misc/ **distribution.h**

6.223 NormalTimeDistribution Class Reference

```
#include <distribution.h>
```

Inheritance diagram for NormalTimeDistribution:

Public Member Functions

- **NormalTimeDistribution** (**SubsecondTime** mean, **SubsecondTime** stddev, unsigned seed=1000)
- **SubsecondTime next** ()

Private Attributes

- **NormalFloatDistribution normal_dist**

6.223.1 Detailed Description

Definition at line 55 of file distribution.h.

6.223.2 Constructor & Destructor Documentation

6.223.2.1 NormalTimeDistribution()

```
NormalTimeDistribution::NormalTimeDistribution (
              SubsecondTime mean,
              SubsecondTime stddev,
             unsigned seed = 1000 )  [inline]
```

Definition at line 58 of file distribution.h.

6.223.3 Member Function Documentation

6.223.3.1 next()

```
SubsecondTime NormalTimeDistribution::next ( )  [inline], [virtual]
```

Implements **TimeDistribution** (p. 1408).

Definition at line 62 of file distribution.h.

References SubsecondTime::FS(), NormalFloatDistribution::next(), and normal_dist.

6.223.4 Member Data Documentation

6.223.4.1 normal_dist

NormalFloatDistribution NormalTimeDistribution::normal_dist [private]

Definition at line 68 of file distribution.h.

Referenced by next().

The documentation for this class was generated from the following file:

- common/misc/ **distribution.h**

6.224 NucaCache Class Reference

#include <nuca_cache.h>

Public Member Functions

- **NucaCache** (**MemoryManagerBase** *memory_manager, **ShmemPerfModel** *shmem_perf_model, **AddressHomeLookup** *home_lookup, **UInt32** cache_block_size, **ParametricDramDirectoryMSI::CacheParameters** &parameters)
- ~**NucaCache** ()
- boost::tuple< **SubsecondTime**, **HitWhere::where_t** > **read** (**IntPtr** address, **Byte** *data_buf, **SubsecondTime** now, **ShmemPerf** *perf, bool count)
- boost::tuple< **SubsecondTime**, **HitWhere::where_t** > **write** (**IntPtr** address, **Byte** *data_buf, bool &eviction, **IntPtr** &evict_address, **Byte** *evict_buf, **SubsecondTime** now, bool count)

Private Member Functions

- **SubsecondTime accessDataArray** (**Cache::access_t** access, **SubsecondTime t_start**, **ShmemPerf** *perf)

Private Attributes

- **core_id_t m_core_id**
- **MemoryManagerBase** * **m_memory_manager**
- **ShmemPerfModel** * **m_shmem_perf_model**
- **AddressHomeLookup** * **m_home_lookup**
- **UInt32 m_cache_block_size**
- **ComponentLatency m_data_access_time**
- **ComponentLatency m_tags_access_time**
- **ComponentBandwidth m_data_array_bandwidth**
- **Cache** * **m_cache**
- **QueueModel** * **m_queue_model**
- **UInt64 m_reads**
- **UInt64 m_writes**
- **UInt64 m_read_misses**
- **UInt64 m_write_misses**
- **ShmemPerf m_dummy_shmem_perf**

6.224.1 Detailed Description

Definition at line 17 of file nuca_cache.h.

6.224.2 Constructor & Destructor Documentation

6.224.2.1 NucaCache()

```
NucaCache::NucaCache (
            MemoryManagerBase * memory_manager,
            ShmemPerfModel * shmem_perf_model,
            AddressHomeLookup * home_lookup,
            UInt32 cache_block_size,
            ParametricDramDirectoryMSI::CacheParameters & parameters )
```

Definition at line 9 of file nuca_cache.cc.

References ParametricDramDirectoryMSI::CacheParameters::associativity, QueueModel::create(), ComponentBandwidth::getRoundedLatency(), ParametricDramDirectoryMSI::CacheParameters::hash_function, m_cache, m_cache_block_size, m_core_id, m_data_array_bandwidth, m_queue_model, m_read_misses, m_reads, m_write_misses, m_writes, ParametricDramDirectoryMSI::CacheParameters::num_sets, CacheBase::parseAddressHash(), CacheBase::PR_L1_CACHE, registerStatsMetric(), and ParametricDramDirectoryMSI::CacheParameters::replacement_policy.

6.224.2.2 ~NucaCache()

```
NucaCache::~NucaCache ( )
```

Definition at line 49 of file nuca_cache.cc.

References m_cache, and m_queue_model.

6.224.3 Member Function Documentation

6.224.3.1 accessDataArray()

```
SubsecondTime NucaCache::accessDataArray (
            Cache::access_t access,
            SubsecondTime t_start,
            ShmemPerf * perf )  [private]
```

Definition at line 123 of file nuca_cache.cc.

References QueueModel::computeQueueDelay(), ComponentLatency::getLatency(), ComponentBandwidth::getRoundedLatency(), m_cache_block_size, m_core_id, m_data_access_time, m_data_array_bandwidth, m_queue_model, ShmemPerf::NUCA_BUS, ShmemPerf::NUCA_DATA, ShmemPerf::NUCA_QUEUE, t_start, ShmemPerf::updateTime(), and SubsecondTime::Zero().

Referenced by read(), and write().

6.224.3.2 read()

```
boost::tuple< SubsecondTime, HitWhere::where_t > NucaCache::read (
              IntPtr address,
              Byte * data_buf,
              SubsecondTime now,
              ShmemPerf * perf,
             bool count )
```

Definition at line 57 of file nuca_cache.cc.

References accessDataArray(), Cache::accessSingleLine(), ComponentLatency::getLatency(), CacheBase::LOAD, m_cache, m_cache_block_size, m_read_misses, m_reads, m_tags_access_time, HitWhere::MISS, HitWhere::NUCA_CACHE, ShmemPerf::NUCA_TAGS, Cache::peekSingleLine(), and ShmemPerf::updateTime().

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache().

6.224.3.3 write()

```
boost::tuple< SubsecondTime, HitWhere::where_t > NucaCache::write (
              IntPtr address,
              Byte * data_buf,
             bool & eviction,
              IntPtr & evict_address,
              Byte * evict_buf,
              SubsecondTime now,
             bool count )
```

Definition at line 83 of file nuca_cache.cc.

References accessDataArray(), Cache::accessSingleLine(), CacheBlockInfo::getCState(), ComponentLatency::getLatency(), Cache::insertSingleLine(), m_cache, m_cache_block_size, m_dummy_shmem_perf, m_tags_access_time, m_write_misses, m_writes, HitWhere::MISS, CacheState::MODIFIED, HitWhere::NUCA_CACHE, Cache::peekSingleLine(), CacheBlockInfo::setCState(), and CacheBase::STORE.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::sendDataToNUCA().

6.224.4 Member Data Documentation

6.224.4.1 m_cache

```
Cache* NucaCache::m_cache  [private]
```

Definition at line 29 of file nuca_cache.h.

Referenced by NucaCache(), read(), write(), and ~NucaCache().

6.224.4.2 m_cache_block_size

```
UInt32 NucaCache::m_cache_block_size [private]
```

Definition at line 24 of file nuca_cache.h.

Referenced by accessDataArray(), NucaCache(), read(), and write().

6.224.4.3 m_core_id

```
core_id_t NucaCache::m_core_id [private]
```

Definition at line 20 of file nuca_cache.h.

Referenced by accessDataArray(), and NucaCache().

6.224.4.4 m_data_access_time

```
ComponentLatency NucaCache::m_data_access_time [private]
```

Definition at line 25 of file nuca_cache.h.

Referenced by accessDataArray().

6.224.4.5 m_data_array_bandwidth

```
ComponentBandwidth NucaCache::m_data_array_bandwidth [private]
```

Definition at line 27 of file nuca_cache.h.

Referenced by accessDataArray(), and NucaCache().

6.224.4.6 m_dummy_shmem_perf

```
ShmemPerf NucaCache::m_dummy_shmem_perf [private]
```

Definition at line 34 of file nuca_cache.h.

Referenced by write().

6.224.4.7 m_home_lookup

```
AddressHomeLookup* NucaCache::m_home_lookup  [private]
```

Definition at line 23 of file nuca_cache.h.

6.224.4.8 m_memory_manager

```
MemoryManagerBase* NucaCache::m_memory_manager  [private]
```

Definition at line 21 of file nuca_cache.h.

6.224.4.9 m_queue_model

```
QueueModel* NucaCache::m_queue_model  [private]
```

Definition at line 30 of file nuca_cache.h.

Referenced by accessDataArray(), NucaCache(), and ~NucaCache().

6.224.4.10 m_read_misses

```
UInt64 NucaCache::m_read_misses  [private]
```

Definition at line 32 of file nuca_cache.h.

Referenced by NucaCache(), and read().

6.224.4.11 m_reads

```
UInt64 NucaCache::m_reads  [private]
```

Definition at line 32 of file nuca_cache.h.

Referenced by NucaCache(), and read().

6.224.4.12 m_shmem_perf_model

```
ShmemPerfModel* NucaCache::m_shmem_perf_model  [private]
```

Definition at line 22 of file nuca_cache.h.

6.224.4.13 m_tags_access_time

```
ComponentLatency NucaCache::m_tags_access_time  [private]
```

Definition at line 26 of file nuca_cache.h.

Referenced by read(), and write().

6.224.4.14 m_write_misses

```
UInt64 NucaCache::m_write_misses  [private]
```

Definition at line 32 of file nuca_cache.h.

Referenced by NucaCache(), and write().

6.224.4.15 m_writes

```
UInt64 NucaCache::m_writes  [private]
```

Definition at line 32 of file nuca_cache.h.

Referenced by NucaCache(), and write().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **nuca_cache.h**
- common/core/memory_subsystem/parametric_dram_directory_msi/ **nuca_cache.cc**

6.225 OneBitBranchPredictor Class Reference

```
#include <one_bit_branch_predictor.h>
```

Inheritance diagram for OneBitBranchPredictor:

Public Member Functions

- **OneBitBranchPredictor** (String name, **core_id_t** core_id, **UInt32** size)
- ∼**OneBitBranchPredictor** ()
- bool **predict** (**IntPtr** ip, **IntPtr** target)
- void **update** (bool predicted, bool actual, **IntPtr** ip, **IntPtr** target)

Private Attributes

- std::vector< bool > **m_bits**

Additional Inherited Members

6.225.1 Detailed Description

Definition at line 8 of file one_bit_branch_predictor.h.

6.225.2 Constructor & Destructor Documentation

6.225.2.1 OneBitBranchPredictor()

```
OneBitBranchPredictor::OneBitBranchPredictor (
            String name,
             core_id_t core_id,
             UInt32 size )
```

Definition at line 4 of file one_bit_branch_predictor.cc.

6.225.2.2 ∼OneBitBranchPredictor()

```
OneBitBranchPredictor::~OneBitBranchPredictor ( )
```

Definition at line 10 of file one_bit_branch_predictor.cc.

6.225.3 Member Function Documentation

6.225.3.1 predict()

```
bool OneBitBranchPredictor::predict (
              IntPtr ip,
              IntPtr target )  [virtual]
```

Implements **BranchPredictor** (p. 125).

Definition at line 14 of file one_bit_branch_predictor.cc.

References m_bits.

6.225.3.2 update()

```
void OneBitBranchPredictor::update (
             bool predicted,
             bool actual,
              IntPtr ip,
              IntPtr target )  [virtual]
```

Implements **BranchPredictor** (p. 126).

Definition at line 20 of file one_bit_branch_predictor.cc.

References m_bits, and BranchPredictor::updateCounters().

6.225.4 Member Data Documentation

6.225.4.1 m_bits

```
std::vector<bool> OneBitBranchPredictor::m_bits  [private]
```

Definition at line 18 of file one_bit_branch_predictor.h.

Referenced by predict(), and update().

The documentation for this class was generated from the following files:

- common/performance_model/branch_predictors/ **one_bit_branch_predictor.h**
- common/performance_model/branch_predictors/ **one_bit_branch_predictor.cc**

6.226 OneIPCPerformanceModel Class Reference

`#include <oneipc_performance_model.h>`

Inheritance diagram for OneIPCPerformanceModel:

Public Member Functions

- **OneIPCPerformanceModel** (**Core** ∗core)
- ∼**OneIPCPerformanceModel** ()

Private Member Functions

- void **handleInstruction** (**DynamicInstruction** ∗instruction)
- bool **isModeled** (**Instruction** const ∗instruction) const

Private Attributes

- **UInt64 m_latency_cutoff**
- **SubsecondTime m_cpiBase**
- **SubsecondTime m_cpiBranchPredictor**
- std::vector< **SubsecondTime** > **m_cpiDataCache**

Additional Inherited Members

6.226.1 Detailed Description

Definition at line 6 of file oneipc_performance_model.h.

6.226.2 Constructor & Destructor Documentation

6.226.2.1 OneIPCPerformanceModel()

```
OneIPCPerformanceModel::OneIPCPerformanceModel (
            Core * core )
```

Definition at line 15 of file oneipc_performance_model.cc.

References Core::getId(), HitWhereIsValid(), HitWhereString(), m_cpiBase, m_cpiBranchPredictor, m_cpiData↩Cache, m_latency_cutoff, HitWhere::NUM_HITWHERES, registerStatsMetric(), HitWhere::WHERE_FIRST, and SubsecondTime::Zero().

6.226.2.2 ~OneIPCPerformanceModel()

```
OneIPCPerformanceModel::~OneIPCPerformanceModel ( )
```

Definition at line 35 of file oneipc_performance_model.cc.

6.226.3 Member Function Documentation

6.226.3.1 handleInstruction()

```
void OneIPCPerformanceModel::handleInstruction (
            DynamicInstruction * instruction )  [private], [virtual]
```

Implements **PerformanceModel** (p. 911).

Definition at line 39 of file oneipc_performance_model.cc.

References DynamicInstruction::accessMemory(), ComponentTime::addLatency(), DynamicInstruction::MemoryInfo::dir, PerformanceModel::getCore(), Instruction::getCost(), ComponentTime::getElapsedTime(), ComponentTime::getLatencyGenerator(), Instruction::getOperands(), Instruction::getType(), DynamicInstruction::MemoryInfo::hit_where, INST_BRANCH, INST_RECV, INST_SYNC, DynamicInstruction::instruction, isModeled(), DynamicInstruction::MemoryInfo::latency, LOG_ASSERT_ERROR, m_cpiBase, m_cpiBranchPredictor, m_cpiDataCache, Operand::m_direction, PerformanceModel::m_elapsed_time, PerformanceModel::m_instruction_count, m_latency_cutoff, Operand::m_type, Operand::MEMORY, DynamicInstruction::memory_info, DynamicInstruction::num_memory, and Operand::READ.

6.226.3.2 isModeled()

```
bool OneIPCPerformanceModel::isModeled (
            Instruction const * instruction ) const  [private]
```

Definition at line 103 of file oneipc_performance_model.cc.

References Instruction::getType(), INST_BRANCH, and Instruction::isPseudo().

Referenced by handleInstruction().

6.226.4 Member Data Documentation

6.226.4.1 m_cpiBase

```
SubsecondTime OneIPCPerformanceModel::m_cpiBase [private]
```

Definition at line 19 of file oneipc_performance_model.h.

Referenced by handleInstruction(), and OneIPCPerformanceModel().

6.226.4.2 m_cpiBranchPredictor

```
SubsecondTime OneIPCPerformanceModel::m_cpiBranchPredictor [private]
```

Definition at line 20 of file oneipc_performance_model.h.

Referenced by handleInstruction(), and OneIPCPerformanceModel().

6.226.4.3 m_cpiDataCache

```
std::vector< SubsecondTime> OneIPCPerformanceModel::m_cpiDataCache [private]
```

Definition at line 21 of file oneipc_performance_model.h.

Referenced by handleInstruction(), and OneIPCPerformanceModel().

6.226.4.4 m_latency_cutoff

```
UInt64 OneIPCPerformanceModel::m_latency_cutoff [private]
```

Definition at line 17 of file oneipc_performance_model.h.

Referenced by handleInstruction(), and OneIPCPerformanceModel().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/ **oneipc_performance_model.h**
- common/performance_model/performance_models/ **oneipc_performance_model.cc**

6.227 Operand Class Reference

```
#include <operand.h>
```

Public Types

- enum **Type** { **REG**, **MEMORY**, **IMMEDIATE** }
- enum **Direction** { **READ**, **WRITE** }
- typedef **UInt64** **Value**

Public Member Functions

- **Operand** (const **Operand** & **src**)
- **Operand** (**Type** type, **Value** value=0, **Direction** direction= **READ**, const String &value_name=String(), bool mem_operand=false)
- String **toString** (void) const

Public Attributes

- **Type** **m_type**
- **Value** **m_value**
- **Direction** **m_direction**
- String **m_value_name**
- bool **m_mem_operand**

6.227.1 Detailed Description

Definition at line 10 of file operand.h.

6.227.2 Member Typedef Documentation

6.227.2.1 Value

`typedef` **`UInt64 Operand::Value`**

Definition at line 26 of file operand.h.

6.227.3 Member Enumeration Documentation

6.227.3.1 Direction

`enum` **`Operand::Direction`**

Enumerator

READ	
WRITE	

Definition at line 20 of file operand.h.

6.227.3.2 Type

```
enum Operand::Type
```

Enumerator

REG	
MEMORY	
IMMEDIATE	

Definition at line 13 of file operand.h.

6.227.4 Constructor & Destructor Documentation

6.227.4.1 Operand() [1/2]

```
Operand::Operand (
            const  Operand & src )  [inline]
```

Definition at line 28 of file operand.h.

6.227.4.2 Operand() [2/2]

```
Operand::Operand (
             Type type,
             Value value = 0,
             Direction direction =  READ,
            const String & value_name = String(),
            bool mem_operand = false )  [inline]
```

Definition at line 31 of file operand.h.

6.227.5 Member Function Documentation

6.227.5.1 toString()

```
String Operand::toString (
            void  ) const  [inline]
```

Definition at line 34 of file operand.h.

References IMMEDIATE, m_direction, m_mem_operand, m_type, m_value, m_value_name, MEMORY, READ, R↩EG, and WRITE.

6.227.6 Member Data Documentation

6.227.6.1 m_direction

```
Direction Operand::m_direction
```

Definition at line 64 of file operand.h.

Referenced by OneIPCPerformanceModel::handleInstruction(), MicroOpPerformanceModel::handleInstruction(), and toString().

6.227.6.2 m_mem_operand

```
bool Operand::m_mem_operand
```

Definition at line 66 of file operand.h.

Referenced by toString().

6.227.6.3 m_type

```
Type Operand::m_type
```

Definition at line 62 of file operand.h.

Referenced by OneIPCPerformanceModel::handleInstruction(), MicroOpPerformanceModel::handleInstruction(), and toString().

6.227.6.4 m_value

```
Value Operand::m_value
```

Definition at line 63 of file operand.h.

Referenced by toString().

6.227.6.5 m_value_name

```
String Operand::m_value_name
```

Definition at line 65 of file operand.h.

Referenced by toString().

The documentation for this class was generated from the following file:

- common/performance_model/ **operand.h**

6.228 config::parserError Class Reference

```
#include <config_exceptions.hpp>
```

Inheritance diagram for config::parserError:

Public Member Functions

- **parserError** (const String &leftover)
- virtual ~**parserError** () throw ()
- virtual const char * **what** () const throw ()

Private Attributes

- String **m_leftover**

6.228.1 Detailed Description

Definition at line 14 of file config_exceptions.hpp.

6.228.2 Constructor & Destructor Documentation

6.228.2.1 parserError()

```
config::parserError::parserError (
            const String & leftover )  [inline]
```

Definition at line 17 of file config_exceptions.hpp.

6.228.2.2 ~parserError()

```
virtual config::parserError::~parserError ( ) throw ( )   [inline], [virtual]
```

Definition at line 23 of file config_exceptions.hpp.

6.228.3 Member Function Documentation

6.228.3.1 what()

```
virtual const char* config::parserError::what ( ) const throw ( )   [inline], [virtual]
```

Definition at line 25 of file config_exceptions.hpp.

References m_leftover.

6.228.4 Member Data Documentation

6.228.4.1 m_leftover

```
String config::parserError::m_leftover  [private]
```

Definition at line 31 of file config_exceptions.hpp.

Referenced by what().

The documentation for this class was generated from the following file:

- common/config/ **config_exceptions.hpp**

6.229 PentiumMBimodalTable Class Reference

`#include <pentium_m_bimodal_table.h>`

Inheritance diagram for PentiumMBimodalTable:

Public Member Functions

- **PentiumMBimodalTable** ()

6.229.1 Detailed Description

Definition at line 6 of file pentium_m_bimodal_table.h.

6.229.2 Constructor & Destructor Documentation

6.229.2.1 PentiumMBimodalTable()

`PentiumMBimodalTable::PentiumMBimodalTable ( ) [inline]`

Definition at line 16 of file pentium_m_bimodal_table.h.

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **pentium_m_bimodal_table.h**

6.230 PentiumMBranchPredictor Class Reference

`#include <pentium_m_branch_predictor.h>`

Inheritance diagram for PentiumMBranchPredictor:

Public Member Functions

- **PentiumMBranchPredictor** (String name, **core_id_t** core_id)
- ~**PentiumMBranchPredictor** ()
- bool **predict** (**IntPtr** ip, **IntPtr** target)
- void **update** (bool predicted, bool actual, **IntPtr** ip, **IntPtr** target)

Private Member Functions

- void **update_pir** (bool actual, **IntPtr** ip, **IntPtr** target, **BranchPredictorReturnValue::BranchType** branch_type)
- **IntPtr** **hash_function** (**IntPtr** ip, **IntPtr** pir)

Private Attributes

- **PentiumMGlobalPredictor** **m_global_predictor**
- **PentiumMBranchTargetBuffer** **m_btb**
- **PentiumMBimodalTable** **m_bimodal_table**
- **PentiumMLoopBranchPredictor** **m_lpb**
- **IntPtr** **m_pir**
- bool **m_last_gp_hit**
- bool **m_last_bm_pred**
- bool **m_last_lpb_hit**

Additional Inherited Members

6.230.1 Detailed Description

Definition at line 13 of file pentium_m_branch_predictor.h.

6.230.2 Constructor & Destructor Documentation

6.230.2.1 PentiumMBranchPredictor()

```
PentiumMBranchPredictor::PentiumMBranchPredictor (
            String name,
             core_id_t core_id )
```

Definition at line 5 of file pentium_m_branch_predictor.cc.

6.230.2.2 ~PentiumMBranchPredictor()

```
PentiumMBranchPredictor::~PentiumMBranchPredictor ( )
```

Definition at line 13 of file pentium_m_branch_predictor.cc.

6.230.3 Member Function Documentation

6.230.3.1 hash_function()

```
IntPtr PentiumMBranchPredictor::hash_function (
            IntPtr ip,
            IntPtr pir )  [private]
```

6.230.3.2 predict()

```
bool PentiumMBranchPredictor::predict (
            IntPtr ip,
            IntPtr target )  [virtual]
```

Implements **BranchPredictor** (p. 125).

Definition at line 17 of file pentium_m_branch_predictor.cc.

References BranchPredictorReturnValue::hit, GlobalPredictor::lookup(), LoopBranchPredictor::lookup(), PentiumMBranchTargetBuffer::lookup(), m_bimodal_table, m_btb, m_global_predictor, m_last_bm_pred, m_last_gp_hit, m_last_lpb_hit, m_lpb, m_pir, SimpleBimodalTable::predict(), and BranchPredictorReturnValue::prediction.

6.230.3.3 update()

```
void PentiumMBranchPredictor::update (
            bool predicted,
            bool actual,
            IntPtr ip,
            IntPtr target )  [virtual]
```

Implements **BranchPredictor** (p. 126).

Definition at line 48 of file pentium_m_branch_predictor.cc.

References BranchPredictorReturnValue::ConditionalBranch, GlobalPredictor::evict(), m_bimodal_table, m_btb, m_global_predictor, m_last_bm_pred, m_last_gp_hit, m_last_lpb_hit, m_lpb, m_pir, GlobalPredictor::update(), SimpleBimodalTable::update(), PentiumMBranchTargetBuffer::update(), LoopBranchPredictor::update(), update_pir(), and BranchPredictor::updateCounters().

6.230.3.4 update_pir()

```
void PentiumMBranchPredictor::update_pir (
            bool actual,
             IntPtr ip,
             IntPtr target,
             BranchPredictorReturnValue::BranchType branch_type )  [private]
```

Definition at line 69 of file pentium_m_branch_predictor.cc.

References BranchPredictorReturnValue::ConditionalBranch, BranchPredictorReturnValue::IndirectBranch, and m_pir.

Referenced by update().

6.230.4 Member Data Documentation

6.230.4.1 m_bimodal_table

```
PentiumMBimodalTable PentiumMBranchPredictor::m_bimodal_table  [private]
```

Definition at line 30 of file pentium_m_branch_predictor.h.

Referenced by predict(), and update().

6.230.4.2 m_btb

```
PentiumMBranchTargetBuffer PentiumMBranchPredictor::m_btb  [private]
```

Definition at line 29 of file pentium_m_branch_predictor.h.

Referenced by predict(), and update().

6.230.4.3 m_global_predictor

```
PentiumMGlobalPredictor PentiumMBranchPredictor::m_global_predictor  [private]
```

Definition at line 28 of file pentium_m_branch_predictor.h.

Referenced by predict(), and update().

6.230.4.4 m_last_bm_pred

```
bool PentiumMBranchPredictor::m_last_bm_pred  [private]
```

Definition at line 36 of file pentium_m_branch_predictor.h.

Referenced by predict(), and update().

6.230.4.5 m_last_gp_hit

```
bool PentiumMBranchPredictor::m_last_gp_hit  [private]
```

Definition at line 35 of file pentium_m_branch_predictor.h.

Referenced by predict(), and update().

6.230.4.6 m_last_lpb_hit

```
bool PentiumMBranchPredictor::m_last_lpb_hit  [private]
```

Definition at line 37 of file pentium_m_branch_predictor.h.

Referenced by predict(), and update().

6.230.4.7 m_lpb

```
PentiumMLoopBranchPredictor PentiumMBranchPredictor::m_lpb  [private]
```

Definition at line 31 of file pentium_m_branch_predictor.h.

Referenced by predict(), and update().

6.230.4.8 m_pir

```
IntPtr PentiumMBranchPredictor::m_pir  [private]
```

Definition at line 33 of file pentium_m_branch_predictor.h.

Referenced by predict(), update(), and update_pir().

The documentation for this class was generated from the following files:

- common/performance_model/branch_predictors/ **pentium_m_branch_predictor.h**
- common/performance_model/branch_predictors/ **pentium_m_branch_predictor.cc**

6.231 PentiumMBranchTargetBuffer Class Reference

`#include <pentium_m_branch_target_buffer.h>`

Inheritance diagram for PentiumMBranchTargetBuffer:

Classes

- class **Way**

Public Member Functions

- **PentiumMBranchTargetBuffer** ()
- bool **predict** (**IntPtr** ip, **IntPtr** target)
- **BranchPredictorReturnValue** **lookup** (**IntPtr** ip, **IntPtr** target)
- void **update** (bool predicted, bool actual, **IntPtr** ip, **IntPtr** target)

Private Attributes

- std::vector< **Way** > **m_ways**
- **UInt64** **m_lru_use_count**

Additional Inherited Members

6.231.1 Detailed Description

Definition at line 11 of file pentium_m_branch_target_buffer.h.

6.231.2 Constructor & Destructor Documentation

6.231.2.1 PentiumMBranchTargetBuffer()

`PentiumMBranchTargetBuffer::PentiumMBranchTargetBuffer ( ) [inline]`

Definition at line 33 of file pentium_m_branch_target_buffer.h.

6.231.3 Member Function Documentation

6.231.3.1 lookup()

```
BranchPredictorReturnValue PentiumMBranchTargetBuffer::lookup (
            IntPtr ip,
            IntPtr target )  [inline]
```

Definition at line 43 of file pentium_m_branch_target_buffer.h.

References BranchPredictorReturnValue::InvalidBranch, IP_TO_INDEX, IP_TO_TAGOFF, m_ways, and NUM_↩ WAYS.

Referenced by PentiumMBranchPredictor::predict().

6.231.3.2 predict()

```
bool PentiumMBranchTargetBuffer::predict (
            IntPtr ip,
            IntPtr target )  [inline], [virtual]
```

Implements **BranchPredictor** (p. 125).

Definition at line 38 of file pentium_m_branch_target_buffer.h.

6.231.3.3 update()

```
void PentiumMBranchTargetBuffer::update (
            bool predicted,
            bool actual,
            IntPtr ip,
            IntPtr target )  [inline], [virtual]
```

Implements **BranchPredictor** (p. 126).

Definition at line 62 of file pentium_m_branch_target_buffer.h.

References IP_TO_INDEX, IP_TO_TAGOFF, m_lru_use_count, m_ways, and NUM_WAYS.

Referenced by PentiumMBranchPredictor::update().

6.231.4 Member Data Documentation

6.231.4.1 m_lru_use_count

```
UInt64 PentiumMBranchTargetBuffer::m_lru_use_count  [private]
```

Definition at line 95 of file pentium_m_branch_target_buffer.h.

Referenced by update().

6.231.4.2 m_ways

```
std::vector< Way> PentiumMBranchTargetBuffer::m_ways  [private]
```

Definition at line 94 of file pentium_m_branch_target_buffer.h.

Referenced by lookup(), and update().

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **pentium_m_branch_target_buffer.h**

6.232 PentiumMGlobalPredictor Class Reference

```
#include <pentium_m_global_predictor.h>
```

Inheritance diagram for PentiumMGlobalPredictor:

Public Member Functions

- **PentiumMGlobalPredictor** ()

6.232.1 Detailed Description

Definition at line 6 of file pentium_m_global_predictor.h.

6.232.2 Constructor & Destructor Documentation

6.232.2.1 PentiumMGlobalPredictor()

```
PentiumMGlobalPredictor::PentiumMGlobalPredictor ( )  [inline]
```

Definition at line 16 of file pentium_m_global_predictor.h.

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **pentium_m_global_predictor.h**

6.233 PentiumMIndirectBranchTargetBuffer Class Reference

```
#include <pentium_m_indirect_branch_target_buffer.h>
```

Inheritance diagram for PentiumMIndirectBranchTargetBuffer:

Public Member Functions

- **PentiumMIndirectBranchTargetBuffer** ()

6.233.1 Detailed Description

Definition at line 6 of file pentium_m_indirect_branch_target_buffer.h.

6.233.2 Constructor & Destructor Documentation

6.233.2.1 PentiumMIndirectBranchTargetBuffer()

```
PentiumMIndirectBranchTargetBuffer::PentiumMIndirectBranchTargetBuffer ( )  [inline]
```

Definition at line 15 of file pentium_m_indirect_branch_target_buffer.h.

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **pentium_m_indirect_branch_target_buffer.h**

6.234 PentiumMLoopBranchPredictor Class Reference

`#include <pentium_m_loop_branch_predictor.h>`

Inheritance diagram for PentiumMLoopBranchPredictor:

Public Member Functions

- **PentiumMLoopBranchPredictor** ()

6.234.1 Detailed Description

Definition at line 6 of file pentium_m_loop_branch_predictor.h.

6.234.2 Constructor & Destructor Documentation

6.234.2.1 PentiumMLoopBranchPredictor()

`PentiumMLoopBranchPredictor::PentiumMLoopBranchPredictor ( ) [inline]`

Definition at line 14 of file pentium_m_loop_branch_predictor.h.

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **pentium_m_loop_branch_predictor.h**

6.235 PerformanceModel Class Reference

`#include <performance_model.h>`

Inheritance diagram for PerformanceModel:

Public Member Functions

- **PerformanceModel** (**Core** *core)
- virtual ~**PerformanceModel** ()
- void **queueInstruction** (**DynamicInstruction** *i)
- void **queuePseudoInstruction** (**PseudoInstruction** *i)
- void **handleIdleInstruction** (**PseudoInstruction** *i)
- void **iterate** ()
- virtual void **synchronize** ()
- **UInt64** **getInstructionCount** () const
- **SubsecondTime** **getElapsedTime** () const
- **SubsecondTime** **getNonIdleElapsedTime** () const
- void **countInstructions** (**IntPtr** address, **UInt32** count)
- void **handleMemoryLatency** (**SubsecondTime** latency, **HitWhere::where_t** hit_where)
- void **handleBranchMispredict** ()
- **BranchPredictor** * **getBranchPredictor** ()
- **BranchPredictor** const * **getConstBranchPredictor** () const
- **FastforwardPerformanceModel** * **getFastforwardPerformanceModel** ()
- **FastforwardPerformanceModel** const * **getFastforwardPerformanceModel** () const
- **DynamicInstruction** * **createDynamicInstruction** (**Instruction** *ins, **IntPtr** eip)
- void **traceInstruction** (const **DynamicMicroOp** *uop, **InstructionTracer::uop_times_t** *times)
- virtual void **barrierEnter** ()
- virtual void **barrierExit** ()
- void **disable** ()
- void **enable** ()
- bool **isEnabled** ()
- void **setHold** (bool hold)
- bool **isFastForward** ()
- void **setFastForward** (bool fastforward, bool detailed_sync=true)

Static Public Member Functions

- static **PerformanceModel** * **create** (**Core** *core)

Protected Types

- typedef **CircularQueue**< **DynamicInstruction** * > **InstructionQueue**

Protected Member Functions

- void **setElapsedTime** (**SubsecondTime** time)
- void **incrementElapsedTime** (**SubsecondTime** time)
- void **incrementIdleElapsedTime** (**SubsecondTime** time)
- **Core** * **getCore** ()

Protected Attributes

- **UInt64** **m_instruction_count**
- **ComponentTime** **m_elapsed_time**

Private Member Functions

- virtual void **handleInstruction** (**DynamicInstruction** *instruction)=0
- virtual void **notifyElapsedTimeUpdate** ()
- virtual void **enableDetailedModel** ()
- virtual void **disableDetailedModel** ()

Private Attributes

- **Core** * **m_core**
- **Allocator** * **m_dynins_alloc**
- bool **m_enabled**
- bool **m_fastforward**
- **FastforwardPerformanceModel** * **m_fastforward_model**
- bool **m_detailed_sync**
- bool **m_hold**
- **ComponentTime** **m_idle_elapsed_time**
- **SubsecondTime** **m_cpiStartTime**
- **SubsecondTime** **m_cpiSyncFutex**
- **SubsecondTime** **m_cpiSyncPthreadMutex**
- **SubsecondTime** **m_cpiSyncPthreadCond**
- **SubsecondTime** **m_cpiSyncPthreadBarrier**
- **SubsecondTime** **m_cpiSyncJoin**
- **SubsecondTime** **m_cpiSyncPause**
- **SubsecondTime** **m_cpiSyncSleep**
- **SubsecondTime** **m_cpiSyncSyscall**
- **SubsecondTime** **m_cpiSyncUnscheduled**
- **SubsecondTime** **m_cpiSyncDvfsTransition**
- **SubsecondTime** **m_cpiRecv**
- **InstructionQueue** **m_instruction_queue**
- **UInt32** **m_current_ins_index**
- **BranchPredictor** * **m_bp**
- **InstructionTracer** * **m_instruction_tracer**

Friends

- class **SpawnInstruction**
- class **FastforwardPerformanceModel**

6.235.1 Detailed Description

Definition at line 24 of file performance_model.h.

6.235.2 Member Typedef Documentation

6.235.2.1 InstructionQueue

```
typedef CircularQueue< DynamicInstruction*> PerformanceModel::InstructionQueue [protected]
```

Definition at line 97 of file performance_model.h.

6.235.3 Constructor & Destructor Documentation

6.235.3.1 PerformanceModel()

```
PerformanceModel::PerformanceModel (
            Core * core )
```

Definition at line 51 of file performance_model.cc.

References InstructionTracer::create(), BranchPredictor::create(), Core::getId(), m_bp, m_cpiRecv, m_cpiStartTime, m_cpiSyncDvfsTransition, m_cpiSyncFutex, m_cpiSyncJoin, m_cpiSyncPause, m_cpiSyncPthreadBarrier, m_cpiSyncPthreadCond, m_cpiSyncPthreadMutex, m_cpiSyncSleep, m_cpiSyncSyscall, m_cpiSyncUnscheduled, m_elapsed_time, m_idle_elapsed_time, m_instruction_count, m_instruction_tracer, and registerStatsMetric().

6.235.3.2 ~PerformanceModel()

```
PerformanceModel::~PerformanceModel ( ) [virtual]
```

Definition at line 94 of file performance_model.cc.

References m_bp, m_fastforward_model, and m_instruction_tracer.

6.235.4 Member Function Documentation

6.235.4.1 barrierEnter()

```
virtual void PerformanceModel::barrierEnter ( ) [inline], [virtual]
```

Definition at line 61 of file performance_model.h.

Referenced by BarrierSyncServer::synchronize().

6.235.4.2 barrierExit()

```
virtual void PerformanceModel::barrierExit ( )  [inline], [virtual]
```

Definition at line 62 of file performance_model.h.

Referenced by BarrierSyncServer::abortBarrier(), BarrierSyncServer::barrierRelease(), and BarrierSyncServer↩ ::synchronize().

6.235.4.3 countInstructions()

```
void PerformanceModel::countInstructions (
              IntPtr address,
              UInt32 count )
```

Definition at line 116 of file performance_model.cc.

References FastforwardPerformanceModel::countInstructions(), m_fastforward, and m_fastforward_model.

Referenced by Core::countInstructions().

6.235.4.4 create()

```
PerformanceModel * PerformanceModel::create (
              Core * core )  [static]
```

Definition at line 17 of file performance_model.cc.

References Core::getId(), and LOG_PRINT_ERROR.

Referenced by Core::Core().

6.235.4.5 createDynamicInstruction()

```
DynamicInstruction * PerformanceModel::createDynamicInstruction (
              Instruction * ins,
              IntPtr eip )
```

Definition at line 140 of file performance_model.cc.

References DynamicInstruction::alloc(), and m_dynins_alloc.

Referenced by InstructionModeling::handleInstruction(), TraceThread::handleInstructionDetailed(), and queue↩ PseudoInstruction().

6.235.4.6 disable()

```
void PerformanceModel::disable ( )
```

Definition at line 109 of file performance_model.cc.

References disableDetailedModel(), and m_enabled.

Referenced by Core::disablePerformanceModels().

6.235.4.7 disableDetailedModel()

```
virtual void PerformanceModel::disableDetailedModel ( )  [inline], [private], [virtual]
```

Reimplemented in **RobSmtPerformanceModel** (p. 1014).

Definition at line 112 of file performance_model.h.

Referenced by disable(), and setFastForward().

6.235.4.8 enable()

```
void PerformanceModel::enable ( )
```

Definition at line 102 of file performance_model.cc.

References enableDetailedModel(), and m_enabled.

Referenced by Core::enablePerformanceModels().

6.235.4.9 enableDetailedModel()

```
virtual void PerformanceModel::enableDetailedModel ( )  [inline], [private], [virtual]
```

Reimplemented in **RobSmtPerformanceModel** (p. 1014).

Definition at line 111 of file performance_model.h.

Referenced by enable(), and setFastForward().

6.235.4.10 getBranchPredictor()

```
BranchPredictor* PerformanceModel::getBranchPredictor ( )  [inline]
```

Definition at line 47 of file performance_model.h.

References m_bp.

Referenced by Core::accessBranchPredictor(), DynamicInstruction::getBranchCost(), and TraceThread::handle↩ EmuFunc().

6.235.4.11 getConstBranchPredictor()

```
BranchPredictor const* PerformanceModel::getConstBranchPredictor ( ) const  [inline]
```

Definition at line 48 of file performance_model.h.

References m_bp.

6.235.4.12 getCore()

```
Core* PerformanceModel::getCore ( )  [inline], [protected]
```

Definition at line 100 of file performance_model.h.

References m_core.

Referenced by handleIdleInstruction(), OneIPCPerformanceModel::handleInstruction(), MicroOpPerformance↩ Model::handleInstruction(), and RobSmtPerformanceModel::∼RobSmtPerformanceModel().

6.235.4.13 getElapsedTime()

```
SubsecondTime PerformanceModel::getElapsedTime ( ) const  [inline]
```

Definition at line 38 of file performance_model.h.

References ComponentTime::getElapsedTime(), and m_elapsed_time.

Referenced by SyncClient::__mutexLock(), SyncClient::barrierWait(), SpinLoopDetector::commitBCT(), Sync↩ Client::condBroadcast(), SyncClient::condSignal(), SyncClient::condWait(), MemoryManagerBase::core↩ InitiateMemoryAccessFast(), FastForwardPerformanceManager::disable(), SamplingManager::disableFast↩ Forward(), lite::emuClockGettime(), lite::emuGettimeofday(), SamplingManager::enableFastForward(), Sampling↩ Manager::getCoreHistoricCPI(), TraceThread::getCurrentTime(), getNonIdleElapsedTime(), InstructionModeling↩ ::handleBasicBlock(), handleCheckScheduled(), SyscallMdl::handleFutexCall(), handleIdleInstruction(), Trace↩ Thread::handleInstructionCountFunc(), handleRdtsc(), Core::initiateMemoryAccess(), ThreadManager::join↩ Thread(), SyncClient::mutexUnlock(), Network::netRecv(), Network::netSend(), ThreadManager::onThreadExit(), ThreadManager::onThreadStart(), lite::pthreadAfter(), lite::pthreadBefore(), FastForwardPerformanceManager↩ ::recalibrateInstructionsCallback(), SamplingManager::recalibrateInstructionsCallback(), SamplingManager↩ ::resetCoreHistoricCPIs(), TraceThread::run(), SyscallMdl::runEnter(), SyscallMdl::runExit(), setElapsedTime(), ThreadManager::spawnThread(), FastForwardPerformanceManager::step(), BarrierSyncClient::synchronize(), SchedulerPinnedBase::threadSetAffinity(), SchedulerPinnedBase::threadYield(), and PthreadEmu::updateState().

6.235.4.14 getFastforwardPerformanceModel() [1/2]

```
FastforwardPerformanceModel* PerformanceModel::getFastforwardPerformanceModel ( )  [inline]
```

Definition at line 50 of file performance_model.h.

References m_fastforward_model.

Referenced by PeriodicSampling::callbackDetailed(), SamplingManager::disableFastForward(), MagicServer::disablePerformance(), FastForwardPerformanceManager::recalibrateInstructionsCallback(), and SamplingManager::recalibrateInstructionsCallback().

6.235.4.15 getFastforwardPerformanceModel() [2/2]

```
FastforwardPerformanceModel const* PerformanceModel::getFastforwardPerformanceModel ( ) const
[inline]
```

Definition at line 51 of file performance_model.h.

References m_fastforward_model.

6.235.4.16 getInstructionCount()

```
UInt64 PerformanceModel::getInstructionCount ( ) const  [inline]
```

Definition at line 36 of file performance_model.h.

References m_instruction_count.

Referenced by IntervalTimer::dispatchInstruction(), IntervalTimer::dispatchWindow(), SamplingManager::getCoreHistoricCPI(), TraceThread::handleEmuFunc(), ThreadStatsManager::metricCallback(), SamplingManager::resetCoreHistoricCPIs(), and ThreadStatsManager::ThreadStats::update().

6.235.4.17 getNonIdleElapsedTime()

```
SubsecondTime PerformanceModel::getNonIdleElapsedTime ( ) const  [inline]
```

Definition at line 39 of file performance_model.h.

References getElapsedTime(), ComponentTime::getElapsedTime(), and m_idle_elapsed_time.

Referenced by MagicServer::disablePerformance(), SamplingManager::getCoreHistoricCPI(), ThreadStatsManager::metricCallback(), SamplingManager::resetCoreHistoricCPIs(), and ThreadStatsManager::ThreadStats::update().

6.235.4.18 handleBranchMispredict()

```
void PerformanceModel::handleBranchMispredict ( )
```

Definition at line 132 of file performance_model.cc.

References FastforwardPerformanceModel::handleBranchMispredict(), m_fastforward, and m_fastforward_model.

Referenced by handleBranchWarming(), TraceThread::handleCacheOnlyFunc(), and TraceThread::handleInstructionWarmup().

6.235.4.19 handleIdleInstruction()

```
void PerformanceModel::handleIdleInstruction (
              PseudoInstruction * i )
```

Definition at line 203 of file performance_model.cc.

References __attribute__, DelayInstruction::DVFS_TRANSITION, SyncInstruction::FUTEX, getCore(), PseudoInstruction::getCost(), DelayInstruction::getDelayType(), getElapsedTime(), SyncInstruction::getSyncType(), SyncInstruction::getTime(), Instruction::getType(), incrementIdleElapsedTime(), INST_DELAY, INST_RECV, INST_SYNC, SyncInstruction::JOIN, LOG_ASSERT_ERROR, LOG_PRINT_ERROR, m_cpiRecv, m_cpiSyncDvfsTransition, m_cpiSyncFutex, m_cpiSyncJoin, m_cpiSyncPause, m_cpiSyncPthreadBarrier, m_cpiSyncPthreadCond, m_cpiSyncPthreadMutex, m_cpiSyncSleep, m_cpiSyncSyscall, m_cpiSyncUnscheduled, m_detailed_sync, m_fastforward, m_fastforward_model, FastforwardPerformanceModel::notifyElapsedTimeUpdate(), SyncInstruction::PAUSE, SyncInstruction::PTHREAD_BARRIER, SyncInstruction::PTHREAD_COND, SyncInstruction::PTHREAD_MUTEX, SyncInstruction::SLEEP, SyncInstruction::SYSCALL, SyncInstruction::UNSCHEDULED, and SubsecondTime::Zero().

Referenced by queuePseudoInstruction().

6.235.4.20 handleInstruction()

```
virtual void PerformanceModel::handleInstruction (
              DynamicInstruction * instruction )  [private], [pure virtual]
```

Implemented in **MicroOpPerformanceModel** (p. 788), and **OneIPCPerformanceModel** (p. 887).

Referenced by iterate().

6.235.4.21 handleMemoryLatency()

```
void PerformanceModel::handleMemoryLatency (
              SubsecondTime latency,
              HitWhere::where_t hit_where )
```

Definition at line 124 of file performance_model.cc.

References FastforwardPerformanceModel::handleMemoryLatency(), m_fastforward, and m_fastforward_model.

Referenced by Core::accessMemoryFast(), and Core::initiateMemoryAccess().

6.235.4.22 incrementElapsedTime()

```
void PerformanceModel::incrementElapsedTime (
              SubsecondTime time )  [inline], [protected]
```

Definition at line 91 of file performance_model.h.

References ComponentTime::addLatency(), and m_elapsed_time.

Referenced by FastforwardPerformanceModel::incrementElapsedTime(), and incrementIdleElapsedTime().

6.235.4.23 incrementIdleElapsedTime()

```
void PerformanceModel::incrementIdleElapsedTime (
              SubsecondTime time )  [protected]
```

Definition at line 326 of file performance_model.cc.

References ComponentTime::addLatency(), incrementElapsedTime(), m_fastforward, m_fastforward_model, m_↩ idle_elapsed_time, FastforwardPerformanceModel::notifyElapsedTimeUpdate(), and notifyElapsedTimeUpdate().

Referenced by handleIdleInstruction(), and setElapsedTime().

6.235.4.24 isEnabled()

```
bool PerformanceModel::isEnabled ( )  [inline]
```

Definition at line 66 of file performance_model.h.

References m_enabled.

Referenced by Core::initiateMemoryAccess().

6.235.4.25 isFastForward()

```
bool PerformanceModel::isFastForward ( )  [inline]
```

Definition at line 69 of file performance_model.h.

References m_fastforward.

6.235.4.26 iterate()

```
void PerformanceModel::iterate ( )
```

Definition at line 294 of file performance_model.cc.

References CircularQueue< T >::front(), handleInstruction(), DynamicInstruction::instruction, Instruction::isIdle(), LOG_ASSERT_ERROR, m_enabled, m_fastforward, m_hold, m_instruction_queue, CircularQueue< T >::pop(), CircularQueue< T >::size(), and synchronize().

Referenced by InstructionModeling::handleBasicBlock(), TraceThread::handleInstructionCountFunc(), TraceThread::handleInstructionDetailed(), and CoreThread::run().

6.235.4.27 notifyElapsedTimeUpdate()

```
virtual void PerformanceModel::notifyElapsedTimeUpdate ( )  [inline], [private], [virtual]
```

Reimplemented in **MicroOpPerformanceModel** (p. 788), **RobSmtPerformanceModel** (p. 1014), **IntervalPerformanceModel** (p. 648), and **RobPerformanceModel** (p. 1011).

Definition at line 109 of file performance_model.h.

Referenced by incrementIdleElapsedTime(), and setFastForward().

6.235.4.28 queueInstruction()

```
void PerformanceModel::queueInstruction (
              DynamicInstruction * i )
```

Definition at line 186 of file performance_model.cc.

References m_enabled, m_fastforward, m_instruction_queue, and CircularQueue< T >::push().

Referenced by InstructionModeling::handleBasicBlock(), InstructionModeling::handleInstruction(), and TraceThread::handleInstructionDetailed().

6.235.4.29 queuePseudoInstruction()

```
void PerformanceModel::queuePseudoInstruction (
              PseudoInstruction * i )
```

Definition at line 145 of file performance_model.cc.

References createDynamicInstruction(), SpawnInstruction::getTime(), Instruction::getType(), handleIdleInstruction(), INST_SPAWN, Instruction::isIdle(), LOG_ASSERT_ERROR, m_enabled, m_fastforward, m_fastforward_model, m_instruction_queue, CircularQueue< T >::push(), FastforwardPerformanceModel::queuePseudoInstruction(), and setElapsedTime().

Referenced by SyncClient::__mutexLock(), ParametricDramDirectoryMSI::MemoryManager::accessTLB(), SyncClient::barrierWait(), SyncClient::condWait(), InstructionModeling::countInstructions(), handleCheckScheduled(), SyscallMdl::handleFutexCall(), TraceThread::handleInstructionCountFunc(), Core::initiateMemoryAccess(), ThreadManager::joinThread(), SyncClient::mutexUnlock(), Network::netRecv(), ThreadManager::onThreadStart(), Thread::reschedule(), SyscallMdl::runEnter(), SyscallMdl::runExit(), and TraceThread::unblock().

6.235.4.30 setElapsedTime()

```
void PerformanceModel::setElapsedTime (
              SubsecondTime time )  [protected]
```

Definition at line 339 of file performance_model.cc.

References getElapsedTime(), incrementIdleElapsedTime(), LOG_ASSERT_ERROR, m_cpiStartTime, m_cpi↩SyncUnscheduled, and SubsecondTime::Zero().

Referenced by queuePseudoInstruction().

6.235.4.31 setFastForward()

```
void PerformanceModel::setFastForward (
             bool fastforward,
             bool detailed_sync = true )  [inline]
```

Definition at line 70 of file performance_model.h.

References disableDetailedModel(), enableDetailedModel(), m_detailed_sync, m_fastforward, and notifyElapsed↩TimeUpdate().

Referenced by FastForwardPerformanceManager::disable(), SamplingManager::disableFastForward(), Sampling↩Manager::enableFastForward(), and FastForwardPerformanceManager::step().

6.235.4.32 setHold()

```
void PerformanceModel::setHold (
             bool hold )  [inline]
```

Definition at line 67 of file performance_model.h.

References m_hold.

6.235.4.33 synchronize()

```
void PerformanceModel::synchronize ( )  [virtual]
```

Reimplemented in **RobSmtPerformanceModel** (p. 1015).

Definition at line 319 of file performance_model.cc.

References Core::getClockSkewMinimizationClient(), m_core, ClockSkewMinimizationClient::synchronize(), and SubsecondTime::Zero().

Referenced by iterate().

6.235.4.34 traceInstruction()

```
void PerformanceModel::traceInstruction (
             const  DynamicMicroOp * uop,
              InstructionTracer::uop_times_t * times )  [inline]
```

Definition at line 55 of file performance_model.h.

References m_instruction_tracer, and InstructionTracer::traceInstruction().

Referenced by IntervalTimer::dispatchWindow(), RobTimer::doCommit(), and RobSmtTimer::doCommit().

6.235.5 Friends And Related Function Documentation

6.235.5.1 FastforwardPerformanceModel

```
friend class  FastforwardPerformanceModel  [friend]
```

Definition at line 88 of file performance_model.h.

6.235.5.2 SpawnInstruction

```
friend class  SpawnInstruction  [friend]
```

Definition at line 87 of file performance_model.h.

6.235.6 Member Data Documentation

6.235.6.1 m_bp

```
 BranchPredictor* PerformanceModel::m_bp  [private]
```

Definition at line 150 of file performance_model.h.

Referenced by getBranchPredictor(), getConstBranchPredictor(), PerformanceModel(), and ~PerformanceModel().

6.235.6.2 m_core

```
Core* PerformanceModel::m_core  [private]
```

Definition at line 114 of file performance_model.h.

Referenced by getCore(), and synchronize().

6.235.6.3 m_cpiRecv

```
SubsecondTime PerformanceModel::m_cpiRecv  [private]
```

Definition at line 144 of file performance_model.h.

Referenced by handleIdleInstruction(), and PerformanceModel().

6.235.6.4 m_cpiStartTime

```
SubsecondTime PerformanceModel::m_cpiStartTime  [private]
```

Definition at line 132 of file performance_model.h.

Referenced by PerformanceModel(), and setElapsedTime().

6.235.6.5 m_cpiSyncDvfsTransition

```
SubsecondTime PerformanceModel::m_cpiSyncDvfsTransition  [private]
```

Definition at line 143 of file performance_model.h.

Referenced by handleIdleInstruction(), and PerformanceModel().

6.235.6.6 m_cpiSyncFutex

```
SubsecondTime PerformanceModel::m_cpiSyncFutex  [private]
```

Definition at line 134 of file performance_model.h.

Referenced by handleIdleInstruction(), and PerformanceModel().

6.235.6.7 m_cpiSyncJoin

SubsecondTime PerformanceModel::m_cpiSyncJoin [private]

Definition at line 138 of file performance_model.h.

Referenced by handleIdleInstruction(), and PerformanceModel().

6.235.6.8 m_cpiSyncPause

SubsecondTime PerformanceModel::m_cpiSyncPause [private]

Definition at line 139 of file performance_model.h.

Referenced by handleIdleInstruction(), and PerformanceModel().

6.235.6.9 m_cpiSyncPthreadBarrier

SubsecondTime PerformanceModel::m_cpiSyncPthreadBarrier [private]

Definition at line 137 of file performance_model.h.

Referenced by handleIdleInstruction(), and PerformanceModel().

6.235.6.10 m_cpiSyncPthreadCond

SubsecondTime PerformanceModel::m_cpiSyncPthreadCond [private]

Definition at line 136 of file performance_model.h.

Referenced by handleIdleInstruction(), and PerformanceModel().

6.235.6.11 m_cpiSyncPthreadMutex

SubsecondTime PerformanceModel::m_cpiSyncPthreadMutex [private]

Definition at line 135 of file performance_model.h.

Referenced by handleIdleInstruction(), and PerformanceModel().

6.235.6.12 m_cpiSyncSleep

SubsecondTime PerformanceModel::m_cpiSyncSleep [private]

Definition at line 140 of file performance_model.h.

Referenced by handleIdleInstruction(), and PerformanceModel().

6.235.6.13 m_cpiSyncSyscall

SubsecondTime PerformanceModel::m_cpiSyncSyscall [private]

Definition at line 141 of file performance_model.h.

Referenced by handleIdleInstruction(), and PerformanceModel().

6.235.6.14 m_cpiSyncUnscheduled

SubsecondTime PerformanceModel::m_cpiSyncUnscheduled [private]

Definition at line 142 of file performance_model.h.

Referenced by handleIdleInstruction(), PerformanceModel(), and setElapsedTime().

6.235.6.15 m_current_ins_index

UInt32 PerformanceModel::m_current_ins_index [private]

Definition at line 148 of file performance_model.h.

6.235.6.16 m_detailed_sync

bool PerformanceModel::m_detailed_sync [private]

Definition at line 121 of file performance_model.h.

Referenced by handleIdleInstruction(), and setFastForward().

6.235.6.17 m_dynins_alloc

```
Allocator* PerformanceModel::m_dynins_alloc [private]
```

Definition at line 115 of file performance_model.h.

Referenced by createDynamicInstruction().

6.235.6.18 m_elapsed_time

```
ComponentTime PerformanceModel::m_elapsed_time [protected]
```

Definition at line 128 of file performance_model.h.

Referenced by getElapsedTime(), OneIPCPerformanceModel::handleInstruction(), MicroOpPerformance↩ Model::handleInstruction(), incrementElapsedTime(), RobPerformanceModel::notifyElapsedTimeUpdate(), IntervalPerformanceModel::notifyElapsedTimeUpdate(), RobSmtPerformanceModel::notifyElapsedTimeUpdate(), PerformanceModel(), RobPerformanceModel::simulate(), and RobSmtPerformanceModel::simulate().

6.235.6.19 m_enabled

```
bool PerformanceModel::m_enabled [private]
```

Definition at line 117 of file performance_model.h.

Referenced by disable(), enable(), isEnabled(), iterate(), queueInstruction(), and queuePseudoInstruction().

6.235.6.20 m_fastforward

```
bool PerformanceModel::m_fastforward [private]
```

Definition at line 119 of file performance_model.h.

Referenced by countInstructions(), handleBranchMispredict(), handleIdleInstruction(), handleMemoryLatency(), incrementIdleElapsedTime(), isFastForward(), iterate(), queueInstruction(), queuePseudoInstruction(), and set↩ FastForward().

6.235.6.21 m_fastforward_model

```
FastforwardPerformanceModel* PerformanceModel::m_fastforward_model [private]
```

Definition at line 120 of file performance_model.h.

Referenced by countInstructions(), getFastforwardPerformanceModel(), handleBranchMispredict(), handle↩ IdleInstruction(), handleMemoryLatency(), incrementIdleElapsedTime(), queuePseudoInstruction(), and ∼↩ PerformanceModel().

6.235.6.22 m_hold

```
bool PerformanceModel::m_hold  [private]
```

Definition at line 123 of file performance_model.h.

Referenced by iterate(), and setHold().

6.235.6.23 m_idle_elapsed_time

```
ComponentTime PerformanceModel::m_idle_elapsed_time  [private]
```

Definition at line 130 of file performance_model.h.

Referenced by getNonIdleElapsedTime(), incrementIdleElapsedTime(), and PerformanceModel().

6.235.6.24 m_instruction_count

```
UInt64 PerformanceModel::m_instruction_count  [protected]
```

Definition at line 126 of file performance_model.h.

Referenced by getInstructionCount(), OneIPCPerformanceModel::handleInstruction(), MicroOpPerformance↩Model::handleInstruction(), and PerformanceModel().

6.235.6.25 m_instruction_queue

```
InstructionQueue PerformanceModel::m_instruction_queue  [private]
```

Definition at line 146 of file performance_model.h.

Referenced by iterate(), queueInstruction(), and queuePseudoInstruction().

6.235.6.26 m_instruction_tracer

```
InstructionTracer* PerformanceModel::m_instruction_tracer  [private]
```

Definition at line 152 of file performance_model.h.

Referenced by PerformanceModel(), traceInstruction(), and ∼PerformanceModel().

The documentation for this class was generated from the following files:

- common/performance_model/ **performance_model.h**
- common/performance_model/ **performance_model.cc**

6.236 PeriodicSampling Class Reference

`#include <periodic_sampling.h>`

Inheritance diagram for PeriodicSampling:

Public Member Functions

- **PeriodicSampling** (**SamplingManager** *sampling_manager)
- virtual void **callbackDetailed** (**SubsecondTime** now)
- virtual void **callbackFastForward** (**SubsecondTime** now, bool in_warmup)

Protected Member Functions

- bool **stepFastForward** (**SubsecondTime** time, bool in_warmup)

Protected Attributes

- **SubsecondTime m_detailed_interval**
- **SubsecondTime m_fastforward_interval**
- **SubsecondTime m_fastforward_sync_interval**
- **SubsecondTime m_warmup_interval**
- **SubsecondTime m_detailed_warmup_interval**
- bool **m_detailed_sync**
- bool **m_constant_ipc**
- std::vector< double > **m_constant_ipcs**
- bool **m_random_placement**
- **SubsecondTime m_random_offset**
- **Random m_prng**
- bool **m_random_start**
- bool **m_random_first**
- **SubsecondTime m_periodic_last**
- **SubsecondTime m_fastforward_time_remaining**
- **SubsecondTime m_warmup_time_remaining**
- **SubsecondTime m_detailed_warmup_time_remaining**
- **UInt32 m_num_historic_cpi_intervals**
- std::vector< **CircularQueue**< **SubsecondTime** > * > **m_historic_cpi_intervals**
- int **m_dispatch_width**

Additional Inherited Members

6.236.1 Detailed Description

Definition at line 11 of file periodic_sampling.h.

6.236.2 Constructor & Destructor Documentation

6.236.2.1 PeriodicSampling()

```
PeriodicSampling::PeriodicSampling (
            SamplingManager * sampling_manager )
```

Definition at line 11 of file periodic_sampling.cc.

References LOG_ASSERT_ERROR, m_constant_ipc, m_constant_ipcs, m_fastforward_interval, m_fastforward↩
_sync_interval, m_historic_cpi_intervals, m_prng, m_random_offset, m_random_placement, m_random_start, m↩
_warmup_interval, Random::next(), Random::seed(), and SubsecondTime::Zero().

6.236.3 Member Function Documentation

6.236.3.1 callbackDetailed()

```
void PeriodicSampling::callbackDetailed (
            SubsecondTime now )  [virtual]
```

Implements **SamplingAlgorithm** (p. 1108).

Definition at line 66 of file periodic_sampling.cc.

References arithmetic_mean(), SamplingManager::getCoreHistoricCPI(), Core::getDvfsDomain(), Performance↩
Model::getFastforwardPerformanceModel(), SubsecondTime::getNS(), Core::getPerformanceModel(), Component↩
Period::getPeriod(), LOG_ASSERT_ERROR, m_constant_ipc, m_constant_ipcs, m_detailed_interval, m_↩
detailed_sync, m_detailed_warmup_interval, m_detailed_warmup_time_remaining, m_dispatch_width, m_↩
fastforward_interval, m_fastforward_sync_interval, m_fastforward_time_remaining, m_historic_cpi_intervals, m↩
_periodic_last, m_prng, m_random_offset, m_random_placement, m_random_start, SamplingAlgorithm::m_↩
sampling_manager, m_warmup_interval, m_warmup_time_remaining, SubsecondTime::MaxTime(), Random↩
::next(), SamplingManager::resetCoreHistoricCPIs(), FastforwardPerformanceModel::setCurrentCPI(), stepFast↩
Forward(), and SubsecondTime::Zero().

6.236.3.2 callbackFastForward()

```
void PeriodicSampling::callbackFastForward (
            SubsecondTime now,
           bool in_warmup )  [virtual]
```

Implements **SamplingAlgorithm** (p. 1108).

Definition at line 179 of file periodic_sampling.cc.

References SamplingManager::disableFastForward(), m_detailed_warmup_interval, m_detailed_warmup_time_↩
remaining, m_periodic_last, SamplingAlgorithm::m_sampling_manager, SamplingManager::resetCoreHistoricC↩
PIs(), stepFastForward(), and SubsecondTime::Zero().

6.236.3.3 stepFastForward()

```
bool PeriodicSampling::stepFastForward (
              SubsecondTime time,
             bool in_warmup )  [protected]
```

Definition at line 150 of file periodic_sampling.cc.

References SamplingManager::enableFastForward(), m_detailed_sync, m_fastforward_sync_interval, m_fastforward_time_remaining, SamplingAlgorithm::m_sampling_manager, m_warmup_time_remaining, and SubsecondTime::Zero().

Referenced by callbackDetailed(), and callbackFastForward().

6.236.4 Member Data Documentation

6.236.4.1 m_constant_ipc

```
bool PeriodicSampling::m_constant_ipc  [protected]
```

Definition at line 22 of file periodic_sampling.h.

Referenced by callbackDetailed(), and PeriodicSampling().

6.236.4.2 m_constant_ipcs

```
std::vector<double> PeriodicSampling::m_constant_ipcs  [protected]
```

Definition at line 23 of file periodic_sampling.h.

Referenced by callbackDetailed(), and PeriodicSampling().

6.236.4.3 m_detailed_interval

```
SubsecondTime PeriodicSampling::m_detailed_interval  [protected]
```

Definition at line 14 of file periodic_sampling.h.

Referenced by callbackDetailed().

6.236.4.4 m_detailed_sync

```
bool PeriodicSampling::m_detailed_sync  [protected]
```

Definition at line 20 of file periodic_sampling.h.

Referenced by callbackDetailed(), and stepFastForward().

6.236.4.5 m_detailed_warmup_interval

```
SubsecondTime PeriodicSampling::m_detailed_warmup_interval  [protected]
```

Definition at line 18 of file periodic_sampling.h.

Referenced by callbackDetailed(), and callbackFastForward().

6.236.4.6 m_detailed_warmup_time_remaining

```
SubsecondTime PeriodicSampling::m_detailed_warmup_time_remaining  [protected]
```

Definition at line 35 of file periodic_sampling.h.

Referenced by callbackDetailed(), and callbackFastForward().

6.236.4.7 m_dispatch_width

```
int PeriodicSampling::m_dispatch_width  [protected]
```

Definition at line 40 of file periodic_sampling.h.

Referenced by callbackDetailed().

6.236.4.8 m_fastforward_interval

```
SubsecondTime PeriodicSampling::m_fastforward_interval  [protected]
```

Definition at line 15 of file periodic_sampling.h.

Referenced by callbackDetailed(), and PeriodicSampling().

6.236.4.9 m_fastforward_sync_interval

SubsecondTime PeriodicSampling::m_fastforward_sync_interval [protected]

Definition at line 16 of file periodic_sampling.h.

Referenced by callbackDetailed(), PeriodicSampling(), and stepFastForward().

6.236.4.10 m_fastforward_time_remaining

SubsecondTime PeriodicSampling::m_fastforward_time_remaining [protected]

Definition at line 33 of file periodic_sampling.h.

Referenced by callbackDetailed(), and stepFastForward().

6.236.4.11 m_historic_cpi_intervals

std::vector< **CircularQueue**< **SubsecondTime**>* > PeriodicSampling::m_historic_cpi_intervals [protected]

Definition at line 38 of file periodic_sampling.h.

Referenced by callbackDetailed(), and PeriodicSampling().

6.236.4.12 m_num_historic_cpi_intervals

UInt32 PeriodicSampling::m_num_historic_cpi_intervals [protected]

Definition at line 37 of file periodic_sampling.h.

6.236.4.13 m_periodic_last

SubsecondTime PeriodicSampling::m_periodic_last [protected]

Definition at line 32 of file periodic_sampling.h.

Referenced by callbackDetailed(), and callbackFastForward().

6.236.4.14 m_prng

Random PeriodicSampling::m_prng [protected]

Definition at line 27 of file periodic_sampling.h.

Referenced by callbackDetailed(), and PeriodicSampling().

6.236.4.15 m_random_first

bool PeriodicSampling::m_random_first [protected]

Definition at line 30 of file periodic_sampling.h.

6.236.4.16 m_random_offset

SubsecondTime PeriodicSampling::m_random_offset [protected]

Definition at line 26 of file periodic_sampling.h.

Referenced by callbackDetailed(), and PeriodicSampling().

6.236.4.17 m_random_placement

bool PeriodicSampling::m_random_placement [protected]

Definition at line 25 of file periodic_sampling.h.

Referenced by callbackDetailed(), and PeriodicSampling().

6.236.4.18 m_random_start

bool PeriodicSampling::m_random_start [protected]

Definition at line 29 of file periodic_sampling.h.

Referenced by callbackDetailed(), and PeriodicSampling().

6.236.4.19 m_warmup_interval

SubsecondTime PeriodicSampling::m_warmup_interval [protected]

Definition at line 17 of file periodic_sampling.h.

Referenced by callbackDetailed(), and PeriodicSampling().

6.236.4.20 m_warmup_time_remaining

SubsecondTime PeriodicSampling::m_warmup_time_remaining [protected]

Definition at line 34 of file periodic_sampling.h.

Referenced by callbackDetailed(), and stepFastForward().

The documentation for this class was generated from the following files:

- common/sampling/ **periodic_sampling.h**
- common/sampling/ **periodic_sampling.cc**

6.237 _SetLock::PersetLock Class Reference

Public Member Functions

- **PersetLock** ()
- void **acquire** ()
- void **release** ()

Private Attributes

- pthread_mutex_t **_mutx**

6.237.1 Detailed Description

Definition at line 24 of file setlock.h.

6.237.2 Constructor & Destructor Documentation

6.237.2.1 PersetLock()

```
_SetLock::PersetLock::PersetLock ( )  [inline]
```

Definition at line 27 of file setlock.h.

6.237.3 Member Function Documentation

6.237.3.1 acquire()

```
void _SetLock::PersetLock::acquire ( )  [inline]
```

Definition at line 28 of file setlock.h.

References _mutx.

6.237.3.2 release()

```
void _SetLock::PersetLock::release ( )  [inline]
```

Definition at line 29 of file setlock.h.

References _mutx.

6.237.4 Member Data Documentation

6.237.4.1 _mutx

```
pthread_mutex_t _SetLock::PersetLock::_mutx  [private]
```

Definition at line 31 of file setlock.h.

Referenced by acquire(), and release().

The documentation for this class was generated from the following file:

- common/misc/ **setlock.h**

6.238 PinLock Class Reference

`#include <pin_lock.h>`

Inheritance diagram for PinLock:

Public Member Functions

- **PinLock** ()
- ~**PinLock** ()
- void **acquire** ()
- void **release** ()

Private Attributes

- PIN_LOCK **_lock**

6.238.1 Detailed Description

Definition at line 6 of file pin_lock.h.

6.238.2 Constructor & Destructor Documentation

6.238.2.1 PinLock()

`PinLock::PinLock ( )`

Definition at line 3 of file pin_lock.cc.

References _lock.

6.238.2.2 ~PinLock()

`PinLock::~PinLock ( )`

Definition at line 8 of file pin_lock.cc.

6.238.3 Member Function Documentation

6.238.3.1 acquire()

```
void PinLock::acquire ( )  [virtual]
```

Implements **LockImplementation** (p. 684).

Definition at line 12 of file pin_lock.cc.

References _lock.

6.238.3.2 release()

```
void PinLock::release ( )  [virtual]
```

Implements **LockImplementation** (p. 685).

Definition at line 17 of file pin_lock.cc.

References _lock.

6.238.4 Member Data Documentation

6.238.4.1 _lock

```
PIN_LOCK PinLock::_lock  [private]
```

Definition at line 16 of file pin_lock.h.

Referenced by acquire(), PinLock(), and release().

The documentation for this class was generated from the following files:

- pin/ **pin_lock.h**
- pin/ **pin_lock.cc**

6.239 PinThread Class Reference

`#include <pin_thread.h>`

Inheritance diagram for PinThread:

Public Member Functions

- **PinThread** (**ThreadFunc** func, void ∗param)
- ∼**PinThread** ()
- void **run** ()

Private Attributes

- THREADID **m_thread_p**
- **_Thread::ThreadFunc m_func**
- void ∗ **m_param**

Static Private Attributes

- static const int **STACK_SIZE** =65536

Additional Inherited Members

6.239.1 Detailed Description

Definition at line 7 of file pin_thread.h.

6.239.2 Constructor & Destructor Documentation

6.239.2.1 PinThread()

```
PinThread::PinThread (
              ThreadFunc func,
             void * param )
```

Definition at line 4 of file pin_thread.cc.

6.239.2.2 ~PinThread()

```
PinThread::~PinThread ( )
```

Definition at line 11 of file pin_thread.cc.

6.239.3 Member Function Documentation

6.239.3.1 run()

```
void PinThread::run ( )  [virtual]
```

Implements **_Thread** (p. 70).

Definition at line 15 of file pin_thread.cc.

References m_func, m_param, and m_thread_p.

6.239.4 Member Data Documentation

6.239.4.1 m_func

```
_Thread::ThreadFunc PinThread::m_func  [private]
```

Definition at line 18 of file pin_thread.h.

Referenced by run().

6.239.4.2 m_param

```
void* PinThread::m_param  [private]
```

Definition at line 19 of file pin_thread.h.

Referenced by run().

6.239.4.3 m_thread_p

```
THREADID PinThread::m_thread_p  [private]
```

Definition at line 17 of file pin_thread.h.

Referenced by run().

6.239.4.4 STACK_SIZE

```
const int PinThread::STACK_SIZE =65536  [static], [private]
```

Definition at line 15 of file pin_thread.h.

The documentation for this class was generated from the following files:

- pin/ **pin_thread.h**
- pin/ **pin_thread.cc**

6.240 PinTLS Class Reference

Inheritance diagram for PinTLS:

Public Member Functions

- **PinTLS** ()
- ∼**PinTLS** ()
- void ∗ **get** (int thread_id)
- const void ∗ **get** (int thread_id) const
- void **set** (void ∗vp)

Private Attributes

- TLS_KEY **m_key**

Additional Inherited Members

6.240.1 Detailed Description

Definition at line 5 of file pin_tls.cc.

6.240.2 Constructor & Destructor Documentation

6.240.2.1 PinTLS()

```
PinTLS::PinTLS ( )  [inline]
```

Definition at line 8 of file pin_tls.cc.

References m_key.

6.240.2.2 ~PinTLS()

```
PinTLS::~PinTLS ( )  [inline]
```

Definition at line 13 of file pin_tls.cc.

References m_key.

6.240.3 Member Function Documentation

6.240.3.1 get() [1/2]

```
void* PinTLS::get (
            int thread_id )  [inline], [virtual]
```

Implements **TLS** (p. 1422).

Definition at line 18 of file pin_tls.cc.

References m_key.

6.240.3.2 get() [2/2]

```
const void* PinTLS::get (
            int thread_id ) const  [inline], [virtual]
```

Implements **TLS** (p. 1421).

Definition at line 26 of file pin_tls.cc.

6.240.3.3 set()

```
void PinTLS::set (
            void * vp )  [inline], [virtual]
```

Implements **TLS** (p. 1423).

Definition at line 31 of file pin_tls.cc.

References __attribute__, LOG_ASSERT_ERROR, LOG_PRINT, and m_key.

6.240.4 Member Data Documentation

6.240.4.1 m_key

```
TLS_KEY PinTLS::m_key  [private]
```

Definition at line 39 of file pin_tls.cc.

Referenced by get(), PinTLS(), set(), and ~PinTLS().

The documentation for this class was generated from the following file:

- pin/ **pin_tls.cc**

6.241 Pow2< N > Class Template Reference

```
#include <saturating_predictor.h>
```

Public Types

- enum { **pow** = 2 * Pow2<N-1>::pow }

6.241.1 Detailed Description

template<int N>
class Pow2< N >

Definition at line 8 of file saturating_predictor.h.

6.241.2 Member Enumeration Documentation

6.241.2.1 anonymous enum

```
template<int N>
anonymous enum
```

Enumerator

pow	

Definition at line 11 of file saturating_predictor.h.

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **saturating_predictor.h**

6.242 Pow2< 0 > Class Reference

`#include <saturating_predictor.h>`

Public Types

- enum { **pow** = 1 }

6.242.1 Detailed Description

Definition at line 14 of file saturating_predictor.h.

6.242.2 Member Enumeration Documentation

6.242.2.1 anonymous enum

`anonymous enum`

Enumerator

pow	

Definition at line 17 of file saturating_predictor.h.

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **saturating_predictor.h**

6.243 ParametricDramDirectoryMSI::Prefetch Class Reference

`#include <cache_cntlr.h>`

Public Types

- enum **prefetch_type_t** { **NONE**, **OWN**, **OTHER** }

6.243.1 Detailed Description

Definition at line 65 of file cache_cntlr.h.

6.243.2 Member Enumeration Documentation

6.243.2.1 prefetch_type_t

enum **ParametricDramDirectoryMSI::Prefetch::prefetch_type_t**

Enumerator

NONE	
OWN	
OTHER	

Definition at line 68 of file cache_cntlr.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_cntlr.h**

6.244 Prefetcher Class Reference

#include <prefetcher.h>

Inheritance diagram for Prefetcher:

Public Member Functions

- virtual std::vector< **IntPtr** > **getNextAddress** (**IntPtr** current_address, **core_id_t** core_id)=0

Static Public Member Functions

- static **Prefetcher** * **createPrefetcher** (String type, String configName, **core_id_t** core_id, **UInt32** shared_cores)

6.244.1 Detailed Description

Definition at line 8 of file prefetcher.h.

6.244.2 Member Function Documentation

6.244.2.1 createPrefetcher()

```
Prefetcher * Prefetcher::createPrefetcher (
            String type,
            String configName,
             core_id_t core_id,
             UInt32 shared_cores )  [static]
```

Definition at line 8 of file prefetcher.cc.

References LOG_PRINT_ERROR.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), and DramCache::DramCache().

6.244.2.2 getNextAddress()

```
virtual std::vector< IntPtr> Prefetcher::getNextAddress (
             IntPtr current_address,
             core_id_t core_id )  [pure virtual]
```

Implemented in **GhbPrefetcher** (p. 585), and **SimplePrefetcher** (p. 1219).

Referenced by DramCache::callPrefetcher(), and ParametricDramDirectoryMSI::CacheCntlr::trainPrefetcher().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **prefetcher.h**
- common/core/memory_subsystem/parametric_dram_directory_msi/ **prefetcher.cc**

6.245 PrL1CacheBlockInfo Class Reference

`#include <pr_l1_cache_block_info.h>`

Inheritance diagram for PrL1CacheBlockInfo:

Public Member Functions

- **PrL1CacheBlockInfo** (**IntPtr** tag=∼0, **CacheState::cstate_t** cstate= **CacheState::INVALID**)
- ∼**PrL1CacheBlockInfo** ()

Additional Inherited Members

6.245.1 Detailed Description

Definition at line 7 of file pr_l1_cache_block_info.h.

6.245.2 Constructor & Destructor Documentation

6.245.2.1 PrL1CacheBlockInfo()

```
PrL1CacheBlockInfo::PrL1CacheBlockInfo (
              IntPtr tag = ∼0,
              CacheState::cstate_t cstate =  CacheState::INVALID )  [inline]
```

Definition at line 10 of file pr_l1_cache_block_info.h.

6.245.2.2 ∼PrL1CacheBlockInfo()

```
PrL1CacheBlockInfo::∼PrL1CacheBlockInfo ( )  [inline]
```

Definition at line 15 of file pr_l1_cache_block_info.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/cache/ **pr_l1_cache_block_info.h**

6.246 PrL2CacheBlockInfo Class Reference

`#include <pr_l2_cache_block_info.h>`

Inheritance diagram for PrL2CacheBlockInfo:

Public Member Functions

- **PrL2CacheBlockInfo** (**IntPtr** tag=∼0, **CacheState::cstate_t** cstate= **CacheState::INVALID**)
- ∼**PrL2CacheBlockInfo** ()
- **MemComponent::component_t getCachedLoc** ()
- void **setCachedLoc** (**MemComponent::component_t** cached_loc)
- void **clearCachedLoc** (**MemComponent::component_t** cached_loc)
- **UInt32 getCachedLocBitVec** ()
- void **invalidate** ()
- void **clone** (**CacheBlockInfo** ∗cache_block_info)

Private Attributes

- **UInt32 m_cached_loc_bitvec**

Additional Inherited Members

6.246.1 Detailed Description

Definition at line 8 of file pr_l2_cache_block_info.h.

6.246.2 Constructor & Destructor Documentation

6.246.2.1 PrL2CacheBlockInfo()

```
PrL2CacheBlockInfo::PrL2CacheBlockInfo (
            IntPtr tag = ∼0,
            CacheState::cstate_t cstate =  CacheState::INVALID )  [inline]
```

Definition at line 14 of file pr_l2_cache_block_info.h.

6.246.2.2 ~PrL2CacheBlockInfo()

```
PrL2CacheBlockInfo::~PrL2CacheBlockInfo ( )  [inline]
```

Definition at line 20 of file pr_l2_cache_block_info.h.

6.246.3 Member Function Documentation

6.246.3.1 clearCachedLoc()

```
void PrL2CacheBlockInfo::clearCachedLoc (
              MemComponent::component_t cached_loc )
```

Definition at line 39 of file pr_l2_cache_block_info.cc.

References MemComponent::INVALID_MEM_COMPONENT, LOG_ASSERT_ERROR, and m_cached_loc_bitvec.

6.246.3.2 clone()

```
void PrL2CacheBlockInfo::clone (
              CacheBlockInfo * cache_block_info )  [virtual]
```

Reimplemented from **CacheBlockInfo** (p. 152).

Definition at line 58 of file pr_l2_cache_block_info.cc.

References CacheBlockInfo::clone(), and m_cached_loc_bitvec.

6.246.3.3 getCachedLoc()

```
MemComponent::component_t PrL2CacheBlockInfo::getCachedLoc ( )
```

Definition at line 5 of file pr_l2_cache_block_info.cc.

References MemComponent::INVALID_MEM_COMPONENT, MemComponent::L1_DCACHE, MemComponent↩
::L1_ICACHE, LOG_ASSERT_ERROR, LOG_PRINT_ERROR, and m_cached_loc_bitvec.

6.246.3.4 getCachedLocBitVec()

```
UInt32 PrL2CacheBlockInfo::getCachedLocBitVec ( )  [inline]
```

Definition at line 26 of file pr_l2_cache_block_info.h.

References m_cached_loc_bitvec.

6.246.3.5 invalidate()

```
void PrL2CacheBlockInfo::invalidate ( )  [virtual]
```

Reimplemented from **CacheBlockInfo** (p. 154).

Definition at line 51 of file pr_l2_cache_block_info.cc.

References CacheBlockInfo::invalidate(), and m_cached_loc_bitvec.

6.246.3.6 setCachedLoc()

```
void PrL2CacheBlockInfo::setCachedLoc (
            MemComponent::component_t cached_loc )
```

Definition at line 27 of file pr_l2_cache_block_info.cc.

References MemComponent::INVALID_MEM_COMPONENT, LOG_ASSERT_ERROR, and m_cached_loc_bitvec.

6.246.4 Member Data Documentation

6.246.4.1 m_cached_loc_bitvec

```
UInt32 PrL2CacheBlockInfo::m_cached_loc_bitvec  [private]
```

Definition at line 11 of file pr_l2_cache_block_info.h.

Referenced by clearCachedLoc(), clone(), getCachedLoc(), getCachedLocBitVec(), invalidate(), and setCached↩Loc().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **pr_l2_cache_block_info.h**
- common/core/memory_subsystem/cache/ **pr_l2_cache_block_info.cc**

6.247 MemoryDependencies::Producer Struct Reference

Public Attributes

- uint64_t **seqnr**
- uint64_t **address**

6.247.1 Detailed Description

Definition at line 11 of file memory_dependencies.h.

6.247.2 Member Data Documentation

6.247.2.1 address

```
uint64_t MemoryDependencies::Producer::address
```

Definition at line 14 of file memory_dependencies.h.

6.247.2.2 seqnr

```
uint64_t MemoryDependencies::Producer::seqnr
```

Definition at line 13 of file memory_dependencies.h.

The documentation for this struct was generated from the following file:

- common/performance_model/performance_models/micro_op/ **memory_dependencies.h**

6.248 Progress Class Reference

```
#include <progress.h>
```

Public Member Functions

- **Progress** ()
- ~**Progress** ()
- void **setProgress** (float progress)

Private Member Functions

- void **record** (**UInt64** time)

Static Private Member Functions

- static **SInt64** **__record** (**UInt64** arg, **UInt64** val)

Private Attributes

- bool **m_enabled**
- FILE * **m_fp**
- time_t **m_t_last**
- bool **m_manual**
- float **m_manual_value**

Static Private Attributes

- static const time_t **m_interval** = 2

6.248.1 Detailed Description

Definition at line 6 of file progress.h.

6.248.2 Constructor & Destructor Documentation

6.248.2.1 Progress()

```
Progress::Progress ( )
```

Definition at line 9 of file progress.cc.

References __record(), HookType::HOOK_PERIODIC_INS, m_enabled, and m_fp.

6.248.2.2 ~Progress()

```
Progress::~Progress ( )
```

Definition at line 26 of file progress.cc.

References m_enabled, and m_fp.

6.248.3 Member Function Documentation

6.248.3.1 __record()

```
static  SInt64 Progress::__record (
              UInt64 arg,
              UInt64 val )  [inline], [static], [private]
```

Definition at line 15 of file progress.h.

Referenced by Progress().

6.248.3.2 record()

```
void Progress::record (
              UInt64 time )  [private]
```

Definition at line 40 of file progress.cc.

References MagicServer::getGlobalInstructionCount(), m_fp, m_interval, m_manual, m_manual_value, and m_t↩
_last.

6.248.3.3 setProgress()

```
void Progress::setProgress (
             float progress )
```

Definition at line 34 of file progress.cc.

References m_manual, and m_manual_value.

Referenced by MagicServer::setProgress().

6.248.4 Member Data Documentation

6.248.4.1 m_enabled

```
bool Progress::m_enabled  [private]
```

Definition at line 19 of file progress.h.

Referenced by Progress(), and ∼Progress().

6.248.4.2 m_fp

```
FILE* Progress::m_fp  [private]
```

Definition at line 20 of file progress.h.

Referenced by Progress(), record(), and ~Progress().

6.248.4.3 m_interval

```
const time_t Progress::m_interval = 2  [static], [private]
```

Definition at line 22 of file progress.h.

Referenced by record().

6.248.4.4 m_manual

```
bool Progress::m_manual  [private]
```

Definition at line 24 of file progress.h.

Referenced by record(), and setProgress().

6.248.4.5 m_manual_value

```
float Progress::m_manual_value  [private]
```

Definition at line 25 of file progress.h.

Referenced by record(), and setProgress().

6.248.4.6 m_t_last

```
time_t Progress::m_t_last  [private]
```

Definition at line 21 of file progress.h.

Referenced by record().

The documentation for this class was generated from the following files:

- common/misc/ **progress.h**
- common/misc/ **progress.cc**

6.249 PseudoInstruction Class Reference

`#include <instruction.h>`

Inheritance diagram for PseudoInstruction:

Public Member Functions

- **PseudoInstruction** (**SubsecondTime** cost, **InstructionType** type= **INST_PSEUDO_MISC**)
- ∼**PseudoInstruction** ()
- **SubsecondTime getCost** (**Core** ∗core) const

Private Attributes

- **SubsecondTime m_cost**

Additional Inherited Members

6.249.1 Detailed Description

Definition at line 123 of file instruction.h.

6.249.2 Constructor & Destructor Documentation

6.249.2.1 PseudoInstruction()

```
PseudoInstruction::PseudoInstruction (
              SubsecondTime cost,
              InstructionType type =  INST_PSEUDO_MISC )
```

Definition at line 62 of file instruction.cc.

6.249.2.2 ∼PseudoInstruction()

```
PseudoInstruction::~PseudoInstruction ( )
```

Definition at line 68 of file instruction.cc.

6.249.3 Member Function Documentation

6.249.3.1 getCost()

```
SubsecondTime PseudoInstruction::getCost (
            Core * core ) const  [virtual]
```

Reimplemented from **Instruction** (p. 623).

Reimplemented in **SpawnInstruction** (p. 1265), and **SyncInstruction** (p. 1318).

Definition at line 72 of file instruction.cc.

References m_cost.

Referenced by PerformanceModel::handleIdleInstruction(), and FastforwardPerformanceModel::queuePseudo↩ Instruction().

6.249.4 Member Data Documentation

6.249.4.1 m_cost

```
SubsecondTime PseudoInstruction::m_cost  [private]
```

Definition at line 132 of file instruction.h.

Referenced by getCost().

The documentation for this class was generated from the following files:

- common/performance_model/ **instruction.h**
- common/performance_model/ **instruction.cc**

6.250 PthreadEmu::pthread_counters_t Struct Reference

Public Attributes

- **UInt64 pthread_count** [7]
- **UInt64 __unused1**
- **SubsecondTime pthread_total_delay_sync** [7]
- **SubsecondTime pthread_total_delay_mem** [7]
- **UInt64 pthread_mutex_lock_contended**
- **UInt64 pthread_mutex_unlock_contended**

6.250.1 Detailed Description

Definition at line 27 of file pthread_emu.cc.

6.250.2 Member Data Documentation

6.250.2.1 __unused1

UInt64 PthreadEmu::pthread_counters_t::__unused1

Definition at line 30 of file pthread_emu.cc.

6.250.2.2 pthread_count

UInt64 PthreadEmu::pthread_counters_t::pthread_count[7]

Definition at line 29 of file pthread_emu.cc.

Referenced by PthreadEmu::pthreadCount().

6.250.2.3 pthread_mutex_lock_contended

UInt64 PthreadEmu::pthread_counters_t::pthread_mutex_lock_contended

Definition at line 33 of file pthread_emu.cc.

Referenced by PthreadEmu::MutexLock().

6.250.2.4 pthread_mutex_unlock_contended

UInt64 PthreadEmu::pthread_counters_t::pthread_mutex_unlock_contended

Definition at line 34 of file pthread_emu.cc.

Referenced by PthreadEmu::MutexUnlock().

6.250.2.5 pthread_total_delay_mem

SubsecondTime PthreadEmu::pthread_counters_t::pthread_total_delay_mem[7]

Definition at line 32 of file pthread_emu.cc.

Referenced by PthreadEmu::pthreadCount().

6.250.2.6 pthread_total_delay_sync

SubsecondTime PthreadEmu::pthread_counters_t::pthread_total_delay_sync[7]

Definition at line 31 of file pthread_emu.cc.

Referenced by PthreadEmu::pthreadCount().

The documentation for this struct was generated from the following file:

- common/system/ **pthread_emu.cc**

6.251 lite::pthread_functions_t Struct Reference

Public Attributes

- String **name**
- **PthreadEmu::pthread_enum_t function**
- **PthreadEmu::state_t state_after**

6.251.1 Detailed Description

Definition at line 31 of file routine_replace.cc.

6.251.2 Member Data Documentation

6.251.2.1 function

PthreadEmu::pthread_enum_t lite::pthread_functions_t::function

Definition at line 33 of file routine_replace.cc.

6.251.2.2 name

```
String lite::pthread_functions_t::name
```

Definition at line 32 of file routine_replace.cc.

Referenced by lite::routineCallback().

6.251.2.3 state_after

```
PthreadEmu::state_t lite::pthread_functions_t::state_after
```

Definition at line 34 of file routine_replace.cc.

Referenced by lite::pthreadAfter().

The documentation for this struct was generated from the following file:

- pin/lite/ **routine_replace.cc**

6.252 PthreadLock Class Reference

```
#include <pthread_lock.h>
```

Inheritance diagram for PthreadLock:

Public Member Functions

- **PthreadLock** ()
- ~**PthreadLock** ()
- void **acquire** ()
- void **release** ()

Private Attributes

- pthread_mutex_t **_mutx**

6.252.1 Detailed Description

Definition at line 8 of file pthread_lock.h.

6.252.2 Constructor & Destructor Documentation

6.252.2.1 PthreadLock()

```
PthreadLock::PthreadLock ( )
```

Definition at line 3 of file pthread_lock.cc.

References _mutx.

6.252.2.2 ~PthreadLock()

```
PthreadLock::~PthreadLock ( )
```

Definition at line 8 of file pthread_lock.cc.

References _mutx.

6.252.3 Member Function Documentation

6.252.3.1 acquire()

```
void PthreadLock::acquire ( )  [virtual]
```

Implements **LockImplementation** (p. 684).

Definition at line 13 of file pthread_lock.cc.

References _mutx.

6.252.3.2 release()

```
void PthreadLock::release ( )  [virtual]
```

Implements **LockImplementation** (p. 685).

Definition at line 18 of file pthread_lock.cc.

References _mutx.

6.252.4 Member Data Documentation

6.252.4.1 _mutx

```
pthread_mutex_t PthreadLock::_mutx  [private]
```

Definition at line 18 of file pthread_lock.h.

Referenced by acquire(), PthreadLock(), release(), and ∼PthreadLock().

The documentation for this class was generated from the following files:

- common/misc/ **pthread_lock.h**
- common/misc/ **pthread_lock.cc**

6.253 PthreadThread Class Reference

```
#include <pthread_thread.h>
```

Inheritance diagram for PthreadThread:

Classes

- struct **FuncData**

Public Member Functions

- **PthreadThread** (**ThreadFunc** func, void ∗param)
- ∼**PthreadThread** ()
- void **run** ()

Static Private Member Functions

- static void ∗ **spawnedThreadFunc** (void ∗)

Private Attributes

- **FuncData m_data**
- pthread_t **m_thread**

Additional Inherited Members

6.253.1 Detailed Description

Definition at line 7 of file pthread_thread.h.

6.253.2 Constructor & Destructor Documentation

6.253.2.1 PthreadThread()

```
PthreadThread::PthreadThread (
             ThreadFunc func,
            void * param )
```

Definition at line 4 of file pthread_thread.cc.

6.253.2.2 ~PthreadThread()

```
PthreadThread::~PthreadThread ( )
```

Definition at line 9 of file pthread_thread.cc.

6.253.3 Member Function Documentation

6.253.3.1 run()

```
void PthreadThread::run ( )  [virtual]
```

Implements **_Thread** (p. 70).

Definition at line 23 of file pthread_thread.cc.

References PthreadThread::FuncData::arg, PthreadThread::FuncData::func, LOG_PRINT, m_data, m_thread, and spawnedThreadFunc().

6.253.3.2 spawnedThreadFunc()

```
void * PthreadThread::spawnedThreadFunc (
            void * vp )  [static], [private]
```

Definition at line 16 of file pthread_thread.cc.

References PthreadThread::FuncData::arg, and PthreadThread::FuncData::func.

Referenced by run().

6.253.4 Member Data Documentation

6.253.4.1 m_data

```
FuncData PthreadThread::m_data  [private]
```

Definition at line 28 of file pthread_thread.h.

Referenced by run().

6.253.4.2 m_thread

```
pthread_t PthreadThread::m_thread  [private]
```

Definition at line 29 of file pthread_thread.h.

Referenced by run().

The documentation for this class was generated from the following files:

- common/misc/ **pthread_thread.h**
- common/misc/ **pthread_thread.cc**

6.254 PthreadTLS Class Reference

Inheritance diagram for PthreadTLS:

Public Member Functions

- **PthreadTLS** ()
- ∼**PthreadTLS** ()
- void ∗ **get** (int thread_id=-1)
- const void ∗ **get** (int thread_id=-1) const
- void **set** (void ∗vp)

Private Attributes

- pthread_key_t **m_key**

Additional Inherited Members

6.254.1 Detailed Description

Definition at line 6 of file pthread_tls.cc.

6.254.2 Constructor & Destructor Documentation

6.254.2.1 PthreadTLS()

```
PthreadTLS::PthreadTLS ( )  [inline]
```

Definition at line 9 of file pthread_tls.cc.

References m_key.

6.254.2.2 ∼PthreadTLS()

```
PthreadTLS::∼PthreadTLS ( )  [inline]
```

Definition at line 14 of file pthread_tls.cc.

References m_key.

6.254.3 Member Function Documentation

6.254.3.1 get() [1/2]

```
void* PthreadTLS::get (
            int thread_id = -1 )  [inline], [virtual]
```

Implements **TLS** (p. 1422).

Definition at line 19 of file pthread_tls.cc.

References m_key.

6.254.3.2 get() [2/2]

```
const void* PthreadTLS::get (
            int thread_id = -1 ) const  [inline], [virtual]
```

Implements **TLS** (p. 1421).

Definition at line 24 of file pthread_tls.cc.

References m_key.

6.254.3.3 set()

```
void PthreadTLS::set (
            void * vp )  [inline], [virtual]
```

Implements **TLS** (p. 1423).

Definition at line 29 of file pthread_tls.cc.

References m_key.

6.254.4 Member Data Documentation

6.254.4.1 m_key

```
pthread_key_t PthreadTLS::m_key  [private]
```

Definition at line 35 of file pthread_tls.cc.

Referenced by get(), PthreadTLS(), set(), and ~PthreadTLS().

The documentation for this class was generated from the following file:

- common/misc/ **pthread_tls.cc**

6.255 HooksPy::PyBbv Class Reference

Static Public Member Functions

- static void **setup** (void)

6.255.1 Detailed Description

Definition at line 41 of file hooks_py.h.

6.255.2 Member Function Documentation

6.255.2.1 setup()

```
void HooksPy::PyBbv::setup (
            void  )  [static]
```

Definition at line 63 of file py_bbv.cc.

References BbvCount::NUM_BBV, and PyBbvMethods.

Referenced by HooksPy::setup().

The documentation for this class was generated from the following files:

- common/scripting/ **hooks_py.h**
- common/scripting/ **py_bbv.cc**

6.256 HooksPy::PyConfig Class Reference

Static Public Member Functions

- static void **setup** (void)

6.256.1 Detailed Description

Definition at line 21 of file hooks_py.h.

6.256.2 Member Function Documentation

6.256.2.1 setup()

```
void HooksPy::PyConfig::setup (
            void  )  [static]
```

Definition at line 86 of file py_config.cc.

References PyConfigMethods.

Referenced by HooksPy::setup().

The documentation for this class was generated from the following files:

- common/scripting/ **hooks_py.h**
- common/scripting/ **py_config.cc**

6.257 HooksPy::PyControl Class Reference

Static Public Member Functions

- static void **setup** (void)

6.257.1 Detailed Description

Definition at line 37 of file hooks_py.h.

6.257.2 Member Function Documentation

6.257.2.1 setup()

```
void HooksPy::PyControl::setup (
            void  )  [static]
```

Definition at line 78 of file py_control.cc.

References PyControlMethods.

Referenced by HooksPy::setup().

The documentation for this class was generated from the following files:

- common/scripting/ **hooks_py.h**
- common/scripting/ **py_control.cc**

6.258 HooksPy::PyDvfs Class Reference

Static Public Member Functions

- static void **setup** (void)

6.258.1 Detailed Description

Definition at line 33 of file hooks_py.h.

6.258.2 Member Function Documentation

6.258.2.1 setup()

```
void HooksPy::PyDvfs::setup (
            void  )  [static]
```

Definition at line 75 of file py_dvfs.cc.

References PyDvfsMethods.

Referenced by HooksPy::setup().

The documentation for this class was generated from the following files:

- common/scripting/ **hooks_py.h**
- common/scripting/ **py_dvfs.cc**

6.259 HooksPy::PyHooks Class Reference

Static Public Member Functions

- static void **setup** (void)

6.259.1 Detailed Description

Definition at line 29 of file hooks_py.h.

6.259.2 Member Function Documentation

6.259.2.1 setup()

```
void HooksPy::PyHooks::setup (
            void  )  [static]
```

Definition at line 209 of file py_hooks.cc.

References HookType::hook_type_names, HookType::HOOK_TYPES_MAX, and PyHooksMethods.

Referenced by HooksPy::setup().

The documentation for this class was generated from the following files:

- common/scripting/ **hooks_py.h**
- common/scripting/ **py_hooks.cc**

6.260 HooksPy::PyMem Class Reference

Static Public Member Functions

- static void **setup** (void)

6.260.1 Detailed Description

Definition at line 45 of file hooks_py.h.

6.260.2 Member Function Documentation

6.260.2.1 setup()

```
void HooksPy::PyMem::setup (
            void  )  [static]
```

Definition at line 62 of file py_mem.cc.

References PyMemMethods.

Referenced by HooksPy::setup().

The documentation for this class was generated from the following files:

- common/scripting/ **hooks_py.h**
- common/scripting/ **py_mem.cc**

6.261 HooksPy::PyStats Class Reference

Static Public Member Functions

- static void **setup** (void)

6.261.1 Detailed Description

Definition at line 25 of file hooks_py.h.

6.261.2 Member Function Documentation

6.261.2.1 setup()

```
void HooksPy::PyStats::setup (
             void  )  [static]
```

Definition at line 258 of file py_stats.cc.

References PyStatsMethods, and statsGetterType.

Referenced by HooksPy::setup().

The documentation for this class was generated from the following files:

- common/scripting/ **hooks_py.h**
- common/scripting/ **py_stats.cc**

6.262 HooksPy::PyThread Class Reference

Static Public Member Functions

- static void **setup** (void)

6.262.1 Detailed Description

Definition at line 49 of file hooks_py.h.

6.262.2 Member Function Documentation

6.262.2.1 setup()

```
void HooksPy::PyThread::setup (
            void  )  [static]
```

Definition at line 148 of file py_thread.cc.

References PyThreadMethods.

Referenced by HooksPy::setup().

The documentation for this class was generated from the following files:

- common/scripting/ **hooks_py.h**
- common/scripting/ **py_thread.cc**

6.263 QueueModel Class Reference

```
#include <queue_model.h>
```

Inheritance diagram for QueueModel:

Public Member Functions

- **QueueModel** ()
- virtual ~**QueueModel** ()
- virtual **SubsecondTime** **computeQueueDelay** (**SubsecondTime** pkt_time, **SubsecondTime** processing_time, **core_id_t** requester= **INVALID_CORE_ID**)=0

Static Public Member Functions

- static **QueueModel** * **create** (String name, **UInt32** id, String model_type, **SubsecondTime** min_↩ processing_time)

6.263.1 Detailed Description

Definition at line 9 of file queue_model.h.

6.263.2 Constructor & Destructor Documentation

6.263.2.1 QueueModel()

```
QueueModel::QueueModel ( )  [inline]
```

Definition at line 12 of file queue_model.h.

6.263.2.2 ~QueueModel()

```
virtual QueueModel::~QueueModel ( )  [inline], [virtual]
```

Definition at line 13 of file queue_model.h.

6.263.3 Member Function Documentation

6.263.3.1 computeQueueDelay()

```
virtual  SubsecondTime QueueModel::computeQueueDelay (
              SubsecondTime pkt_time,
              SubsecondTime processing_time,
              core_id_t requester =  INVALID_CORE_ID )  [pure virtual]
```

Implemented in **QueueModelHistoryList** (p. 970), **QueueModelWindowedMG1** (p. 976), **QueueModelBasic** (p. 966), and **QueueModelContention** (p. 968).

Referenced by NucaCache::accessDataArray(), DramCache::accessDataArray(), NetworkModelEMeshHopBy↩Hop::computeEjectionPortQueueDelay(), NetworkModelEMeshHopByHop::computeInjectionPortQueueDelay(), NetworkModelEMeshHopByHop::computeLatency(), DramPerfModelConstant::getAccessLatency(), DramPerf↩ModelNormal::getAccessLatency(), DramPerfModelReadWrite::getAccessLatency(), and NetworkModelBus↩Global::useBus().

6.263.3.2 create()

```
QueueModel * QueueModel::create (
             String name,
              UInt32 id,
             String model_type,
              SubsecondTime min_processing_time )  [static]
```

Definition at line 12 of file queue_model.cc.

References LOG_PRINT_ERROR.

Referenced by NetworkModelEMeshHopByHop::createQueueModels(), DramCache::DramCache(), DramPerf↩ModelConstant::DramPerfModelConstant(), DramPerfModelNormal::DramPerfModelNormal(), DramPerfModel↩ReadWrite::DramPerfModelReadWrite(), NetworkModelBusGlobal::NetworkModelBusGlobal(), and NucaCache↩::NucaCache().

The documentation for this class was generated from the following files:

- common/performance_model/ **queue_model.h**
- common/performance_model/ **queue_model.cc**

6.264 QueueModelBasic Class Reference

`#include <queue_model_basic.h>`

Inheritance diagram for QueueModelBasic:

Public Member Functions

- **QueueModelBasic** (String name, **UInt32** id, bool moving_avg_enabled, **UInt32** moving_avg_window_size, String moving_avg_type_str)
- ~**QueueModelBasic** ()
- **SubsecondTime computeQueueDelay** (**SubsecondTime** pkt_time, **SubsecondTime** processing_time, **core_id_t** requester= **INVALID_CORE_ID**)

Private Attributes

- **SubsecondTime m_queue_time**
- **MovingAverage**< **SubsecondTime** > * **m_moving_average**

Additional Inherited Members

6.264.1 Detailed Description

Definition at line 9 of file queue_model_basic.h.

6.264.2 Constructor & Destructor Documentation

6.264.2.1 QueueModelBasic()

```
QueueModelBasic::QueueModelBasic (
            String name,
             UInt32 id,
            bool moving_avg_enabled,
             UInt32 moving_avg_window_size,
            String moving_avg_type_str )
```

Definition at line 5 of file queue_model_basic.cc.

References MovingAverage< T >::createAvgType(), m_moving_average, and MovingAverage< T >::parseAvg↩Type().

6.264.2.2 ~QueueModelBasic()

```
QueueModelBasic::~QueueModelBasic ( )
```

Definition at line 19 of file queue_model_basic.cc.

6.264.3 Member Function Documentation

6.264.3.1 computeQueueDelay()

```
SubsecondTime QueueModelBasic::computeQueueDelay (
            SubsecondTime pkt_time,
            SubsecondTime processing_time,
            core_id_t requester = INVALID_CORE_ID ) [virtual]
```

Implements **QueueModel** (p. 964).

Definition at line 23 of file queue_model_basic.cc.

References MovingAverage< T >::compute(), itostr(), LOG_PRINT, m_moving_average, m_queue_time, SubsecondTime::NS(), and SubsecondTime::Zero().

6.264.4 Member Data Documentation

6.264.4.1 m_moving_average

```
MovingAverage< SubsecondTime >* QueueModelBasic::m_moving_average [private]
```

Definition at line 19 of file queue_model_basic.h.

Referenced by computeQueueDelay(), and QueueModelBasic().

6.264.4.2 m_queue_time

```
SubsecondTime QueueModelBasic::m_queue_time [private]
```

Definition at line 18 of file queue_model_basic.h.

Referenced by computeQueueDelay().

The documentation for this class was generated from the following files:

- common/performance_model/ **queue_model_basic.h**
- common/performance_model/ **queue_model_basic.cc**

6.265 QueueModelContention Class Reference

`#include <queue_model_contention.h>`

Inheritance diagram for QueueModelContention:

Public Member Functions

- **QueueModelContention** (String name, **UInt32** id, **UInt32** num_outstanding=1)
- ~**QueueModelContention** ()
- **SubsecondTime computeQueueDelay** (**SubsecondTime** pkt_time, **SubsecondTime** processing_time, **core_id_t** requester= **INVALID_CORE_ID**)

Private Attributes

- **ContentionModel m_contention**

Additional Inherited Members

6.265.1 Detailed Description

Definition at line 8 of file queue_model_contention.h.

6.265.2 Constructor & Destructor Documentation

6.265.2.1 QueueModelContention()

```
QueueModelContention::QueueModelContention (
             String name,
              UInt32 id,
              UInt32 num_outstanding = 1 )
```

Definition at line 3 of file queue_model_contention.cc.

6.265.2.2 ~QueueModelContention()

```
QueueModelContention::~QueueModelContention ( )
```

Definition at line 7 of file queue_model_contention.cc.

6.265.3 Member Function Documentation

6.265.3.1 computeQueueDelay()

```
SubsecondTime QueueModelContention::computeQueueDelay (
            SubsecondTime pkt_time,
            SubsecondTime processing_time,
            core_id_t requester = INVALID_CORE_ID ) [virtual]
```

Implements **QueueModel** (p. 964).

Definition at line 10 of file queue_model_contention.cc.

References ContentionModel::getCompletionTime(), m_contention, and t_start.

6.265.4 Member Data Documentation

6.265.4.1 m_contention

```
ContentionModel QueueModelContention::m_contention [private]
```

Definition at line 17 of file queue_model_contention.h.

Referenced by computeQueueDelay().

The documentation for this class was generated from the following files:

- common/performance_model/ **queue_model_contention.h**
- common/performance_model/ **queue_model_contention.cc**

6.266 QueueModelHistoryList Class Reference

```
#include <queue_model_history_list.h>
```

Inheritance diagram for QueueModelHistoryList:

Public Types

- typedef std::list< std::pair< **SubsecondTime**, **SubsecondTime** > > **FreeIntervalList**

Public Member Functions

- **QueueModelHistoryList** (String name, **UInt32** id, **SubsecondTime** min_processing_time)
- ~**QueueModelHistoryList** ()
- **SubsecondTime computeQueueDelay** (**SubsecondTime** pkt_time, **SubsecondTime** processing_time, **core_id_t** requester= **INVALID_CORE_ID**)
- float **getQueueUtilization** ()
- float **getFracRequestsUsingAnalyticalModel** ()

Private Member Functions

- void **updateQueueUtilization** (**SubsecondTime** processing_time)
- void **updateAverageDelay** (**SubsecondTime** queue_delay)
- **SubsecondTime computeUsingHistoryList** (**SubsecondTime** pkt_time, **SubsecondTime** processing↩ _time)
- **SubsecondTime computeUsingAnalyticalModel** (**SubsecondTime** pkt_time, **SubsecondTime** processing_time)

Private Attributes

- **SubsecondTime m_min_processing_time**
- **UInt32 m_max_free_interval_list_size**
- **FreeIntervalList m_free_interval_list**
- **SubsecondTime m_utilized_time**
- **SubsecondTime m_total_queue_delay**
- **MovingAverage**< **SubsecondTime** > * **m_average_delay**
- bool **m_analytical_model_enabled**
- **UInt64 m_total_requests**
- **UInt64 m_total_requests_using_analytical_model**

Additional Inherited Members

6.266.1 Detailed Description

Definition at line 10 of file queue_model_history_list.h.

6.266.2 Member Typedef Documentation

6.266.2.1 FreeIntervalList

```
typedef std::list<std::pair< SubsecondTime, SubsecondTime> >  QueueModelHistoryList::Free↩
IntervalList
```

Definition at line 13 of file queue_model_history_list.h.

6.266.3 Constructor & Destructor Documentation

6.266.3.1 QueueModelHistoryList()

```
QueueModelHistoryList::QueueModelHistoryList (
            String name,
             UInt32 id,
             SubsecondTime min_processing_time )
```

Definition at line 10 of file queue_model_history_list.cc.

References MovingAverage< T >::createAvgType(), SubsecondTime::FS(), LOG_PRINT_ERROR, m_analytical↩_model_enabled, m_average_delay, m_free_interval_list, m_max_free_interval_list_size, m_total_queue_↩delay, m_total_requests, m_total_requests_using_analytical_model, m_utilized_time, registerStatsMetric(), and SubsecondTime::Zero().

6.266.3.2 ~QueueModelHistoryList()

```
QueueModelHistoryList::~QueueModelHistoryList ( )
```

Definition at line 50 of file queue_model_history_list.cc.

References m_average_delay.

6.266.4 Member Function Documentation

6.266.4.1 computeQueueDelay()

```
SubsecondTime QueueModelHistoryList::computeQueueDelay (
             SubsecondTime pkt_time,
             SubsecondTime processing_time,
             core_id_t requester =  INVALID_CORE_ID )  [virtual]
```

Implements **QueueModel** (p. 964).

Definition at line 56 of file queue_model_history_list.cc.

References computeUsingAnalyticalModel(), computeUsingHistoryList(), LOG_ASSERT_ERROR, m_analytical↩_model_enabled, m_free_interval_list, m_total_queue_delay, m_total_requests, m_total_requests_using_↩analytical_model, updateAverageDelay(), and updateQueueUtilization().

6.266.4.2 computeUsingAnalyticalModel()

```
SubsecondTime QueueModelHistoryList::computeUsingAnalyticalModel (
              SubsecondTime pkt_time,
              SubsecondTime processing_time )  [private]
```

Definition at line 129 of file queue_model_history_list.cc.

References MovingAverage< T >::compute(), and m_average_delay.

Referenced by computeQueueDelay().

6.266.4.3 computeUsingHistoryList()

```
SubsecondTime QueueModelHistoryList::computeUsingHistoryList (
              SubsecondTime pkt_time,
              SubsecondTime processing_time )  [private]
```

Definition at line 141 of file queue_model_history_list.cc.

References itostr(), LOG_ASSERT_ERROR, LOG_PRINT, m_free_interval_list, m_max_free_interval_list_size, m_min_processing_time, SubsecondTime::MaxTime(), and SubsecondTime::Zero().

Referenced by computeQueueDelay().

6.266.4.4 getFracRequestsUsingAnalyticalModel()

```
float QueueModelHistoryList::getFracRequestsUsingAnalyticalModel ( )
```

Definition at line 107 of file queue_model_history_list.cc.

References m_total_requests, and m_total_requests_using_analytical_model.

6.266.4.5 getQueueUtilization()

```
float QueueModelHistoryList::getQueueUtilization ( )
```

Definition at line 89 of file queue_model_history_list.cc.

References SubsecondTime::getInternalDataForced(), itostr(), LOG_ASSERT_ERROR, m_free_interval_list, m_↩ utilized_time, and SubsecondTime::Zero().

6.266.4.6 updateAverageDelay()

```
void QueueModelHistoryList::updateAverageDelay (
            SubsecondTime queue_delay )  [private]
```

Definition at line 123 of file queue_model_history_list.cc.

References m_average_delay, and MovingAverage< T >::update().

Referenced by computeQueueDelay().

6.266.4.7 updateQueueUtilization()

```
void QueueModelHistoryList::updateQueueUtilization (
            SubsecondTime processing_time )  [private]
```

Definition at line 116 of file queue_model_history_list.cc.

References m_utilized_time.

Referenced by computeQueueDelay().

6.266.5 Member Data Documentation

6.266.5.1 m_analytical_model_enabled

```
bool QueueModelHistoryList::m_analytical_model_enabled  [private]
```

Definition at line 35 of file queue_model_history_list.h.

Referenced by computeQueueDelay(), and QueueModelHistoryList().

6.266.5.2 m_average_delay

```
MovingAverage< SubsecondTime >* QueueModelHistoryList::m_average_delay  [private]
```

Definition at line 32 of file queue_model_history_list.h.

Referenced by computeUsingAnalyticalModel(), QueueModelHistoryList(), updateAverageDelay(), and ~Queue↩ ModelHistoryList().

6.266.5.3 m_free_interval_list

FreeIntervalList QueueModelHistoryList::m_free_interval_list [private]

Definition at line 27 of file queue_model_history_list.h.

Referenced by computeQueueDelay(), computeUsingHistoryList(), getQueueUtilization(), and QueueModel↩ HistoryList().

6.266.5.4 m_max_free_interval_list_size

UInt32 QueueModelHistoryList::m_max_free_interval_list_size [private]

Definition at line 25 of file queue_model_history_list.h.

Referenced by computeUsingHistoryList(), and QueueModelHistoryList().

6.266.5.5 m_min_processing_time

SubsecondTime QueueModelHistoryList::m_min_processing_time [private]

Definition at line 24 of file queue_model_history_list.h.

Referenced by computeUsingHistoryList().

6.266.5.6 m_total_queue_delay

SubsecondTime QueueModelHistoryList::m_total_queue_delay [private]

Definition at line 31 of file queue_model_history_list.h.

Referenced by computeQueueDelay(), and QueueModelHistoryList().

6.266.5.7 m_total_requests

UInt64 QueueModelHistoryList::m_total_requests [private]

Definition at line 38 of file queue_model_history_list.h.

Referenced by computeQueueDelay(), getFracRequestsUsingAnalyticalModel(), and QueueModelHistoryList().

6.266.5.8 m_total_requests_using_analytical_model

UInt64 QueueModelHistoryList::m_total_requests_using_analytical_model [private]

Definition at line 39 of file queue_model_history_list.h.

Referenced by computeQueueDelay(), getFracRequestsUsingAnalyticalModel(), and QueueModelHistoryList().

6.266.5.9 m_utilized_time

SubsecondTime QueueModelHistoryList::m_utilized_time [private]

Definition at line 30 of file queue_model_history_list.h.

Referenced by getQueueUtilization(), QueueModelHistoryList(), and updateQueueUtilization().

The documentation for this class was generated from the following files:

- common/performance_model/ **queue_model_history_list.h**
- common/performance_model/ **queue_model_history_list.cc**

6.267 QueueModelWindowedMG1 Class Reference

#include <queue_model_windowed_mg1.h>

Inheritance diagram for QueueModelWindowedMG1:

Public Member Functions

- **QueueModelWindowedMG1** (String name, **UInt32** id)
- ~**QueueModelWindowedMG1** ()
- **SubsecondTime computeQueueDelay** (**SubsecondTime** pkt_time, **SubsecondTime** processing_time, **core_id_t** requester= **INVALID_CORE_ID**)

Private Member Functions

- void **addItem** (**SubsecondTime** pkt_time, **SubsecondTime** service_time)
- void **removeItems** (**SubsecondTime** earliest_time)

Private Attributes

- const **SubsecondTime m_window_size**
- **UInt64 m_total_requests**
- **SubsecondTime m_total_utilized_time**
- **SubsecondTime m_total_queue_delay**
- std::multimap< **SubsecondTime**, **SubsecondTime** > **m_window**
- **UInt64 m_num_arrivals**
- **UInt64 m_service_time_sum**
- **UInt64 m_service_time_sum2**

Additional Inherited Members

6.267.1 Detailed Description

Definition at line 10 of file queue_model_windowed_mg1.h.

6.267.2 Constructor & Destructor Documentation

6.267.2.1 QueueModelWindowedMG1()

```
QueueModelWindowedMG1::QueueModelWindowedMG1 (
            String name,
             UInt32 id )
```

Definition at line 7 of file queue_model_windowed_mg1.cc.

References m_total_queue_delay, m_total_requests, m_total_utilized_time, and registerStatsMetric().

6.267.2.2 ~QueueModelWindowedMG1()

```
QueueModelWindowedMG1::~QueueModelWindowedMG1 ( )
```

Definition at line 21 of file queue_model_windowed_mg1.cc.

6.267.3 Member Function Documentation

6.267.3.1 addItem()

```
void QueueModelWindowedMG1::addItem (
              SubsecondTime pkt_time,
              SubsecondTime service_time )  [private]
```

Definition at line 61 of file queue_model_windowed_mg1.cc.

References SubsecondTime::getPS(), m_num_arrivals, m_service_time_sum, m_service_time_sum2, and m_↩ window.

Referenced by computeQueueDelay().

6.267.3.2 computeQueueDelay()

```
SubsecondTime QueueModelWindowedMG1::computeQueueDelay (
              SubsecondTime pkt_time,
              SubsecondTime processing_time,
              core_id_t requester =  INVALID_CORE_ID )  [virtual]
```

Implements **QueueModel** (p. 964).

Definition at line 25 of file queue_model_windowed_mg1.cc.

References addItem(), SubsecondTime::getPS(), m_num_arrivals, m_service_time_sum, m_service_time_sum2, m_total_queue_delay, m_total_requests, m_total_utilized_time, m_window_size, SubsecondTime::PS(), remove↩ Items(), and SubsecondTime::Zero().

6.267.3.3 removeItems()

```
void QueueModelWindowedMG1::removeItems (
              SubsecondTime earliest_time )  [private]
```

Definition at line 70 of file queue_model_windowed_mg1.cc.

References m_num_arrivals, m_service_time_sum, m_service_time_sum2, and m_window.

Referenced by computeQueueDelay().

6.267.4 Member Data Documentation

6.267.4.1 m_num_arrivals

UInt64 QueueModelWindowedMG1::m_num_arrivals [private]

Definition at line 26 of file queue_model_windowed_mg1.h.

Referenced by addItem(), computeQueueDelay(), and removeItems().

6.267.4.2 m_service_time_sum

UInt64 QueueModelWindowedMG1::m_service_time_sum [private]

Definition at line 27 of file queue_model_windowed_mg1.h.

Referenced by addItem(), computeQueueDelay(), and removeItems().

6.267.4.3 m_service_time_sum2

UInt64 QueueModelWindowedMG1::m_service_time_sum2 [private]

Definition at line 28 of file queue_model_windowed_mg1.h.

Referenced by addItem(), computeQueueDelay(), and removeItems().

6.267.4.4 m_total_queue_delay

SubsecondTime QueueModelWindowedMG1::m_total_queue_delay [private]

Definition at line 23 of file queue_model_windowed_mg1.h.

Referenced by computeQueueDelay(), and QueueModelWindowedMG1().

6.267.4.5 m_total_requests

UInt64 QueueModelWindowedMG1::m_total_requests [private]

Definition at line 21 of file queue_model_windowed_mg1.h.

Referenced by computeQueueDelay(), and QueueModelWindowedMG1().

6.267.4.6 m_total_utilized_time

SubsecondTime QueueModelWindowedMG1::m_total_utilized_time [private]

Definition at line 22 of file queue_model_windowed_mg1.h.

Referenced by computeQueueDelay(), and QueueModelWindowedMG1().

6.267.4.7 m_window

std::multimap< **SubsecondTime**, **SubsecondTime**> QueueModelWindowedMG1::m_window [private]

Definition at line 25 of file queue_model_windowed_mg1.h.

Referenced by addItem(), and removeItems().

6.267.4.8 m_window_size

const **SubsecondTime** QueueModelWindowedMG1::m_window_size [private]

Definition at line 19 of file queue_model_windowed_mg1.h.

Referenced by computeQueueDelay().

The documentation for this class was generated from the following files:

- common/performance_model/ **queue_model_windowed_mg1.h**
- common/performance_model/ **queue_model_windowed_mg1.cc**

6.268 Random Class Reference

#include <random.h>

Public Types

- typedef **UInt32** **value_t**

Public Member Functions

- **Random** ()
- ~**Random** ()
- void **seed** (**value_t** s)
- **value_t** **next** (**value_t** limit=32768)

Private Attributes

- **value_t _seed**

6.268.1 Detailed Description

Definition at line 9 of file random.h.

6.268.2 Member Typedef Documentation

6.268.2.1 value_t

```
typedef UInt32 Random::value_t
```

Definition at line 12 of file random.h.

6.268.3 Constructor & Destructor Documentation

6.268.3.1 Random()

```
Random::Random ( ) [inline]
```

Definition at line 18 of file random.h.

6.268.3.2 ~Random()

```
Random::~Random ( ) [inline]
```

Definition at line 19 of file random.h.

6.268.4 Member Function Documentation

6.268.4.1 next()

```
value_t Random::next (
            value_t limit = 32768 )  [inline]
```

Definition at line 26 of file random.h.

References _seed.

Referenced by PeriodicSampling::callbackDetailed(), CacheSetRandom::getReplacementIndex(), and PeriodicSampling::PeriodicSampling().

6.268.4.2 seed()

```
void Random::seed (
            value_t s )  [inline]
```

Definition at line 21 of file random.h.

References _seed.

Referenced by CacheSetRandom::CacheSetRandom(), and PeriodicSampling::PeriodicSampling().

6.268.5 Member Data Documentation

6.268.5.1 _seed

```
value_t Random::_seed  [private]
```

Definition at line 15 of file random.h.

Referenced by next(), and seed().

The documentation for this class was generated from the following file:

- common/misc/ **random.h**

6.269 raw_spinlock_t Struct Reference

```
#include <spinlock.h>
```

Public Attributes

- volatile unsigned int **slock**

6.269.1 Detailed Description

Definition at line 4 of file spinlock.h.

6.269.2 Member Data Documentation

6.269.2.1 slock

```
volatile unsigned int raw_spinlock_t::slock
```

Definition at line 5 of file spinlock.h.

Referenced by __raw_spin_trylock().

The documentation for this struct was generated from the following file:

- common/misc/ **spinlock.h**

6.270 RecvInstruction Class Reference

```
#include <instruction.h>
```

Inheritance diagram for RecvInstruction:

Public Member Functions

- **RecvInstruction** (**SubsecondTime** cost)

Additional Inherited Members

6.270.1 Detailed Description

Definition at line 135 of file instruction.h.

6.270.2 Constructor & Destructor Documentation

6.270.2.1 RecvInstruction()

```
RecvInstruction::RecvInstruction (
              SubsecondTime cost )  [inline]
```

Definition at line 138 of file instruction.h.

The documentation for this class was generated from the following file:

- common/performance_model/ **instruction.h**

6.271 RegisterDependencies Class Reference

`#include <register_dependencies.h>`

Public Member Functions

- **RegisterDependencies** ()
- void **setDependencies** (**DynamicMicroOp** &microOp, uint64_t lowestValidSequenceNumber)
- uint64_t **peekProducer** (dl::Decoder::decoder_reg reg, uint64_t lowestValidSequenceNumber)
- void **clear** ()

Private Attributes

- uint64_t **producers** [280]

6.271.1 Detailed Description

Definition at line 13 of file register_dependencies.h.

6.271.2 Constructor & Destructor Documentation

6.271.2.1 RegisterDependencies()

```
RegisterDependencies::RegisterDependencies ( )
```

Definition at line 4 of file register_dependencies.cc.

References clear().

6.271.3 Member Function Documentation

6.271.3.1 clear()

```
void RegisterDependencies::clear ( )
```

Definition at line 52 of file register_dependencies.cc.

References INVALID_SEQNR, and producers.

Referenced by Windows::clear(), and RegisterDependencies().

6.271.3.2 peekProducer()

```
uint64_t RegisterDependencies::peekProducer (
             dl::Decoder::decoder_reg reg,
             uint64_t lowestValidSequenceNumber )
```

Definition at line 40 of file register_dependencies.cc.

References INVALID_SEQNR, and producers.

Referenced by RobSmtTimer::setStoreAddressProducers(), and RobTimer::simulate().

6.271.3.3 setDependencies()

```
void RegisterDependencies::setDependencies (
              DynamicMicroOp & microOp,
             uint64_t lowestValidSequenceNumber )
```

Definition at line 9 of file register_dependencies.cc.

References DynamicMicroOp::addDependency(), MicroOp::getDestinationRegister(), DynamicMicroOp::get↩MicroOp(), DynamicMicroOp::getSequenceNumber(), MicroOp::getSourceRegister(), INVALID_SEQNR, LOG_A↩SSERT_ERROR, and producers.

Referenced by Windows::add(), RobSmtTimer::setDependencies(), and RobTimer::simulate().

6.271.4 Member Data Documentation

6.271.4.1 producers

```
uint64_t RegisterDependencies::producers[280]  [private]
```

Definition at line 18 of file register_dependencies.h.

Referenced by clear(), peekProducer(), and setDependencies().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/micro_op/ **register_dependencies.h**
- common/performance_model/performance_models/micro_op/ **register_dependencies.cc**

6.272 ReqQueueListTemplate< T_Req > Class Template Reference

```
#include <req_queue_list_template.h>
```

Public Member Functions

- **ReqQueueListTemplate** ()
- ~**ReqQueueListTemplate** ()
- void **enqueue** (**IntPtr** address, T_Req *shmem_req)
- T_Req * **dequeue** (**IntPtr** address)
- T_Req * **front** (**IntPtr** address)
- T_Req * **back** (**IntPtr** address)
- **UInt32** **size** (**IntPtr** address)
- bool **empty** (**IntPtr** address)

Private Attributes

- std::map< **IntPtr**, std::queue< T_Req * > * > **m_req_queue_list**

6.272.1 Detailed Description

template<class T_Req>
class ReqQueueListTemplate< T_Req >

Definition at line 15 of file req_queue_list_template.h.

6.272.2 Constructor & Destructor Documentation

6.272.2.1 ReqQueueListTemplate()

```
template<class T_Req >
ReqQueueListTemplate< T_Req >::  ReqQueueListTemplate ( )  [inline]
```

Definition at line 21 of file req_queue_list_template.h.

6.272.2.2 ~ReqQueueListTemplate()

```
template<class T_Req >
ReqQueueListTemplate< T_Req >::~ ReqQueueListTemplate ( )  [inline]
```

Definition at line 22 of file req_queue_list_template.h.

6.272.3 Member Function Documentation

6.272.3.1 back()

```
template<class T_Req >
T_Req *  ReqQueueListTemplate< T_Req >::back (
              IntPtr address )
```

Definition at line 72 of file req_queue_list_template.h.

6.272.3.2 dequeue()

```
template<class T_Req >
T_Req *  ReqQueueListTemplate< T_Req >::dequeue (
              IntPtr address )
```

Definition at line 45 of file req_queue_list_template.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), and PrL1PrL2Dram↩DirectoryMSI::DramDirectoryCntlr::processNextReqFromL2Cache().

6.272.3.3 empty()

```
template<class T_Req >
bool  ReqQueueListTemplate< T_Req >::empty (
              IntPtr address )
```

Definition at line 92 of file req_queue_list_template.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), and PrL1PrL2Dram↩DirectoryMSI::DramDirectoryCntlr::processNextReqFromL2Cache().

6.272.3.4 enqueue()

```
template<class T_Req >
void  ReqQueueListTemplate< T_Req >::enqueue (
              IntPtr address,
             T_Req * shmem_req )
```

Definition at line 34 of file req_queue_list_template.h.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::handleMsgFromL2Cache(), ParametricDram↩DirectoryMSI::CacheCntlr::initiateDirectoryAccess(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr↩::processDirectoryEntryAllocationReq().

6.272.3.5 front()

```
template<class T_Req >
T_Req *  ReqQueueListTemplate< T_Req >::front (
              IntPtr address )
```

Definition at line 62 of file req_queue_list_template.h.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), PrL1PrL2Dram↩DirectoryMSI::DramDirectoryCntlr::processDRAMReply(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr↩::processFlushRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2↩Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processNextReqFromL2Cache(), PrL1PrL2Dram↩DirectoryMSI::DramDirectoryCntlr::processWbRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectory↩Cntlr::processWbReqFromL2Cache(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAnd↩SendToL2Cache().

6.272.3.6 size()

```
template<class T_Req >
UInt32  ReqQueueListTemplate< T_Req >::size (
              IntPtr address )
```

Definition at line 82 of file req_queue_list_template.h.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::handleMsgFromL2Cache(), ParametricDram↩DirectoryMSI::CacheCntlr::initiateDirectoryAccess(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::process↩DirectoryEntryAllocationReq(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDRAMReply(), PrL1↩PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::↩DramDirectoryCntlr::processInvRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::process↩NextReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbRepFromL2Cache(), Pr↩L1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache(), and PrL1PrL2DramDirectoryM↩SI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache().

6.272.4 Member Data Documentation

6.272.4.1 m_req_queue_list

```
template<class T_Req >
std::map< IntPtr, std::queue<T_Req*>* >  ReqQueueListTemplate< T_Req >::m_req_queue_list
[private]
```

Definition at line 18 of file req_queue_list_template.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/cache/ **req_queue_list_template.h**

6.273 riscvinstr Struct Reference

```
#include <riscv_meta.h>
```

Public Attributes

- unsigned int **opcode**
- bool **has_alu**
- bool **has_mul**
- bool **has_div**
- bool **has_fpu**
- bool **has_fdiv**
- bool **has_ifpu**
- bool **is_memory**

6.273.1 Detailed Description

Definition at line 327 of file riscv_meta.h.

6.273.2 Member Data Documentation

6.273.2.1 has_alu

```
bool riscvinstr::has_alu
```

Definition at line 329 of file riscv_meta.h.

6.273.2.2 has_div

```
bool riscvinstr::has_div
```

Definition at line 331 of file riscv_meta.h.

Referenced by DynamicMicroOpBoomV1::getPort().

6.273.2.3 has_fdiv

```
bool riscvinstr::has_fdiv
```

Definition at line 333 of file riscv_meta.h.

Referenced by DynamicMicroOpBoomV1::getPort().

6.273.2.4 has_fpu

```
bool riscvinstr::has_fpu
```

Definition at line 332 of file riscv_meta.h.

Referenced by DynamicMicroOpBoomV1::getPort().

6.273.2.5 has_ifpu

```
bool riscvinstr::has_ifpu
```

Definition at line 334 of file riscv_meta.h.

6.273.2.6 has_mul

```
bool riscvinstr::has_mul
```

Definition at line 330 of file riscv_meta.h.

Referenced by DynamicMicroOpBoomV1::getPort().

6.273.2.7 is_memory

```
bool riscvinstr::is_memory
```

Definition at line 335 of file riscv_meta.h.

Referenced by DynamicMicroOpBoomV1::getPort().

6.273.2.8 opcode

```
unsigned int riscvinstr::opcode
```

Definition at line 328 of file riscv_meta.h.

The documentation for this struct was generated from the following file:

- common/performance_model/performance_models/core_model/ **riscv_meta.h**

6.274 RobContention Class Reference

```
#include <rob_contention.h>
```

Inheritance diagram for RobContention:

Public Member Functions

- virtual void **initCycle** (**SubsecondTime** now)=0
- virtual bool **tryIssue** (const **DynamicMicroOp** &uop)=0
- virtual bool **noMore** ()
- virtual void **doIssue** (**DynamicMicroOp** &uop)=0

Static Public Member Functions

- static **RobContention** * **createRobContentionModel** (**Core** *core, const **CoreModel** *core_model)

6.274.1 Detailed Description

Definition at line 14 of file rob_contention.h.

6.274.2 Member Function Documentation

6.274.2.1 createRobContentionModel()

```
static RobContention* RobContention::createRobContentionModel (
            Core * core,
           const CoreModel * core_model ) [static]
```

6.274.2.2 doIssue()

```
virtual void RobContention::doIssue (
            DynamicMicroOp & uop ) [pure virtual]
```

Implemented in **RobContentionBoomV1** (p. 992), and **RobContentionNehalem** (p. 996).

Referenced by RobTimer::issueInstruction(), and RobSmtTimer::issueInstruction().

6.274.2.3 initCycle()

```
virtual void RobContention::initCycle (
            SubsecondTime now ) [pure virtual]
```

Implemented in **RobContentionBoomV1** (p. 992), and **RobContentionNehalem** (p. 996).

Referenced by RobTimer::doIssue(), and RobSmtTimer::doIssue().

6.274.2.4 noMore()

```
virtual bool RobContention::noMore ( ) [inline], [virtual]
```

Reimplemented in **RobContentionBoomV1** (p. 992), and **RobContentionNehalem** (p. 996).

Definition at line 20 of file rob_contention.h.

Referenced by RobTimer::doIssue(), and RobSmtTimer::tryIssue().

6.274.2.5 tryIssue()

```
virtual bool RobContention::tryIssue (
            const  DynamicMicroOp & uop )  [pure virtual]
```

Implemented in **RobContentionBoomV1** (p. 993), and **RobContentionNehalem** (p. 996).

Referenced by RobTimer::doIssue(), and RobSmtTimer::tryIssue().

The documentation for this class was generated from the following file:

- common/performance_model/performance_models/rob_performance_model/ **rob_contention.h**

6.275 RobContentionBoomV1 Class Reference

```
#include <rob_contention_boom_v1.h>
```

Inheritance diagram for RobContentionBoomV1:

Public Member Functions

- **RobContentionBoomV1** (const **Core** *core, const **CoreModel** *core_model)
- void **initCycle** (**SubsecondTime** now)
- bool **tryIssue** (const **DynamicMicroOp** &uop)
- bool **noMore** ()
- void **doIssue** (**DynamicMicroOp** &uop)

Private Attributes

- const **CoreModel** * **m_core_model**
- uint64_t **m_cache_block_mask**
- **ComponentTime** **m_now**
- bool **ports** [**DynamicMicroOpBoomV1::UOP_PORT_SIZE**]
- int **ports_generic012**
- std::vector< **SubsecondTime** > **alu_used_until**

Additional Inherited Members

6.275.1 Detailed Description

Definition at line 15 of file rob_contention_boom_v1.h.

6.275.2 Constructor & Destructor Documentation

6.275.2.1 RobContentionBoomV1()

```
RobContentionBoomV1::RobContentionBoomV1 (
            const  Core * core,
            const  CoreModel * core_model )
```

Definition at line 13 of file rob_contention_boom_v1.cc.

6.275.3 Member Function Documentation

6.275.3.1 doIssue()

```
void RobContentionBoomV1::doIssue (
            DynamicMicroOp & uop )  [virtual]
```

Implements **RobContention** (p. 990).

Definition at line 69 of file rob_contention_boom_v1.cc.

References alu_used_until, DynamicMicroOpBoomV1::getAlu(), CoreModel::getAluLatency(), DynamicMicroOp↩
::getCoreSpecificInfo(), DynamicMicroOp::getMicroOp(), m_core_model, and m_now.

6.275.3.2 initCycle()

```
void RobContentionBoomV1::initCycle (
            SubsecondTime now )  [virtual]
```

Implements **RobContention** (p. 990).

Definition at line 21 of file rob_contention_boom_v1.cc.

References m_now, ports, ports_generic012, ComponentTime::setElapsedTime(), and DynamicMicroOpBoom↩
V1::UOP_PORT_SIZE.

6.275.3.3 noMore()

```
bool RobContentionBoomV1::noMore ( )  [virtual]
```

Reimplemented from **RobContention** (p. 990).

Definition at line 77 of file rob_contention_boom_v1.cc.

References ports, ports_generic012, and DynamicMicroOpBoomV1::UOP_PORT2.

6.275.3.4 tryIssue()

```
bool RobContentionBoomV1::tryIssue (
            const  DynamicMicroOp & uop )  [virtual]
```

Implements **RobContention** (p. 990).

Definition at line 28 of file rob_contention_boom_v1.cc.

References alu_used_until, DynamicMicroOpBoomV1::getAlu(), DynamicMicroOp::getCoreSpecificInfo(), DynamicMicroOpBoomV1::getPort(), m_now, ports, ports_generic012, and DynamicMicroOpBoomV1::UOP_↩ PORT012.

6.275.4 Member Data Documentation

6.275.4.1 alu_used_until

```
std::vector< SubsecondTime> RobContentionBoomV1::alu_used_until  [private]
```

Definition at line 25 of file rob_contention_boom_v1.h.

Referenced by doIssue(), and tryIssue().

6.275.4.2 m_cache_block_mask

```
uint64_t RobContentionBoomV1::m_cache_block_mask  [private]
```

Definition at line 18 of file rob_contention_boom_v1.h.

6.275.4.3 m_core_model

```
const CoreModel* RobContentionBoomV1::m_core_model [private]
```

Definition at line 17 of file rob_contention_boom_v1.h.

Referenced by doIssue().

6.275.4.4 m_now

```
ComponentTime RobContentionBoomV1::m_now [private]
```

Definition at line 19 of file rob_contention_boom_v1.h.

Referenced by doIssue(), initCycle(), and tryIssue().

6.275.4.5 ports

```
bool RobContentionBoomV1::ports[ DynamicMicroOpBoomV1::UOP_PORT_SIZE] [private]
```

Definition at line 22 of file rob_contention_boom_v1.h.

Referenced by initCycle(), noMore(), and tryIssue().

6.275.4.6 ports_generic012

```
int RobContentionBoomV1::ports_generic012 [private]
```

Definition at line 23 of file rob_contention_boom_v1.h.

Referenced by initCycle(), noMore(), and tryIssue().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/rob_performance_model/ **rob_contention_boom_v1.h**
- common/performance_model/performance_models/rob_performance_model/ **rob_contention_boom_v1.↩ cc**

6.276 RobContentionNehalem Class Reference

`#include <rob_contention_nehalem.h>`

Inheritance diagram for RobContentionNehalem:

Public Member Functions

- **RobContentionNehalem** (const **Core** *core, const **CoreModel** *core_model)
- void **initCycle** (**SubsecondTime** now)
- bool **tryIssue** (const **DynamicMicroOp** &uop)
- bool **noMore** ()
- void **doIssue** (**DynamicMicroOp** &uop)

Private Attributes

- const **CoreModel** * **m_core_model**
- uint64_t **m_cache_block_mask**
- **ComponentTime** **m_now**
- bool **ports** [**DynamicMicroOpNehalem::UOP_PORT_SIZE**]
- int **ports_generic**
- int **ports_generic05**
- std::vector< **SubsecondTime** > **alu_used_until**

Additional Inherited Members

6.276.1 Detailed Description

Definition at line 15 of file rob_contention_nehalem.h.

6.276.2 Constructor & Destructor Documentation

6.276.2.1 RobContentionNehalem()

```
RobContentionNehalem::RobContentionNehalem (
            const  Core * core,
            const  CoreModel * core_model )
```

Definition at line 13 of file rob_contention_nehalem.cc.

6.276.3 Member Function Documentation

6.276.3.1 doIssue()

```
void RobContentionNehalem::doIssue (
              DynamicMicroOp & uop )  [virtual]
```

Implements **RobContention** (p. 990).

Definition at line 87 of file rob_contention_nehalem.cc.

References alu_used_until, DynamicMicroOpNehalem::getAlu(), CoreModel::getAluLatency(), DynamicMicroOp↩ ::getCoreSpecificInfo(), DynamicMicroOp::getMicroOp(), m_core_model, and m_now.

6.276.3.2 initCycle()

```
void RobContentionNehalem::initCycle (
              SubsecondTime now )  [virtual]
```

Implements **RobContention** (p. 990).

Definition at line 21 of file rob_contention_nehalem.cc.

References m_now, ports, ports_generic, ports_generic05, ComponentTime::setElapsedTime(), and Dynamic↩ MicroOpNehalem::UOP_PORT_SIZE.

6.276.3.3 noMore()

```
bool RobContentionNehalem::noMore ( )  [virtual]
```

Reimplemented from **RobContention** (p. 990).

Definition at line 95 of file rob_contention_nehalem.cc.

References ports, ports_generic, DynamicMicroOpNehalem::UOP_PORT2, and DynamicMicroOpNehalem::UO↩ P_PORT34.

6.276.3.4 tryIssue()

```
bool RobContentionNehalem::tryIssue (
            const  DynamicMicroOp & uop )  [virtual]
```

Implements **RobContention** (p. 990).

Definition at line 29 of file rob_contention_nehalem.cc.

References alu_used_until, DynamicMicroOpNehalem::getAlu(), DynamicMicroOp::getCoreSpecificInfo(), DynamicMicroOpNehalem::getPort(), m_now, ports, ports_generic, ports_generic05, DynamicMicroOpNehalem::UOP_PORT015, DynamicMicroOpNehalem::UOP_PORT05, DynamicMicroOpNehalem::UOP_PORT1, DynamicMicroOpNehalem::UOP_PORT2, and DynamicMicroOpNehalem::UOP_PORT34.

6.276.4 Member Data Documentation

6.276.4.1 alu_used_until

```
std::vector< SubsecondTime> RobContentionNehalem::alu_used_until  [private]
```

Definition at line 25 of file rob_contention_nehalem.h.

Referenced by doIssue(), and tryIssue().

6.276.4.2 m_cache_block_mask

```
uint64_t RobContentionNehalem::m_cache_block_mask  [private]
```

Definition at line 18 of file rob_contention_nehalem.h.

6.276.4.3 m_core_model

```
const  CoreModel* RobContentionNehalem::m_core_model  [private]
```

Definition at line 17 of file rob_contention_nehalem.h.

Referenced by doIssue().

6.276.4.4 m_now

ComponentTime RobContentionNehalem::m_now [private]

Definition at line 19 of file rob_contention_nehalem.h.

Referenced by doIssue(), initCycle(), and tryIssue().

6.276.4.5 ports

bool RobContentionNehalem::ports[**DynamicMicroOpNehalem::UOP_PORT_SIZE**] [private]

Definition at line 22 of file rob_contention_nehalem.h.

Referenced by initCycle(), noMore(), and tryIssue().

6.276.4.6 ports_generic

int RobContentionNehalem::ports_generic [private]

Definition at line 23 of file rob_contention_nehalem.h.

Referenced by initCycle(), noMore(), and tryIssue().

6.276.4.7 ports_generic05

int RobContentionNehalem::ports_generic05 [private]

Definition at line 23 of file rob_contention_nehalem.h.

Referenced by initCycle(), and tryIssue().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/rob_performance_model/ **rob_contention_nehalem.h**
- common/performance_model/performance_models/rob_performance_model/ **rob_contention_nehalem.cc**

6.277 RobTimer::RobEntry Class Reference

Public Member Functions

- void **init** (**DynamicMicroOp** * **uop**, **UInt64** sequenceNumber)
- void **free** ()
- void **addDependant** (**RobEntry** *dep)
- uint64_t **getNumDependants** () const
- **RobEntry** * **getDependant** (size_t idx) const
- void **addAddressProducer** (**UInt64** sequenceNumber)
- **UInt64** **getNumAddressProducers** () const
- **UInt64** **getAddressProducer** (size_t idx) const

Public Attributes

- **DynamicMicroOp** * **uop**
- **SubsecondTime dispatched**
- **SubsecondTime ready**
- **SubsecondTime readyMax**
- **SubsecondTime addressReady**
- **SubsecondTime addressReadyMax**
- **SubsecondTime issued**
- **SubsecondTime done**

Private Attributes

- size_t **numInlineDependants**
- **RobEntry** * **inlineDependants** [**MAX_INLINE_DEPENDANTS**]
- std::vector< **RobEntry** * > * **vectorDependants**
- std::vector< uint64_t > **addressProducers**

Static Private Attributes

- static const size_t **MAX_INLINE_DEPENDANTS** = 8

6.277.1 Detailed Description

Definition at line 17 of file rob_timer.h.

6.277.2 Member Function Documentation

6.277.2.1 addAddressProducer()

```
void RobTimer::RobEntry::addAddressProducer (
              UInt64 sequenceNumber )  [inline]
```

Definition at line 34 of file rob_timer.h.

References addressProducers.

Referenced by RobTimer::simulate().

6.277.2.2 addDependant()

```
void RobTimer::RobEntry::addDependant (
            RobTimer::RobEntry * dep )
```

Definition at line 207 of file rob_timer.cc.

Referenced by RobTimer::simulate().

6.277.2.3 free()

```
void RobTimer::RobEntry::free ( )
```

Definition at line 200 of file rob_timer.cc.

Referenced by RobTimer::doCommit().

6.277.2.4 getAddressProducer()

```
UInt64 RobTimer::RobEntry::getAddressProducer (
            size_t idx ) const  [inline]
```

Definition at line 36 of file rob_timer.h.

References addressProducers.

Referenced by RobTimer::issueInstruction(), and RobTimer::simulate().

6.277.2.5 getDependant()

```
RobTimer::RobEntry * RobTimer::RobEntry::getDependant (
            size_t idx ) const
```

Definition at line 228 of file rob_timer.cc.

References LOG_ASSERT_ERROR.

Referenced by RobTimer::issueInstruction(), and RobTimer::simulate().

6.277.2.6 getNumAddressProducers()

```
UInt64 RobTimer::RobEntry::getNumAddressProducers ( ) const  [inline]
```

Definition at line 35 of file rob_timer.h.

References addressProducers.

Referenced by RobTimer::issueInstruction(), and RobTimer::simulate().

6.277.2.7 getNumDependants()

```
uint64_t RobTimer::RobEntry::getNumDependants ( ) const
```

Definition at line 223 of file rob_timer.cc.

Referenced by RobTimer::issueInstruction(), and RobTimer::simulate().

6.277.2.8 init()

```
void RobTimer::RobEntry::init (
              DynamicMicroOp * uop,
              UInt64 sequenceNumber )
```

Definition at line 182 of file rob_timer.cc.

References addressProducers, addressReady, addressReadyMax, done, issued, SubsecondTime::MaxTime(), numInlineDependants, ready, readyMax, DynamicMicroOp::setSequenceNumber(), uop, vectorDependants, and SubsecondTime::Zero().

Referenced by RobTimer::simulate().

6.277.3 Member Data Documentation

6.277.3.1 addressProducers

```
std::vector<uint64_t> RobTimer::RobEntry::addressProducers  [private]
```

Definition at line 24 of file rob_timer.h.

Referenced by addAddressProducer(), getAddressProducer(), getNumAddressProducers(), and init().

6.277.3.2 addressReady

SubsecondTime RobTimer::RobEntry::addressReady

Definition at line 42 of file rob_timer.h.

Referenced by RobTimer::doIssue(), init(), RobTimer::issueInstruction(), and RobTimer::simulate().

6.277.3.3 addressReadyMax

SubsecondTime RobTimer::RobEntry::addressReadyMax

Definition at line 43 of file rob_timer.h.

Referenced by init(), RobTimer::issueInstruction(), and RobTimer::simulate().

6.277.3.4 dispatched

SubsecondTime RobTimer::RobEntry::dispatched

Definition at line 39 of file rob_timer.h.

Referenced by RobTimer::doCommit(), and RobTimer::doDispatch().

6.277.3.5 done

SubsecondTime RobTimer::RobEntry::done

Definition at line 45 of file rob_timer.h.

Referenced by RobTimer::countOutstandingMemop(), RobTimer::doCommit(), RobTimer::doIssue(), RobTimer::findCpiComponent(), init(), RobTimer::issueInstruction(), RobTimer::printRob(), and RobTimer::simulate().

6.277.3.6 inlineDependants

RobEntry* RobTimer::RobEntry::inlineDependants[**MAX_INLINE_DEPENDANTS**] [private]

Definition at line 22 of file rob_timer.h.

6.277.3.7 issued

SubsecondTime RobTimer::RobEntry::issued

Definition at line 44 of file rob_timer.h.

Referenced by RobTimer::doCommit(), init(), and RobTimer::issueInstruction().

6.277.3.8 MAX_INLINE_DEPENDANTS

const size_t RobTimer::RobEntry::MAX_INLINE_DEPENDANTS = 8 [static], [private]

Definition at line 20 of file rob_timer.h.

6.277.3.9 numInlineDependants

size_t RobTimer::RobEntry::numInlineDependants [private]

Definition at line 21 of file rob_timer.h.

Referenced by init().

6.277.3.10 ready

SubsecondTime RobTimer::RobEntry::ready

Definition at line 40 of file rob_timer.h.

Referenced by RobTimer::doDispatch(), RobTimer::doIssue(), init(), RobTimer::issueInstruction(), RobTimer↩ ::printRob(), and RobTimer::simulate().

6.277.3.11 readyMax

SubsecondTime RobTimer::RobEntry::readyMax

Definition at line 41 of file rob_timer.h.

Referenced by init(), RobTimer::issueInstruction(), and RobTimer::simulate().

6.277.3.12 uop

```
DynamicMicroOp* RobTimer::RobEntry::uop
```

Definition at line 38 of file rob_timer.h.

Referenced by RobTimer::countOutstandingMemop(), RobTimer::doCommit(), RobTimer::doDispatch(), RobTimer::doIssue(), RobTimer::findCpiComponent(), RobTimer::findEntryBySequenceNumber(), init(), RobTimer::issueInstruction(), RobTimer::printRob(), and RobTimer::simulate().

6.277.3.13 vectorDependants

```
std::vector< RobEntry*>* RobTimer::RobEntry::vectorDependants  [private]
```

Definition at line 23 of file rob_timer.h.

Referenced by init().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/rob_performance_model/ **rob_timer.h**
- common/performance_model/performance_models/rob_performance_model/ **rob_timer.cc**

6.278 RobSmtTimer::RobEntry Class Reference

Public Member Functions

- void **init** (**DynamicMicroOp** * **uop**, **UInt64** sequenceNumber)
- void **free** ()
- void **addDependant** (**RobEntry** *dep)
- uint64_t **getNumDependants** () const
- **RobEntry** * **getDependant** (size_t idx) const
- void **addAddressProducer** (uint64_t sequenceNumber)
- size_t **getNumAddressProducers** () const
- uint64_t **getAddressProducer** (size_t idx) const

Public Attributes

- **DynamicMicroOp** * **uop**
- **SubsecondTime** **dispatched**
- **SubsecondTime** **ready**
- **SubsecondTime** **readyMax**
- **SubsecondTime** **addressReady**
- **SubsecondTime** **addressReadyMax**
- **SubsecondTime** **issued**
- **SubsecondTime** **done**

Private Attributes

- size_t **numInlineDependants**
- **RobEntry** * **inlineDependants** [**MAX_INLINE_DEPENDANTS**]
- std::vector< **RobEntry** * > * **vectorDependants**
- size_t **numAddressProducers**
- uint64_t **addressProducers** [**MAX_ADDRESS_PRODUCERS**]

Static Private Attributes

- static const size_t **MAX_INLINE_DEPENDANTS** = 8
- static const size_t **MAX_ADDRESS_PRODUCERS** = 4

6.278.1 Detailed Description

Definition at line 16 of file rob_smt_timer.h.

6.278.2 Member Function Documentation

6.278.2.1 addAddressProducer()

```
void RobSmtTimer::RobEntry::addAddressProducer (
            uint64_t sequenceNumber )  [inline]
```

Definition at line 35 of file rob_smt_timer.h.

References addressProducers, LOG_ASSERT_ERROR, MAX_ADDRESS_PRODUCERS, and numAddress↩ Producers.

Referenced by RobSmtTimer::setStoreAddressProducers().

6.278.2.2 addDependant()

```
void RobSmtTimer::RobEntry::addDependant (
              RobSmtTimer::RobEntry * dep )
```

Definition at line 260 of file rob_smt_timer.cc.

Referenced by RobSmtTimer::setDependencies().

6.278.2.3 free()

```
void RobSmtTimer::RobEntry::free ( )
```

Definition at line 253 of file rob_smt_timer.cc.

Referenced by RobSmtTimer::doCommit().

6.278.2.4 getAddressProducer()

```
uint64_t RobSmtTimer::RobEntry::getAddressProducer (
            size_t idx ) const  [inline]
```

Definition at line 41 of file rob_smt_timer.h.

References addressProducers.

Referenced by RobSmtTimer::issueInstruction(), and RobSmtTimer::setDependencies().

6.278.2.5 getDependant()

```
RobSmtTimer::RobEntry * RobSmtTimer::RobEntry::getDependant (
            size_t idx ) const
```

Definition at line 281 of file rob_smt_timer.cc.

References LOG_ASSERT_ERROR.

Referenced by RobSmtTimer::issueInstruction(), and RobSmtTimer::setDependencies().

6.278.2.6 getNumAddressProducers()

```
size_t RobSmtTimer::RobEntry::getNumAddressProducers ( ) const  [inline]
```

Definition at line 40 of file rob_smt_timer.h.

References numAddressProducers.

Referenced by RobSmtTimer::issueInstruction(), RobSmtTimer::setDependencies(), and RobSmtTimer::setStore↩
AddressProducers().

6.278.2.7 getNumDependants()

```
uint64_t RobSmtTimer::RobEntry::getNumDependants ( ) const
```

Definition at line 276 of file rob_smt_timer.cc.

Referenced by RobSmtTimer::issueInstruction(), and RobSmtTimer::setDependencies().

6.278.2.8 init()

```
void RobSmtTimer::RobEntry::init (
              DynamicMicroOp * uop,
              UInt64 sequenceNumber )
```

Definition at line 234 of file rob_smt_timer.cc.

References addressReady, addressReadyMax, dispatched, done, issued, SubsecondTime::MaxTime(), numAddressProducers, numInlineDependants, ready, readyMax, DynamicMicroOp::setSequenceNumber(), uop, vectorDependants, and SubsecondTime::Zero().

Referenced by RobSmtTimer::pushInstructions().

6.278.3 Member Data Documentation

6.278.3.1 addressProducers

```
uint64_t RobSmtTimer::RobEntry::addressProducers[ MAX_ADDRESS_PRODUCERS]  [private]
```

Definition at line 25 of file rob_smt_timer.h.

Referenced by addAddressProducer(), and getAddressProducer().

6.278.3.2 addressReady

```
SubsecondTime RobSmtTimer::RobEntry::addressReady
```

Definition at line 47 of file rob_smt_timer.h.

Referenced by init(), RobSmtTimer::issueInstruction(), RobSmtTimer::setStoreAddressProducers(), and RobSmtTimer::tryIssue().

6.278.3.3 addressReadyMax

SubsecondTime RobSmtTimer::RobEntry::addressReadyMax

Definition at line 48 of file rob_smt_timer.h.

Referenced by init(), RobSmtTimer::issueInstruction(), and RobSmtTimer::setStoreAddressProducers().

6.278.3.4 dispatched

SubsecondTime RobSmtTimer::RobEntry::dispatched

Definition at line 44 of file rob_smt_timer.h.

Referenced by RobSmtTimer::doCommit(), init(), and RobSmtTimer::tryDispatch().

6.278.3.5 done

SubsecondTime RobSmtTimer::RobEntry::done

Definition at line 50 of file rob_smt_timer.h.

Referenced by RobSmtTimer::countOutstandingMemop(), RobSmtTimer::doCommit(), RobSmtTimer::findCpiComponent(), init(), RobSmtTimer::issueInstruction(), RobSmtTimer::printRob(), RobSmtTimer::setDependencies(), RobSmtTimer::setStoreAddressProducers(), and RobSmtTimer::tryIssue().

6.278.3.6 inlineDependants

RobEntry* RobSmtTimer::RobEntry::inlineDependants[**MAX_INLINE_DEPENDANTS**] [private]

Definition at line 20 of file rob_smt_timer.h.

6.278.3.7 issued

SubsecondTime RobSmtTimer::RobEntry::issued

Definition at line 49 of file rob_smt_timer.h.

Referenced by RobSmtTimer::doCommit(), RobSmtTimer::findCpiComponent(), init(), and RobSmtTimer::issueInstruction().

6.278.3.8 MAX_ADDRESS_PRODUCERS

```
const size_t RobSmtTimer::RobEntry::MAX_ADDRESS_PRODUCERS = 4  [static], [private]
```

Definition at line 23 of file rob_smt_timer.h.

Referenced by addAddressProducer().

6.278.3.9 MAX_INLINE_DEPENDANTS

```
const size_t RobSmtTimer::RobEntry::MAX_INLINE_DEPENDANTS = 8  [static], [private]
```

Definition at line 18 of file rob_smt_timer.h.

6.278.3.10 numAddressProducers

```
size_t RobSmtTimer::RobEntry::numAddressProducers  [private]
```

Definition at line 24 of file rob_smt_timer.h.

Referenced by addAddressProducer(), getNumAddressProducers(), and init().

6.278.3.11 numInlineDependants

```
size_t RobSmtTimer::RobEntry::numInlineDependants  [private]
```

Definition at line 19 of file rob_smt_timer.h.

Referenced by init().

6.278.3.12 ready

```
SubsecondTime RobSmtTimer::RobEntry::ready
```

Definition at line 45 of file rob_smt_timer.h.

Referenced by init(), RobSmtTimer::issueInstruction(), RobSmtTimer::printRob(), RobSmtTimer::setDependencies(), RobSmtTimer::tryDispatch(), and RobSmtTimer::tryIssue().

6.278.3.13 readyMax

```
SubsecondTime RobSmtTimer::RobEntry::readyMax
```

Definition at line 46 of file rob_smt_timer.h.

Referenced by init(), RobSmtTimer::issueInstruction(), and RobSmtTimer::setDependencies().

6.278.3.14 uop

```
DynamicMicroOp* RobSmtTimer::RobEntry::uop
```

Definition at line 43 of file rob_smt_timer.h.

Referenced by RobSmtTimer::countOutstandingMemop(), RobSmtTimer::doCommit(), RobSmtTimer::findCpi↩Component(), RobSmtTimer::findEntryBySequenceNumber(), init(), RobSmtTimer::issueInstruction(), RobSmt↩Timer::printRob(), RobSmtTimer::pushInstructions(), RobSmtTimer::setDependencies(), RobSmtTimer::setStore↩AddressProducers(), RobSmtTimer::tryDispatch(), and RobSmtTimer::tryIssue().

6.278.3.15 vectorDependants

```
std::vector< RobEntry*>* RobSmtTimer::RobEntry::vectorDependants  [private]
```

Definition at line 21 of file rob_smt_timer.h.

Referenced by init().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/rob_performance_model/ **rob_smt_timer.h**
- common/performance_model/performance_models/rob_performance_model/ **rob_smt_timer.cc**

6.279 RobPerformanceModel Class Reference

```
#include <rob_performance_model.h>
```

Inheritance diagram for RobPerformanceModel:

Public Member Functions

- **RobPerformanceModel** (**Core** ∗core)
- ∼**RobPerformanceModel** ()

Protected Member Functions

- virtual boost::tuple< uint64_t, uint64_t > **simulate** (const std::vector< **DynamicMicroOp** ∗ > &insts)
- virtual void **notifyElapsedTimeUpdate** ()

Private Attributes

- **RobTimer rob_timer**

Additional Inherited Members

6.279.1 Detailed Description

Definition at line 7 of file rob_performance_model.h.

6.279.2 Constructor & Destructor Documentation

6.279.2.1 RobPerformanceModel()

```
RobPerformanceModel::RobPerformanceModel (
              Core * core )
```

Definition at line 4 of file rob_performance_model.cc.

6.279.2.2 ∼RobPerformanceModel()

```
RobPerformanceModel::~RobPerformanceModel ( )
```

Definition at line 16 of file rob_performance_model.cc.

6.279.3 Member Function Documentation

6.279.3.1 notifyElapsedTimeUpdate()

```
void RobPerformanceModel::notifyElapsedTimeUpdate ( )  [protected], [virtual]
```

Implements **MicroOpPerformanceModel** (p. 788).

Definition at line 28 of file rob_performance_model.cc.

References ComponentTime::getElapsedTime(), PerformanceModel::m_elapsed_time, rob_timer, and RobTimer::synchronize().

6.279.3.2 simulate()

```
boost::tuple< uint64_t, uint64_t > RobPerformanceModel::simulate (
            const std::vector<  DynamicMicroOp * > & insts )  [protected], [virtual]
```

Implements **MicroOpPerformanceModel** (p. 789).

Definition at line 20 of file rob_performance_model.cc.

References SubsecondTime::divideRounded(), ComponentTime::getPeriod(), PerformanceModel::m_elapsed_time, rob_timer, and RobTimer::simulate().

6.279.4 Member Data Documentation

6.279.4.1 rob_timer

```
RobTimer RobPerformanceModel::rob_timer  [private]
```

Definition at line 16 of file rob_performance_model.h.

Referenced by notifyElapsedTimeUpdate(), and simulate().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/ **rob_performance_model.h**
- common/performance_model/performance_models/ **rob_performance_model.cc**

6.280 RobSmtPerformanceModel Class Reference

```
#include <rob_smt_performance_model.h>
```

Inheritance diagram for RobSmtPerformanceModel:

Public Member Functions

- **RobSmtPerformanceModel** (**Core** ∗core)
- ∼**RobSmtPerformanceModel** ()

Protected Member Functions

- virtual boost::tuple< uint64_t, uint64_t > **simulate** (const std::vector< **DynamicMicroOp** ∗ > &insts)
- virtual void **notifyElapsedTimeUpdate** ()
- virtual void **enableDetailedModel** ()
- virtual void **disableDetailedModel** ()
- virtual void **synchronize** ()

Static Private Member Functions

- static **RobSmtTimer** ∗ **getRobTimer** (**Core** ∗core, **RobSmtPerformanceModel** ∗perf, const **CoreModel** ∗core_model)

Private Attributes

- **RobSmtTimer** ∗ **m_rob_timer**
- **UInt8 m_thread_id**
- bool **m_enabled**

Static Private Attributes

- static std::unordered_map< **core_id_t**, **RobSmtTimer** ∗ > **s_rob_timers**

Additional Inherited Members

6.280.1 Detailed Description

Definition at line 9 of file rob_smt_performance_model.h.

6.280.2 Constructor & Destructor Documentation

6.280.2.1 RobSmtPerformanceModel()

```
RobSmtPerformanceModel::RobSmtPerformanceModel (
            Core * core )
```

Definition at line 36 of file rob_smt_performance_model.cc.

References m_rob_timer, m_thread_id, and SmtTimer::registerThread().

6.280.2.2 ~RobSmtPerformanceModel()

```
RobSmtPerformanceModel::~RobSmtPerformanceModel ( )
```

Definition at line 44 of file rob_smt_performance_model.cc.

References PerformanceModel::getCore(), Core::getId(), and s_rob_timers.

6.280.3 Member Function Documentation

6.280.3.1 disableDetailedModel()

```
void RobSmtPerformanceModel::disableDetailedModel ( )  [protected], [virtual]
```

Reimplemented from **PerformanceModel** (p. 908).

Definition at line 83 of file rob_smt_performance_model.cc.

References SmtTimer::disable(), m_enabled, SmtTimer::m_lock, and m_rob_timer.

6.280.3.2 enableDetailedModel()

```
void RobSmtPerformanceModel::enableDetailedModel ( )  [protected], [virtual]
```

Reimplemented from **PerformanceModel** (p. 908).

Definition at line 76 of file rob_smt_performance_model.cc.

References SmtTimer::enable(), m_enabled, SmtTimer::m_lock, and m_rob_timer.

6.280.3.3 getRobTimer()

```
RobSmtTimer * RobSmtPerformanceModel::getRobTimer (
            Core * core,
            RobSmtPerformanceModel * perf,
           const  CoreModel * core_model )  [static], [private]
```

Definition at line 6 of file rob_smt_performance_model.cc.

References Core::getId(), and s_rob_timers.

6.280.3.4 notifyElapsedTimeUpdate()

```
void RobSmtPerformanceModel::notifyElapsedTimeUpdate ( )  [protected], [virtual]
```

Implements **MicroOpPerformanceModel** (p. 788).

Definition at line 70 of file rob_smt_performance_model.cc.

References ComponentTime::getElapsedTime(), PerformanceModel::m_elapsed_time, SmtTimer::m_lock, m_↩ rob_timer, m_thread_id, and RobSmtTimer::synchronize().

6.280.3.5 simulate()

```
boost::tuple< uint64_t, uint64_t > RobSmtPerformanceModel::simulate (
            const std::vector< DynamicMicroOp * > & insts )  [protected], [virtual]
```

Implements **MicroOpPerformanceModel** (p. 789).

Definition at line 51 of file rob_smt_performance_model.cc.

References SubsecondTime::divideRounded(), ComponentTime::getPeriod(), PerformanceModel::m_elapsed_↩ time, SmtTimer::m_lock, m_rob_timer, m_thread_id, RobSmtTimer::pushInstructions(), and RobSmtTimer::return↩ Latency().

6.280.3.6 synchronize()

```
void RobSmtPerformanceModel::synchronize ( )  [protected], [virtual]
```

Reimplemented from **PerformanceModel** (p. 914).

Definition at line 63 of file rob_smt_performance_model.cc.

References SmtTimer::m_lock, m_rob_timer, m_thread_id, and SmtTimer::simulate().

6.280.4 Member Data Documentation

6.280.4.1 m_enabled

```
bool RobSmtPerformanceModel::m_enabled  [private]
```

Definition at line 24 of file rob_smt_performance_model.h.

Referenced by disableDetailedModel(), and enableDetailedModel().

6.280.4.2 m_rob_timer

```
RobSmtTimer* RobSmtPerformanceModel::m_rob_timer [private]
```

Definition at line 22 of file rob_smt_performance_model.h.

Referenced by disableDetailedModel(), enableDetailedModel(), notifyElapsedTimeUpdate(), RobSmtPerformance↩Model(), simulate(), and synchronize().

6.280.4.3 m_thread_id

```
UInt8 RobSmtPerformanceModel::m_thread_id [private]
```

Definition at line 23 of file rob_smt_performance_model.h.

Referenced by notifyElapsedTimeUpdate(), RobSmtPerformanceModel(), simulate(), and synchronize().

6.280.4.4 s_rob_timers

```
std::unordered_map< core_id_t, RobSmtTimer * > RobSmtPerformanceModel::s_rob_timers [static],
[private]
```

Definition at line 26 of file rob_smt_performance_model.h.

Referenced by getRobTimer(), and ∼RobSmtPerformanceModel().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/ **rob_smt_performance_model.h**
- common/performance_model/performance_models/ **rob_smt_performance_model.cc**

6.281 RobSmtTimer Class Reference

```
#include <rob_smt_timer.h>
```

Inheritance diagram for RobSmtTimer:

Classes

- class **RobEntry**
- class **RobThread**

Public Member Functions

- **RobSmtTimer** (int num_threads, **Core** *core, **PerformanceModel** * **perf**, const **CoreModel** *core_model, int **misprediction_penalty**, int dispatch_width, int window_size)
- virtual ~**RobSmtTimer** ()
- virtual void **initializeThread** (**smtthread_id_t** thread_num)
- virtual uint64_t **threadNumSurplusInstructions** (**smtthread_id_t** thread_num)
- virtual bool **threadHasEnoughInstructions** (**smtthread_id_t** thread_num)
- virtual void **notifyNumActiveThreadsChange** ()
- virtual void **pushInstructions** (**smtthread_id_t** thread_id, const std::vector< **DynamicMicroOp** * > &insts)
- virtual boost::tuple< uint64_t, **SubsecondTime** > **returnLatency** (**smtthread_id_t** thread_id)
- virtual void **execute** ()
- virtual void **synchronize** (**smtthread_id_t** thread_id, **SubsecondTime** time)

Private Types

- typedef **CircularQueue**< **RobEntry** > **Rob**

Private Member Functions

- void **setDependencies** (**smtthread_id_t** thread_id, **RobEntry** *entry)
- void **setStoreAddressProducers** (**smtthread_id_t** thread_id, **RobEntry** *entry, uint64_t lowestValidSequenceNumber)
- **RobEntry** * **findEntryBySequenceNumber** (**smtthread_id_t** thread_num, **UInt64** sequenceNumber)
- void **countOutstandingMemop** (**smtthread_id_t** thread_num, **SubsecondTime** time)
- void **printRob** (**smtthread_id_t** thread_num)
- void **printRob** ()
- **SubsecondTime** **executeCycle** ()
- **SubsecondTime** **doDispatch** ()
- **SubsecondTime** **doIssue** ()
- **SubsecondTime** **doCommit** ()
- bool **canExecute** (**smtthread_id_t** thread_num)
- bool **canExecute** ()
- bool **tryDispatch** (**smtthread_id_t** thread_num, **SubsecondTime** &next_event)
- **SubsecondTime** * **findCpiComponent** (**smtthread_id_t** thread_num)
- bool **tryIssue** (**smtthread_id_t** thread_num, **SubsecondTime** &next_event, bool would_have_skipped=false)
- void **issueInstruction** (**smtthread_id_t** thread_num, uint64_t idx, **SubsecondTime** &next_event)
- void **computeCurrentWindowsize** ()

Private Attributes

- const uint64_t **dispatchWidth**
- const uint64_t **commitWidth**
- const uint64_t **windowSize**
- const uint64_t **rsEntries**
- uint64_t **currentWindowSize**
- const uint64_t **misprediction_penalty**
- const bool **m_store_to_load_forwarding**
- const bool **m_no_address_disambiguation**
- const bool **inorder**
- const bool **windowRepartition**
- const bool **simultaneousIssue**
- std::vector< **RobThread** * > **m_rob_threads**
- **RobContention** * **m_rob_contention**
- **UInt8 dispatch_thread**
- **UInt8 issue_thread**
- **ComponentTime now**
- **SubsecondTime last_store_done**
- **ContentionModel load_queue**
- **ContentionModel store_queue**
- uint64_t **m_rs_entries_used**
- bool **will_skip**
- **SubsecondTime time_skipped**
- int **addressMask**
- uint64_t **m_numICacheOverlapped**
- uint64_t **m_numBPredOverlapped**
- uint64_t **m_numDCacheOverlapped**
- uint64_t **m_numLongLatencyLoads**
- uint64_t **m_numTotalLongLatencyLoadLatency**
- uint64_t **m_numSerializationInsns**
- uint64_t **m_totalSerializationLatency**
- uint64_t **m_totalHiddenDCacheLatency**
- uint64_t **m_totalHiddenLongerDCacheLatency**
- uint64_t **m_numHiddenLongerDCacheLatency**
- **PerformanceModel** * **perf**
- uint64_t **m_numMfenceInsns**
- uint64_t **m_totalMfenceLatency**
- const bool **m_mlp_histogram**

Static Private Attributes

- static const unsigned int **MAX_OUTSTANDING** = 32

Additional Inherited Members

6.281.1 Detailed Description

Definition at line 14 of file rob_smt_timer.h.

6.281.2 Member Typedef Documentation

6.281.2.1 Rob

```
typedef CircularQueue< RobEntry> RobSmtTimer::Rob [private]
```

Definition at line 52 of file rob_smt_timer.h.

6.281.3 Constructor & Destructor Documentation

6.281.3.1 RobSmtTimer()

```
RobSmtTimer::RobSmtTimer (
            int num_threads,
             Core * core,
             PerformanceModel * perf,
            const  CoreModel * core_model,
            int misprediction_penalty,
            int dispatch_width,
            int window_size )
```

Definition at line 52 of file rob_smt_timer.cc.

References computeCurrentWindowsize(), Core::getId(), m_numMfenceInsns, m_numSerializationInsns, m_totalHiddenDCacheLatency, m_totalMfenceLatency, m_totalSerializationLatency, registerStatsMetric(), and time_skipped.

6.281.3.2 ~RobSmtTimer()

```
RobSmtTimer::~RobSmtTimer ( ) [virtual]
```

Definition at line 104 of file rob_smt_timer.cc.

References m_rob_threads.

6.281.4 Member Function Documentation

6.281.4.1 canExecute() [1/2]

```
bool RobSmtTimer::canExecute ( )  [private]
```

Definition at line 1116 of file rob_smt_timer.cc.

References dispatchWidth, RobSmtTimer::RobThread::m_num_in_rob, m_rob_threads, SmtTimer::m_threads, RobSmtTimer::RobThread::rob, and CircularQueue< T >::size().

Referenced by execute().

6.281.4.2 canExecute() [2/2]

```
bool RobSmtTimer::canExecute (
            smtthread_id_t thread_num )  [private]
```

Definition at line 1096 of file rob_smt_timer.cc.

References currentWindowSize, RobSmtTimer::RobThread::frontend_stalled_until, RobSmtTimer::RobThread::m_num_in_rob, m_rob_threads, SmtTimer::m_threads, RobSmtTimer::RobThread::now, now, and SmtTimer::SmtThread::running.

6.281.4.3 computeCurrentWindowsize()

```
void RobSmtTimer::computeCurrentWindowsize ( )  [private]
```

Definition at line 110 of file rob_smt_timer.cc.

References currentWindowSize, SmtTimer::m_num_threads, SmtTimer::m_threads, windowRepartition, and windowSize.

Referenced by notifyNumActiveThreadsChange(), and RobSmtTimer().

6.281.4.4 countOutstandingMemop()

```
void RobSmtTimer::countOutstandingMemop (
            smtthread_id_t thread_num,
            SubsecondTime time )  [private]
```

Definition at line 1213 of file rob_smt_timer.cc.

References CircularQueue< T >::at(), RobSmtTimer::RobEntry::done, DynamicMicroOp::getDCacheHitWhere(), DynamicMicroOp::getMicroOp(), MicroOp::isLoad(), RobSmtTimer::RobThread::m_num_in_rob, RobSmtTimer::RobThread::m_outstandingLoads, RobSmtTimer::RobThread::m_outstandingLoadsAll, m_rob_threads, MAX_OUTSTANDING, SubsecondTime::MaxTime(), now, HitWhere::NUM_HITWHERES, RobSmtTimer::RobThread::rob, and RobSmtTimer::RobEntry::uop.

Referenced by executeCycle().

6.281.4.5 doCommit()

```
SubsecondTime RobSmtTimer::doCommit ( )  [private]
```

Definition at line 1049 of file rob_smt_timer.cc.

References commitWidth, RobSmtTimer::RobThread::core, RobSmtTimer::RobEntry::dispatched, RobSmtTimer::RobEntry::done, RobSmtTimer::RobEntry::free(), CircularQueue< T >::front(), DynamicMicroOp::getMicroOp(), Core::getPerformanceModel(), RobSmtTimer::RobThread::instrs, DynamicMicroOp::isLast(), RobSmtTimer::RobEntry::issued, LOG_ASSERT_ERROR, RobSmtTimer::RobThread::m_num_in_rob, m_rob_threads, SmtTimer::m_threads, SubsecondTime::MaxTime(), now, CircularQueue< T >::pop(), RobSmtTimer::RobThread::rob, CircularQueue< T >::size(), MicroOp::toShortString(), PerformanceModel::traceInstruction(), RobSmtTimer::RobEntry::uop, and will_skip.

Referenced by executeCycle().

6.281.4.6 doDispatch()

```
SubsecondTime RobSmtTimer::doDispatch ( )  [private]
```

Definition at line 671 of file rob_smt_timer.cc.

References currentWindowSize, dispatch_thread, findCpiComponent(), RobSmtTimer::RobThread::frontend_stalled_until, ComponentTime::getPeriod(), LOG_ASSERT_ERROR, RobSmtTimer::RobThread::m_cpiBase, RobSmtTimer::RobThread::m_cpiCurrentFrontEndStall, RobSmtTimer::RobThread::m_cpiIdle, RobSmtTimer::RobThread::m_cpiSMT, RobSmtTimer::RobThread::m_num_in_rob, m_rob_threads, SmtTimer::m_threads, SubsecondTime::MaxTime(), RobSmtTimer::RobThread::now, now, SmtTimer::SmtThread::running, and tryDispatch().

Referenced by executeCycle().

6.281.4.7 doIssue()

```
SubsecondTime RobSmtTimer::doIssue ( )  [private]
```

Definition at line 1012 of file rob_smt_timer.cc.

References RobContention::initCycle(), issue_thread, m_rob_contention, m_rob_threads, SmtTimer::m_threads, SubsecondTime::MaxTime(), RobSmtTimer::RobThread::next_event, now, simultaneousIssue, and tryIssue().

Referenced by executeCycle().

6.281.4.8 execute()

```
void RobSmtTimer::execute ( )  [virtual]
```

Implements **SmtTimer** (p. 1256).

Definition at line 510 of file rob_smt_timer.cc.

References canExecute(), and executeCycle().

6.281.4.9 executeCycle()

```
SubsecondTime RobSmtTimer::executeCycle ( )  [private]
```

Definition at line 1137 of file rob_smt_timer.cc.

References countOutstandingMemop(), doCommit(), doDispatch(), doIssue(), ComponentTime::getPeriod(), m_↩ mlp_histogram, SmtTimer::m_threads, SubsecondTime::MaxTime(), now, printRob(), time_skipped, will_skip, and SubsecondTime::Zero().

Referenced by execute().

6.281.4.10 findCpiComponent()

```
SubsecondTime * RobSmtTimer::findCpiComponent (
            smtthread_id_t thread_num )  [private]
```

Definition at line 642 of file rob_smt_timer.cc.

References CircularQueue< T >::at(), RobSmtTimer::RobEntry::done, DynamicMicroOp::getDCacheHitWhere(), DynamicMicroOp::getMicroOp(), MicroOp::isLoad(), MicroOp::isMemBarrier(), MicroOp::isSerializing(), MicroOp↩ ::isStore(), RobSmtTimer::RobEntry::issued, RobSmtTimer::RobThread::m_cpiDataCache, RobSmtTimer::Rob↩ Thread::m_cpiSerialization, RobSmtTimer::RobThread::m_num_in_rob, m_rob_threads, now, RobSmtTimer::↩ RobThread::rob, and RobSmtTimer::RobEntry::uop.

Referenced by doDispatch().

6.281.4.11 findEntryBySequenceNumber()

```
RobSmtTimer::RobEntry * RobSmtTimer::findEntryBySequenceNumber (
            smtthread_id_t thread_num,
            UInt64 sequenceNumber )  [private]
```

Definition at line 426 of file rob_smt_timer.cc.

References DynamicMicroOp::getSequenceNumber(), LOG_ASSERT_ERROR, m_rob_threads, CircularQueue< T >::size(), and RobSmtTimer::RobEntry::uop.

Referenced by issueInstruction(), setDependencies(), and setStoreAddressProducers().

6.281.4.12 initializeThread()

```
void RobSmtTimer::initializeThread (
              smtthread_id_t thread_num ) [virtual]
```

Implements **SmtTimer** (p. 1259).

Definition at line 127 of file rob_smt_timer.cc.

References Core::getId(), MicroOp::getSubtypeString(), HitWhereIsValid(), HitWhereString(), itostr(), RobSmtTimer::RobThread::m_cpiBase, RobSmtTimer::RobThread::m_cpiBranchPredictor, RobSmtTimer::RobThread::m_cpiDataCache, RobSmtTimer::RobThread::m_cpiIdle, RobSmtTimer::RobThread::m_cpiInstructionCache, RobSmtTimer::RobThread::m_cpiRSFull, RobSmtTimer::RobThread::m_cpiSerialization, RobSmtTimer::RobThread::m_cpiSMT, RobSmtTimer::RobThread::m_lastAccountedMemoryCycle, RobSmtTimer::RobThread::m_loads_count, RobSmtTimer::RobThread::m_loads_latency, m_mlp_histogram, RobSmtTimer::RobThread::m_outstandingLoads, RobSmtTimer::RobThread::m_outstandingLoadsAll, RobSmtTimer::RobThread::m_outstandingLongLatencyCycles, RobSmtTimer::RobThread::m_outstandingLongLatencyInsns, RobSmtTimer::RobThread::m_producerInsDistance, m_rob_threads, RobSmtTimer::RobThread::m_stores_count, RobSmtTimer::RobThread::m_stores_latency, SmtTimer::m_threads, RobSmtTimer::RobThread::m_totalConsumers, RobSmtTimer::RobThread::m_totalProducerInsDistance, RobSmtTimer::RobThread::m_uop_type_count, RobSmtTimer::RobThread::m_uops_pause, RobSmtTimer::RobThread::m_uops_total, RobSmtTimer::RobThread::m_uops_x87, MAX_OUTSTANDING, HitWhere::NUM_HITWHERES, registerStatsMetric(), MicroOp::UOP_SUBTYPE_SIZE, HitWhere::WHERE_FIRST, windowSize, and SubsecondTime::Zero().

6.281.4.13 issueInstruction()

```
void RobSmtTimer::issueInstruction (
              smtthread_id_t thread_num,
             uint64_t idx,
              SubsecondTime & next_event ) [private]
```

Definition at line 747 of file rob_smt_timer.cc.

References Core::accessMemory(), Memory::Access::address, RobSmtTimer::RobEntry::addressReady, RobSmtTimer::RobEntry::addressReadyMax, SmtTimer::SmtThread::core, SubsecondTime::divideRounded(), RobContention::doIssue(), RobSmtTimer::RobEntry::done, findEntryBySequenceNumber(), CircularQueue< T >::front(), RobSmtTimer::RobThread::frontend_stalled_until, Instruction::getAddress(), DynamicMicroOp::getAddress(), RobSmtTimer::RobEntry::getAddressProducer(), ContentionModel::getCompletionTime(), DynamicMicroOp::getDCacheHitWhere(), RobSmtTimer::RobEntry::getDependant(), DynamicMicroOp::getDependenciesLength(), ComponentTime::getElapsedTime(), DynamicMicroOp::getExecLatency(), MicroOp::getInstruction(), MicroOp::getMemoryAccessSize(), DynamicMicroOp::getMicroOp(), RobSmtTimer::RobEntry::getNumAddressProducers(), RobSmtTimer::RobEntry::getNumDependants(), ComponentTime::getPeriod(), SubsecondTime::getPS(), DynamicMicroOp::getSequenceNumber(), MemoryResult::hit_where, MicroOp::isBranch(), DynamicMicroOp::isBranchMispredicted(), MicroOp::isLoad(), MicroOp::isStore(), RobSmtTimer::RobEntry::issued, last_store_done, MemoryResult::latency, load_queue, LOG_ASSERT_ERROR, RobSmtTimer::RobThread::m_loads_count, RobSmtTimer::RobThread::m_loads_latency, m_rob_contention, m_rob_threads, m_rs_entries_used, RobSmtTimer::RobThread::m_stores_count, RobSmtTimer::RobThread::m_stores_latency, SmtTimer::m_threads, SubsecondTime::MaxTime(), Core::MEM_MODELED_RETURN, misprediction_penalty, Core::NONE, now, Core::READ, RobSmtTimer::RobEntry::ready, RobSmtTimer::RobEntry::readyMax, DynamicMicroOp::removeDependency(), RobSmtTimer::RobThread::rob, DynamicMicroOp::setDCacheHitWhere(), DynamicMicroOp::setExecLatency(), store_queue, MicroOp::toShortString(), HitWhere::UNKNOWN, RobSmtTimer::RobEntry::uop, and Core::WRITE.

Referenced by tryIssue().

6.281.4.14 notifyNumActiveThreadsChange()

```
void RobSmtTimer::notifyNumActiveThreadsChange ( )  [virtual]
```

Reimplemented from **SmtTimer** (p. 1259).

Definition at line 449 of file rob_smt_timer.cc.

References computeCurrentWindowsize().

6.281.4.15 printRob() [1/2]

```
void RobSmtTimer::printRob ( )  [private]
```

Definition at line 1298 of file rob_smt_timer.cc.

References ContentionModel::getNumUsed(), load_queue, m_rs_entries_used, SmtTimer::m_threads, now, and store_queue.

Referenced by executeCycle().

6.281.4.16 printRob() [2/2]

```
void RobSmtTimer::printRob (
            smtthread_id_t thread_num )  [private]
```

Definition at line 1235 of file rob_smt_timer.cc.

References Memory::Access::address, CircularQueue< T >::at(), RobSmtTimer::RobEntry::done, RobSmtTimer::RobThread::frontend_stalled_until, Instruction::getAddress(), DynamicMicroOp::getAddress(), DynamicMicroOp::getDependenciesLength(), DynamicMicroOp::getDependency(), Instruction::getDisassembly(), DynamicMicroOp::getExecLatency(), MicroOp::getInstruction(), DynamicMicroOp::getMicroOp(), DynamicMicroOp::getSequenceNumber(), SmtTimer::SmtThread::in_barrier, RobSmtTimer::RobThread::in_icache_miss, SmtTimer::SmtThread::in_wakeup, MicroOp::isLoad(), MicroOp::isStore(), RobSmtTimer::RobThread::m_num_in_rob, m_rob_threads, SmtTimer::m_threads, SubsecondTime::MaxTime(), RobSmtTimer::RobThread::next_event, now, RobSmtTimer::RobEntry::ready, RobSmtTimer::RobThread::rob, SmtTimer::SmtThread::running, CircularQueue< T >::size(), and RobSmtTimer::RobEntry::uop.

6.281.4.17 pushInstructions()

```
void RobSmtTimer::pushInstructions (
            smtthread_id_t thread_id,
          const std::vector<  DynamicMicroOp * > & insts )  [virtual]
```

Implements **SmtTimer** (p. 1259).

Definition at line 454 of file rob_smt_timer.cc.

References DynamicMicroOp::getMicroOp(), RobSmtTimer::RobEntry::init(), LOG_PRINT_WARNING_ONCE, m_rob_threads, RobSmtTimer::RobThread::m_uop_type_count, RobSmtTimer::RobThread::m_uops_pause, RobSmtTimer::RobThread::m_uops_total, RobSmtTimer::RobThread::m_uops_x87, CircularQueue< T >::next(), RobSmtTimer::RobThread::nextSequenceNumber, RobSmtTimer::RobThread::rob, MicroOp::toShortString(), and RobSmtTimer::RobEntry::uop.

Referenced by RobSmtPerformanceModel::simulate().

6.281.4.18 returnLatency()

```
boost::tuple< uint64_t, SubsecondTime > RobSmtTimer::returnLatency (
              smtthread_id_t thread_id ) [virtual]
```

Implements **SmtTimer** (p. 1260).

Definition at line 485 of file rob_smt_timer.cc.

References RobSmtTimer::RobThread::instrs, RobSmtTimer::RobThread::instrs_returned, m_rob_threads, RobSmtTimer::RobThread::now, now, and SubsecondTime::Zero().

Referenced by RobSmtPerformanceModel::simulate().

6.281.4.19 setDependencies()

```
void RobSmtTimer::setDependencies (
              smtthread_id_t thread_id,
              RobEntry * entry ) [private]
```

Definition at line 295 of file rob_smt_timer.cc.

References RobSmtTimer::RobEntry::addDependant(), DynamicMicroOp::addDependency(), RobSmtTimer::RobEntry::done, findEntryBySequenceNumber(), CircularQueue< T >::front(), RobSmtTimer::RobEntry::getAddressProducer(), RobSmtTimer::RobEntry::getDependant(), DynamicMicroOp::getDependenciesLength(), DynamicMicroOp::getDependency(), DynamicMicroOp::getMicroOp(), RobSmtTimer::RobEntry::getNumAddressProducers(), RobSmtTimer::RobEntry::getNumDependants(), DynamicMicroOp::getSequenceNumber(), MicroOp::isLoad(), MicroOp::isStore(), LOG_ASSERT_ERROR, RobSmtTimer::RobThread::m_producerInsDistance, m_rob_threads, m_store_to_load_forwarding, RobSmtTimer::RobThread::m_totalConsumers, RobSmtTimer::RobThread::m_totalProducerInsDistance, SubsecondTime::MaxTime(), RobSmtTimer::RobThread::memoryDependencies, RobSmtTimer::RobEntry::ready, RobSmtTimer::RobEntry::readyMax, RobSmtTimer::RobThread::registerDependencies, DynamicMicroOp::removeDependency(), RobSmtTimer::RobThread::rob, RegisterDependencies::setDependencies(), MemoryDependencies::setDependencies(), setStoreAddressProducers(), CircularQueue< T >::size(), and RobSmtTimer::RobEntry::uop.

Referenced by tryDispatch().

6.281.4.20 setStoreAddressProducers()

```
void RobSmtTimer::setStoreAddressProducers (
              smtthread_id_t thread_id,
              RobEntry * entry,
             uint64_t lowestValidSequenceNumber ) [private]
```

Definition at line 405 of file rob_smt_timer.cc.

References RobSmtTimer::RobEntry::addAddressProducer(), RobSmtTimer::RobEntry::addressReady, RobSmtTimer::RobEntry::addressReadyMax, RobSmtTimer::RobEntry::done, findEntryBySequenceNumber(), MicroOp::getAddressRegister(), DynamicMicroOp::getMicroOp(), RobSmtTimer::RobEntry::getNumAddressProducers(), INVALID_SEQNR, m_rob_threads, SubsecondTime::MaxTime(), RegisterDependencies::peekProducer(), RobSmtTimer::RobThread::registerDependencies, and RobSmtTimer::RobEntry::uop.

Referenced by setDependencies().

6.281.4.21 synchronize()

```
void RobSmtTimer::synchronize (
              smtthread_id_t thread_id,
              SubsecondTime time ) [virtual]
```

Implements **SmtTimer** (p. 1261).

Definition at line 520 of file rob_smt_timer.cc.

References ComponentTime::getElapsedTime(), m_rob_threads, SmtTimer::m_threads, SubsecondTime::Max↩ Time(), now, and ComponentTime::setElapsedTime().

Referenced by RobSmtPerformanceModel::notifyElapsedTimeUpdate().

6.281.4.22 threadHasEnoughInstructions()

```
bool RobSmtTimer::threadHasEnoughInstructions (
              smtthread_id_t thread_num ) [virtual]
```

Implements **SmtTimer** (p. 1261).

Definition at line 444 of file rob_smt_timer.cc.

References dispatchWidth, and threadNumSurplusInstructions().

6.281.4.23 threadNumSurplusInstructions()

```
uint64_t RobSmtTimer::threadNumSurplusInstructions (
              smtthread_id_t thread_num ) [virtual]
```

Implements **SmtTimer** (p. 1262).

Definition at line 438 of file rob_smt_timer.cc.

References RobSmtTimer::RobThread::m_num_in_rob, m_rob_threads, RobSmtTimer::RobThread::rob, and CircularQueue< T >::size().

Referenced by threadHasEnoughInstructions().

6.281.4.24 tryDispatch()

```
bool RobSmtTimer::tryDispatch (
              smtthread_id_t thread_num,
              SubsecondTime & next_event )  [private]
```

Definition at line 543 of file rob_smt_timer.cc.

References CircularQueue< T >::at(), currentWindowSize, RobSmtTimer::RobEntry::dispatched, dispatchWidth, RobSmtTimer::RobThread::frontend_stalled_until, DynamicMicroOp::getICacheHitWhere(), DynamicMicroOp::getICacheLatency(), DynamicMicroOp::getMicroOp(), RobSmtTimer::RobThread::in_icache_miss, MicroOp::isBranch(), DynamicMicroOp::isBranchMispredicted(), DynamicMicroOp::isLast(), HitWhere::L1I, LOG_ASSERT_ERROR, RobSmtTimer::RobThread::m_cpiBranchPredictor, RobSmtTimer::RobThread::m_cpiCurrentFrontEndStall, RobSmtTimer::RobThread::m_cpiInstructionCache, RobSmtTimer::RobThread::m_cpiRSFull, RobSmtTimer::RobThread::m_num_in_rob, m_rob_threads, m_rs_entries_used, SubsecondTime::MaxTime(), RobSmtTimer::RobThread::next_event, now, RobSmtTimer::RobEntry::ready, RobSmtTimer::RobThread::rob, rsEntries, setDependencies(), CircularQueue< T >::size(), MicroOp::toShortString(), RobSmtTimer::RobEntry::uop, and will_skip.

Referenced by doDispatch().

6.281.4.25 tryIssue()

```
bool RobSmtTimer::tryIssue (
              smtthread_id_t thread_num,
              SubsecondTime & next_event,
             bool would_have_skipped = false )  [private]
```

Definition at line 875 of file rob_smt_timer.cc.

References RobSmtTimer::RobEntry::addressReady, CircularQueue< T >::at(), dispatchWidth, RobSmtTimer::RobEntry::done, DynamicMicroOp::getDCacheHitWhere(), ComponentTime::getElapsedTime(), DynamicMicroOp::getMicroOp(), ContentionModel::hasFreeSlot(), inorder, MicroOp::isLoad(), DynamicMicroOp::isLongLatencyLoad(), MicroOp::isMemBarrier(), MicroOp::isSerializing(), MicroOp::isStore(), issueInstruction(), HitWhere::L1_OWN, last_store_done, load_queue, LOG_ASSERT_ERROR, RobSmtTimer::RobThread::m_lastAccountedMemoryCycle, m_no_address_disambiguation, RobSmtTimer::RobThread::m_num_in_rob, RobSmtTimer::RobThread::m_outstandingLongLatencyCycles, RobSmtTimer::RobThread::m_outstandingLongLatencyInsns, m_rob_contention, m_rob_threads, SubsecondTime::MaxTime(), RobSmtTimer::RobThread::next_event, RobContention::noMore(), now, RobSmtTimer::RobEntry::ready, RobSmtTimer::RobThread::rob, store_queue, RobContention::tryIssue(), RobSmtTimer::RobEntry::uop, and will_skip.

Referenced by doIssue().

6.281.5 Member Data Documentation

6.281.5.1 addressMask

```
int RobSmtTimer::addressMask  [private]
```

Definition at line 139 of file rob_smt_timer.h.

6.281.5.2 commitWidth

```
const uint64_t RobSmtTimer::commitWidth  [private]
```

Definition at line 113 of file rob_smt_timer.h.

Referenced by doCommit().

6.281.5.3 currentWindowSize

```
uint64_t RobSmtTimer::currentWindowSize  [private]
```

Definition at line 116 of file rob_smt_timer.h.

Referenced by canExecute(), computeCurrentWindowsize(), doDispatch(), and tryDispatch().

6.281.5.4 dispatch_thread

```
UInt8 RobSmtTimer::dispatch_thread  [private]
```

Definition at line 127 of file rob_smt_timer.h.

Referenced by doDispatch().

6.281.5.5 dispatchWidth

```
const uint64_t RobSmtTimer::dispatchWidth  [private]
```

Definition at line 112 of file rob_smt_timer.h.

Referenced by canExecute(), threadHasEnoughInstructions(), tryDispatch(), and tryIssue().

6.281.5.6 inorder

```
const bool RobSmtTimer::inorder  [private]
```

Definition at line 120 of file rob_smt_timer.h.

Referenced by tryIssue().

6.281.5.7 issue_thread

```
UInt8 RobSmtTimer::issue_thread [private]
```

Definition at line 128 of file rob_smt_timer.h.

Referenced by doIssue().

6.281.5.8 last_store_done

```
SubsecondTime RobSmtTimer::last_store_done [private]
```

Definition at line 131 of file rob_smt_timer.h.

Referenced by issueInstruction(), and tryIssue().

6.281.5.9 load_queue

```
ContentionModel RobSmtTimer::load_queue [private]
```

Definition at line 132 of file rob_smt_timer.h.

Referenced by issueInstruction(), printRob(), and tryIssue().

6.281.5.10 m_mlp_histogram

```
const bool RobSmtTimer::m_mlp_histogram [private]
```

Definition at line 164 of file rob_smt_timer.h.

Referenced by executeCycle(), and initializeThread().

6.281.5.11 m_no_address_disambiguation

```
const bool RobSmtTimer::m_no_address_disambiguation [private]
```

Definition at line 119 of file rob_smt_timer.h.

Referenced by tryIssue().

6.281.5.12 m_numBPredOverlapped

```
uint64_t RobSmtTimer::m_numBPredOverlapped  [private]
```

Definition at line 142 of file rob_smt_timer.h.

6.281.5.13 m_numDCacheOverlapped

```
uint64_t RobSmtTimer::m_numDCacheOverlapped  [private]
```

Definition at line 143 of file rob_smt_timer.h.

6.281.5.14 m_numHiddenLongerDCacheLatency

```
uint64_t RobSmtTimer::m_numHiddenLongerDCacheLatency  [private]
```

Definition at line 153 of file rob_smt_timer.h.

6.281.5.15 m_numICacheOverlapped

```
uint64_t RobSmtTimer::m_numICacheOverlapped  [private]
```

Definition at line 141 of file rob_smt_timer.h.

6.281.5.16 m_numLongLatencyLoads

```
uint64_t RobSmtTimer::m_numLongLatencyLoads  [private]
```

Definition at line 145 of file rob_smt_timer.h.

6.281.5.17 m_numMfenceInsns

```
uint64_t RobSmtTimer::m_numMfenceInsns  [private]
```

Definition at line 161 of file rob_smt_timer.h.

Referenced by RobSmtTimer().

6.281.5.18 m_numSerializationInsns

```
uint64_t RobSmtTimer::m_numSerializationInsns  [private]
```

Definition at line 148 of file rob_smt_timer.h.

Referenced by RobSmtTimer().

6.281.5.19 m_numTotalLongLatencyLoadLatency

```
uint64_t RobSmtTimer::m_numTotalLongLatencyLoadLatency  [private]
```

Definition at line 146 of file rob_smt_timer.h.

6.281.5.20 m_rob_contention

```
RobContention* RobSmtTimer::m_rob_contention  [private]
```

Definition at line 125 of file rob_smt_timer.h.

Referenced by doIssue(), issueInstruction(), and tryIssue().

6.281.5.21 m_rob_threads

```
std::vector< RobThread *> RobSmtTimer::m_rob_threads  [private]
```

Definition at line 124 of file rob_smt_timer.h.

Referenced by canExecute(), countOutstandingMemop(), doCommit(), doDispatch(), doIssue(), findCpi↩Component(), findEntryBySequenceNumber(), initializeThread(), issueInstruction(), printRob(), pushInstructions(), returnLatency(), setDependencies(), setStoreAddressProducers(), synchronize(), threadNumSurplusInstructions(), tryDispatch(), tryIssue(), and ∼RobSmtTimer().

6.281.5.22 m_rs_entries_used

```
uint64_t RobSmtTimer::m_rs_entries_used  [private]
```

Definition at line 134 of file rob_smt_timer.h.

Referenced by issueInstruction(), printRob(), and tryDispatch().

6.281.5.23 m_store_to_load_forwarding

```
const bool RobSmtTimer::m_store_to_load_forwarding  [private]
```

Definition at line 118 of file rob_smt_timer.h.

Referenced by setDependencies().

6.281.5.24 m_totalHiddenDCacheLatency

```
uint64_t RobSmtTimer::m_totalHiddenDCacheLatency  [private]
```

Definition at line 151 of file rob_smt_timer.h.

Referenced by RobSmtTimer().

6.281.5.25 m_totalHiddenLongerDCacheLatency

```
uint64_t RobSmtTimer::m_totalHiddenLongerDCacheLatency  [private]
```

Definition at line 152 of file rob_smt_timer.h.

6.281.5.26 m_totalMfenceLatency

```
uint64_t RobSmtTimer::m_totalMfenceLatency  [private]
```

Definition at line 162 of file rob_smt_timer.h.

Referenced by RobSmtTimer().

6.281.5.27 m_totalSerializationLatency

```
uint64_t RobSmtTimer::m_totalSerializationLatency  [private]
```

Definition at line 149 of file rob_smt_timer.h.

Referenced by RobSmtTimer().

6.281.5.28 MAX_OUTSTANDING

```
const unsigned int RobSmtTimer::MAX_OUTSTANDING = 32  [static], [private]
```

Definition at line 165 of file rob_smt_timer.h.

Referenced by countOutstandingMemop(), and initializeThread().

6.281.5.29 misprediction_penalty

```
const uint64_t RobSmtTimer::misprediction_penalty  [private]
```

Definition at line 117 of file rob_smt_timer.h.

Referenced by issueInstruction().

6.281.5.30 now

```
ComponentTime RobSmtTimer::now  [private]
```

Definition at line 130 of file rob_smt_timer.h.

Referenced by canExecute(), countOutstandingMemop(), doCommit(), doDispatch(), doIssue(), executeCycle(), findCpiComponent(), issueInstruction(), printRob(), returnLatency(), synchronize(), tryDispatch(), and tryIssue().

6.281.5.31 perf

```
PerformanceModel* RobSmtTimer::perf  [private]
```

Definition at line 155 of file rob_smt_timer.h.

6.281.5.32 rsEntries

```
const uint64_t RobSmtTimer::rsEntries  [private]
```

Definition at line 115 of file rob_smt_timer.h.

Referenced by tryDispatch().

6.281.5.33 simultaneousIssue

```
const bool RobSmtTimer::simultaneousIssue  [private]
```

Definition at line 122 of file rob_smt_timer.h.

Referenced by doIssue().

6.281.5.34 store_queue

```
ContentionModel RobSmtTimer::store_queue  [private]
```

Definition at line 133 of file rob_smt_timer.h.

Referenced by issueInstruction(), printRob(), and tryIssue().

6.281.5.35 time_skipped

```
SubsecondTime RobSmtTimer::time_skipped  [private]
```

Definition at line 137 of file rob_smt_timer.h.

Referenced by executeCycle(), and RobSmtTimer().

6.281.5.36 will_skip

```
bool RobSmtTimer::will_skip  [private]
```

Definition at line 136 of file rob_smt_timer.h.

Referenced by doCommit(), executeCycle(), tryDispatch(), and tryIssue().

6.281.5.37 windowRepartition

```
const bool RobSmtTimer::windowRepartition  [private]
```

Definition at line 121 of file rob_smt_timer.h.

Referenced by computeCurrentWindowsize().

6.281.5.38 windowSize

```
const uint64_t RobSmtTimer::windowSize  [private]
```

Definition at line 114 of file rob_smt_timer.h.

Referenced by computeCurrentWindowsize(), and initializeThread().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/rob_performance_model/ **rob_smt_timer.h**
- common/performance_model/performance_models/rob_performance_model/ **rob_smt_timer.cc**

6.282 RobSmtTimer::RobThread Class Reference

Public Member Functions

- **RobThread** (**Core** * **core**, int window_size)
- ~**RobThread** ()

Public Attributes

- **Core** * **core**
- **SubsecondTime** **now**
- uint64_t **instrs**
- uint64_t **instrs_returned**
- **Rob** **rob**
- uint64_t **m_num_in_rob**
- uint64_t **nextSequenceNumber**
- **SubsecondTime** **frontend_stalled_until**
- bool **in_icache_miss**
- **SubsecondTime** **next_event**
- **RegisterDependencies** *const **registerDependencies**
- **MemoryDependencies** *const **memoryDependencies**
- **SubsecondTime** * **m_cpiCurrentFrontEndStall**
- std::vector< std::vector< **SubsecondTime** > > **m_outstandingLoads**
- std::vector< **SubsecondTime** > **m_outstandingLoadsAll**
- **UInt64** **m_uop_type_count** [**MicroOp::UOP_SUBTYPE_SIZE**]
- **UInt64** **m_uops_total**
- **UInt64** **m_uops_x87**
- **UInt64** **m_uops_pause**
- **SubsecondTime** **m_cpiBase**
- **SubsecondTime** **m_cpiSMT**
- **SubsecondTime** **m_cpiIdle**
- **SubsecondTime** **m_cpiBranchPredictor**
- **SubsecondTime** **m_cpiSerialization**
- **SubsecondTime** **m_cpiRSFull**
- std::vector< **SubsecondTime** > **m_cpiInstructionCache**
- std::vector< **SubsecondTime** > **m_cpiDataCache**
- **SubsecondTime** **m_outstandingLongLatencyInsns**
- **SubsecondTime** **m_outstandingLongLatencyCycles**
- **SubsecondTime** **m_lastAccountedMemoryCycle**
- uint64_t **m_loads_count**
- **SubsecondTime** **m_loads_latency**
- uint64_t **m_stores_count**
- **SubsecondTime** **m_stores_latency**
- uint64_t **m_totalProducerInsDistance**
- uint64_t **m_totalConsumers**
- std::vector< uint64_t > **m_producerInsDistance**

6.282.1 Detailed Description

Definition at line 54 of file rob_smt_timer.h.

6.282.2 Constructor & Destructor Documentation

6.282.2.1 RobThread()

```
RobSmtTimer::RobThread::RobThread (
              Core * core,
             int window_size )
```

Definition at line 29 of file rob_smt_timer.cc.

6.282.2.2 ~RobThread()

```
RobSmtTimer::RobThread::~RobThread ( )
```

Definition at line 46 of file rob_smt_timer.cc.

6.282.3 Member Data Documentation

6.282.3.1 core

```
Core* RobSmtTimer::RobThread::core
```

Definition at line 56 of file rob_smt_timer.h.

Referenced by RobSmtTimer::doCommit().

6.282.3.2 frontend_stalled_until

```
SubsecondTime RobSmtTimer::RobThread::frontend_stalled_until
```

Definition at line 66 of file rob_smt_timer.h.

Referenced by RobSmtTimer::canExecute(), RobSmtTimer::doDispatch(), RobSmtTimer::issueInstruction(), Rob↩SmtTimer::printRob(), and RobSmtTimer::tryDispatch().

6.282.3.3 in_icache_miss

```
bool RobSmtTimer::RobThread::in_icache_miss
```

Definition at line 67 of file rob_smt_timer.h.

Referenced by RobSmtTimer::printRob(), and RobSmtTimer::tryDispatch().

6.282.3.4 instrs

```
uint64_t RobSmtTimer::RobThread::instrs
```

Definition at line 59 of file rob_smt_timer.h.

Referenced by RobSmtTimer::doCommit(), and RobSmtTimer::returnLatency().

6.282.3.5 instrs_returned

```
uint64_t RobSmtTimer::RobThread::instrs_returned
```

Definition at line 60 of file rob_smt_timer.h.

Referenced by RobSmtTimer::returnLatency().

6.282.3.6 m_cpiBase

SubsecondTime RobSmtTimer::RobThread::m_cpiBase

Definition at line 85 of file rob_smt_timer.h.

Referenced by RobSmtTimer::doDispatch(), and RobSmtTimer::initializeThread().

6.282.3.7 m_cpiBranchPredictor

SubsecondTime RobSmtTimer::RobThread::m_cpiBranchPredictor

Definition at line 88 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::tryDispatch().

6.282.3.8 m_cpiCurrentFrontEndStall

```
SubsecondTime* RobSmtTimer::RobThread::m_cpiCurrentFrontEndStall
```

Definition at line 74 of file rob_smt_timer.h.

Referenced by RobSmtTimer::doDispatch(), and RobSmtTimer::tryDispatch().

6.282.3.9 m_cpiDataCache

```
std::vector< SubsecondTime> RobSmtTimer::RobThread::m_cpiDataCache
```

Definition at line 93 of file rob_smt_timer.h.

Referenced by RobSmtTimer::findCpiComponent(), and RobSmtTimer::initializeThread().

6.282.3.10 m_cpiIdle

```
SubsecondTime RobSmtTimer::RobThread::m_cpiIdle
```

Definition at line 87 of file rob_smt_timer.h.

Referenced by RobSmtTimer::doDispatch(), and RobSmtTimer::initializeThread().

6.282.3.11 m_cpiInstructionCache

```
std::vector< SubsecondTime> RobSmtTimer::RobThread::m_cpiInstructionCache
```

Definition at line 92 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::tryDispatch().

6.282.3.12 m_cpiRSFull

```
SubsecondTime RobSmtTimer::RobThread::m_cpiRSFull
```

Definition at line 90 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::tryDispatch().

6.282.3.13 m_cpiSerialization

SubsecondTime RobSmtTimer::RobThread::m_cpiSerialization

Definition at line 89 of file rob_smt_timer.h.

Referenced by RobSmtTimer::findCpiComponent(), and RobSmtTimer::initializeThread().

6.282.3.14 m_cpiSMT

SubsecondTime RobSmtTimer::RobThread::m_cpiSMT

Definition at line 86 of file rob_smt_timer.h.

Referenced by RobSmtTimer::doDispatch(), and RobSmtTimer::initializeThread().

6.282.3.15 m_lastAccountedMemoryCycle

SubsecondTime RobSmtTimer::RobThread::m_lastAccountedMemoryCycle

Definition at line 97 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::tryIssue().

6.282.3.16 m_loads_count

uint64_t RobSmtTimer::RobThread::m_loads_count

Definition at line 99 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::issueInstruction().

6.282.3.17 m_loads_latency

SubsecondTime RobSmtTimer::RobThread::m_loads_latency

Definition at line 100 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::issueInstruction().

6.282.3.18 m_num_in_rob

```
uint64_t RobSmtTimer::RobThread::m_num_in_rob
```

Definition at line 63 of file rob_smt_timer.h.

Referenced by RobSmtTimer::canExecute(), RobSmtTimer::countOutstandingMemop(), RobSmtTimer::doCommit(), RobSmtTimer::doDispatch(), RobSmtTimer::findCpiComponent(), RobSmtTimer::printRob(), RobSmtTimer::threadNumSurplusInstructions(), RobSmtTimer::tryDispatch(), and RobSmtTimer::tryIssue().

6.282.3.19 m_outstandingLoads

```
std::vector<std::vector< SubsecondTime> > RobSmtTimer::RobThread::m_outstandingLoads
```

Definition at line 76 of file rob_smt_timer.h.

Referenced by RobSmtTimer::countOutstandingMemop(), and RobSmtTimer::initializeThread().

6.282.3.20 m_outstandingLoadsAll

```
std::vector< SubsecondTime> RobSmtTimer::RobThread::m_outstandingLoadsAll
```

Definition at line 77 of file rob_smt_timer.h.

Referenced by RobSmtTimer::countOutstandingMemop(), and RobSmtTimer::initializeThread().

6.282.3.21 m_outstandingLongLatencyCycles

```
SubsecondTime RobSmtTimer::RobThread::m_outstandingLongLatencyCycles
```

Definition at line 96 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::tryIssue().

6.282.3.22 m_outstandingLongLatencyInsns

```
SubsecondTime RobSmtTimer::RobThread::m_outstandingLongLatencyInsns
```

Definition at line 95 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::tryIssue().

6.282.3.23 m_producerInsDistance

```
std::vector<uint64_t> RobSmtTimer::RobThread::m_producerInsDistance
```

Definition at line 106 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::setDependencies().

6.282.3.24 m_stores_count

```
uint64_t RobSmtTimer::RobThread::m_stores_count
```

Definition at line 101 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::issueInstruction().

6.282.3.25 m_stores_latency

```
SubsecondTime RobSmtTimer::RobThread::m_stores_latency
```

Definition at line 102 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::issueInstruction().

6.282.3.26 m_totalConsumers

```
uint64_t RobSmtTimer::RobThread::m_totalConsumers
```

Definition at line 105 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::setDependencies().

6.282.3.27 m_totalProducerInsDistance

```
uint64_t RobSmtTimer::RobThread::m_totalProducerInsDistance
```

Definition at line 104 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::setDependencies().

6.282.3.28 m_uop_type_count

UInt64 RobSmtTimer::RobThread::m_uop_type_count[**MicroOp::UOP_SUBTYPE_SIZE**]

Definition at line 79 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::pushInstructions().

6.282.3.29 m_uops_pause

UInt64 RobSmtTimer::RobThread::m_uops_pause

Definition at line 82 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::pushInstructions().

6.282.3.30 m_uops_total

UInt64 RobSmtTimer::RobThread::m_uops_total

Definition at line 80 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::pushInstructions().

6.282.3.31 m_uops_x87

UInt64 RobSmtTimer::RobThread::m_uops_x87

Definition at line 81 of file rob_smt_timer.h.

Referenced by RobSmtTimer::initializeThread(), and RobSmtTimer::pushInstructions().

6.282.3.32 memoryDependencies

MemoryDependencies* const RobSmtTimer::RobThread::memoryDependencies

Definition at line 72 of file rob_smt_timer.h.

Referenced by RobSmtTimer::setDependencies().

6.282.3.33 next_event

SubsecondTime RobSmtTimer::RobThread::next_event

Definition at line 69 of file rob_smt_timer.h.

Referenced by RobSmtTimer::doIssue(), RobSmtTimer::printRob(), RobSmtTimer::tryDispatch(), and RobSmtTimer::tryIssue().

6.282.3.34 nextSequenceNumber

uint64_t RobSmtTimer::RobThread::nextSequenceNumber

Definition at line 64 of file rob_smt_timer.h.

Referenced by RobSmtTimer::pushInstructions().

6.282.3.35 now

SubsecondTime RobSmtTimer::RobThread::now

Definition at line 58 of file rob_smt_timer.h.

Referenced by RobSmtTimer::canExecute(), RobSmtTimer::doDispatch(), and RobSmtTimer::returnLatency().

6.282.3.36 registerDependencies

RegisterDependencies* const RobSmtTimer::RobThread::registerDependencies

Definition at line 71 of file rob_smt_timer.h.

Referenced by RobSmtTimer::setDependencies(), and RobSmtTimer::setStoreAddressProducers().

6.282.3.37 rob

Rob RobSmtTimer::RobThread::rob

Definition at line 62 of file rob_smt_timer.h.

Referenced by RobSmtTimer::canExecute(), RobSmtTimer::countOutstandingMemop(), RobSmtTimer::doCommit(), RobSmtTimer::findCpiComponent(), RobSmtTimer::issueInstruction(), RobSmtTimer::printRob(), RobSmtTimer::pushInstructions(), RobSmtTimer::setDependencies(), RobSmtTimer::threadNumSurplusInstructions(), RobSmtTimer::tryDispatch(), and RobSmtTimer::tryIssue().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/rob_performance_model/ **rob_smt_timer.h**
- common/performance_model/performance_models/rob_performance_model/ **rob_smt_timer.cc**

6.283 RobTimer Class Reference

`#include <rob_timer.h>`

Classes

- class **RobEntry**

Public Member Functions

- **RobTimer** (**Core** *core, **PerformanceModel** * **perf**, const **CoreModel** *core_model, int **misprediction_penalty**, int dispatch_width, int window_size)
- ~**RobTimer** ()
- boost::tuple< uint64_t, **SubsecondTime** > **simulate** (const std::vector< **DynamicMicroOp** * > &insts)
- void **synchronize** (**SubsecondTime** time)

Private Types

- typedef **CircularQueue**< **RobEntry** > **Rob**

Private Member Functions

- **RobEntry** * **findEntryBySequenceNumber** (**UInt64** sequenceNumber)
- **SubsecondTime** * **findCpiComponent** ()
- void **countOutstandingMemop** (**SubsecondTime** time)
- void **printRob** ()
- void **execute** (uint64_t &instructionsExecuted, **SubsecondTime** &latency)
- **SubsecondTime** **doDispatch** (**SubsecondTime** **cpiComponent)
- **SubsecondTime** **doIssue** ()
- **SubsecondTime** **doCommit** (uint64_t &instructionsExecuted)
- void **issueInstruction** (uint64_t idx, **SubsecondTime** &next_event)

Private Attributes

- const uint64_t **dispatchWidth**
- const uint64_t **commitWidth**
- const uint64_t **windowSize**
- const uint64_t **rsEntries**
- const uint64_t **misprediction_penalty**
- const bool **m_store_to_load_forwarding**
- const bool **m_no_address_disambiguation**
- const bool **inorder**
- **Core** * **m_core**
- **Rob** **rob**
- uint64_t **m_num_in_rob**
- uint64_t **m_rs_entries_used**
- **RobContention** * **m_rob_contention**
- **ComponentTime** **now**
- **SubsecondTime** **frontend_stalled_until**

- bool **in_icache_miss**
- **SubsecondTime last_store_done**
- **ContentionModel load_queue**
- **ContentionModel store_queue**
- uint64_t **nextSequenceNumber**
- bool **will_skip**
- **SubsecondTime time_skipped**
- **RegisterDependencies** ∗const **registerDependencies**
- **MemoryDependencies** ∗const **memoryDependencies**
- int **addressMask**
- **UInt64 m_uop_type_count** [**MicroOp::UOP_SUBTYPE_SIZE**]
- **UInt64 m_uops_total**
- **UInt64 m_uops_x87**
- **UInt64 m_uops_pause**
- uint64_t **m_numICacheOverlapped**
- uint64_t **m_numBPredOverlapped**
- uint64_t **m_numDCacheOverlapped**
- uint64_t **m_numLongLatencyLoads**
- uint64_t **m_numTotalLongLatencyLoadLatency**
- uint64_t **m_numSerializationInsns**
- uint64_t **m_totalSerializationLatency**
- uint64_t **m_totalHiddenDCacheLatency**
- uint64_t **m_totalHiddenLongerDCacheLatency**
- uint64_t **m_numHiddenLongerDCacheLatency**
- **SubsecondTime m_outstandingLongLatencyInsns**
- **SubsecondTime m_outstandingLongLatencyCycles**
- **SubsecondTime m_lastAccountedMemoryCycle**
- uint64_t **m_loads_count**
- **SubsecondTime m_loads_latency**
- uint64_t **m_stores_count**
- **SubsecondTime m_stores_latency**
- uint64_t **m_totalProducerInsDistance**
- uint64_t **m_totalConsumers**
- std::vector< uint64_t > **m_producerInsDistance**
- **PerformanceModel** ∗ **perf**
- uint64_t **m_numMfenceInsns**
- uint64_t **m_totalMfenceLatency**
- **SubsecondTime m_cpiBase**
- **SubsecondTime m_cpiBranchPredictor**
- **SubsecondTime m_cpiSerialization**
- **SubsecondTime m_cpiRSFull**
- std::vector< **SubsecondTime** > **m_cpiInstructionCache**
- std::vector< **SubsecondTime** > **m_cpiDataCache**
- **SubsecondTime** ∗ **m_cpiCurrentFrontEndStall**
- const bool **m_mlp_histogram**
- std::vector< std::vector< **SubsecondTime** > > **m_outstandingLoads**
- std::vector< **SubsecondTime** > **m_outstandingLoadsAll**

Static Private Attributes

- static const unsigned int **MAX_OUTSTANDING** = 32

6.283.1 Detailed Description

Definition at line 14 of file rob_timer.h.

6.283.2 Member Typedef Documentation

6.283.2.1 Rob

```
typedef CircularQueue< RobEntry> RobTimer::Rob [private]
```

Definition at line 59 of file rob_timer.h.

6.283.3 Constructor & Destructor Documentation

6.283.3.1 RobTimer()

```
RobTimer::RobTimer (
            Core * core,
            PerformanceModel * perf,
          const  CoreModel * core_model,
          int misprediction_penalty,
          int dispatch_width,
          int window_size )
```

Definition at line 27 of file rob_timer.cc.

References Core::getId(), MicroOp::getSubtypeString(), HitWhereIsValid(), HitWhereString(), itostr(), m_cpiBase, m_cpiBranchPredictor, m_cpiDataCache, m_cpiInstructionCache, m_cpiRSFull, m_cpiSerialization, m_lastAccountedMemoryCycle, m_loads_count, m_loads_latency, m_mlp_histogram, m_numMfenceInsns, m_numSerializationInsns, m_outstandingLoads, m_outstandingLoadsAll, m_outstandingLongLatencyCycles, m_outstandingLongLatencyInsns, m_producerInsDistance, m_stores_count, m_stores_latency, m_totalConsumers, m_totalHiddenDCacheLatency, m_totalMfenceLatency, m_totalProducerInsDistance, m_totalSerializationLatency, m_uop_type_count, m_uops_pause, m_uops_total, m_uops_x87, MAX_OUTSTANDING, HitWhere::NUM_HITWHERES, registerStatsMetric(), time_skipped, MicroOp::UOP_SUBTYPE_SIZE, HitWhere::WHERE_FIRST, windowSize, and SubsecondTime::Zero().

6.283.3.2 ~RobTimer()

```
RobTimer::~RobTimer ( )
```

Definition at line 176 of file rob_timer.cc.

References CircularQueue< T >::begin(), and rob.

6.283.4 Member Function Documentation

6.283.4.1 countOutstandingMemop()

```
void RobTimer::countOutstandingMemop (
            SubsecondTime time )  [private]
```

Definition at line 955 of file rob_timer.cc.

References CircularQueue< T >::at(), RobTimer::RobEntry::done, DynamicMicroOp::getDCacheHitWhere(), DynamicMicroOp::getMicroOp(), MicroOp::isLoad(), m_num_in_rob, m_outstandingLoads, m_outstandingLoads↩All, MAX_OUTSTANDING, SubsecondTime::MaxTime(), now, HitWhere::NUM_HITWHERES, rob, and Rob↩Timer::RobEntry::uop.

Referenced by execute().

6.283.4.2 doCommit()

```
SubsecondTime RobTimer::doCommit (
            uint64_t & instructionsExecuted )  [private]
```

Definition at line 831 of file rob_timer.cc.

References commitWidth, RobTimer::RobEntry::dispatched, RobTimer::RobEntry::done, RobTimer::RobEntry↩::free(), CircularQueue< T >::front(), DynamicMicroOp::getMicroOp(), Core::getPerformanceModel(), Dynamic↩MicroOp::isLast(), RobTimer::RobEntry::issued, LOG_ASSERT_ERROR, m_core, m_num_in_rob, Subsecond↩Time::MaxTime(), now, CircularQueue< T >::pop(), rob, CircularQueue< T >::size(), MicroOp::toShortString(), PerformanceModel::traceInstruction(), RobTimer::RobEntry::uop, and will_skip.

Referenced by execute().

6.283.4.3 doDispatch()

```
SubsecondTime RobTimer::doDispatch (
            SubsecondTime ** cpiComponent )  [private]
```

Definition at line 438 of file rob_timer.cc.

References CircularQueue< T >::at(), RobTimer::RobEntry::dispatched, dispatchWidth, findCpiComponent(), frontend_stalled_until, DynamicMicroOp::getICacheHitWhere(), DynamicMicroOp::getICacheLatency(), Dynamic↩MicroOp::getMicroOp(), in_icache_miss, MicroOp::isBranch(), DynamicMicroOp::isBranchMispredicted(), DynamicMicroOp::isLast(), HitWhere::L1I, LOG_ASSERT_ERROR, m_cpiBase, m_cpiBranchPredictor, m_↩cpiCurrentFrontEndStall, m_cpiInstructionCache, m_cpiRSFull, m_num_in_rob, m_rs_entries_used, Subsecond↩Time::MaxTime(), now, RobTimer::RobEntry::ready, rob, rsEntries, CircularQueue< T >::size(), MicroOp::to↩ShortString(), RobTimer::RobEntry::uop, will_skip, and windowSize.

Referenced by execute().

6.283.4.4 doIssue()

```
SubsecondTime RobTimer::doIssue ( )  [private]
```

Definition at line 697 of file rob_timer.cc.

References RobTimer::RobEntry::addressReady, CircularQueue< T >::at(), dispatchWidth, RobTimer::RobEntry::done, DynamicMicroOp::getDCacheHitWhere(), ComponentTime::getElapsedTime(), DynamicMicroOp::getMicroOp(), ContentionModel::hasFreeSlot(), RobContention::initCycle(), inorder, MicroOp::isLoad(), DynamicMicroOp::isLongLatencyLoad(), MicroOp::isMemBarrier(), MicroOp::isSerializing(), MicroOp::isStore(), issueInstruction(), HitWhere::L1_OWN, last_store_done, load_queue, LOG_ASSERT_ERROR, m_lastAccountedMemoryCycle, m_no_address_disambiguation, m_num_in_rob, m_outstandingLongLatencyCycles, m_outstandingLongLatencyInsns, m_rob_contention, SubsecondTime::MaxTime(), RobContention::noMore(), now, RobTimer::RobEntry::ready, rob, store_queue, RobContention::tryIssue(), RobTimer::RobEntry::uop, and will_skip.

Referenced by execute().

6.283.4.5 execute()

```
void RobTimer::execute (
             uint64_t & instructionsExecuted,
              SubsecondTime & latency )  [private]
```

Definition at line 874 of file rob_timer.cc.

References countOutstandingMemop(), dispatchWidth, doCommit(), doDispatch(), doIssue(), frontend_stalled_until, ComponentTime::getPeriod(), LOG_ASSERT_ERROR, m_mlp_histogram, m_num_in_rob, SubsecondTime::MaxTime(), now, printRob(), rob, CircularQueue< T >::size(), time_skipped, will_skip, and SubsecondTime::Zero().

Referenced by simulate().

6.283.4.6 findCpiComponent()

```
SubsecondTime * RobTimer::findCpiComponent ( )  [private]
```

Definition at line 414 of file rob_timer.cc.

References CircularQueue< T >::at(), RobTimer::RobEntry::done, DynamicMicroOp::getDCacheHitWhere(), DynamicMicroOp::getMicroOp(), MicroOp::isLoad(), MicroOp::isMemBarrier(), MicroOp::isSerializing(), MicroOp::isStore(), m_cpiDataCache, m_cpiSerialization, m_num_in_rob, now, rob, and RobTimer::RobEntry::uop.

Referenced by doDispatch().

6.283.4.7 findEntryBySequenceNumber()

```
RobTimer::RobEntry * RobTimer::findEntryBySequenceNumber (
            UInt64 sequenceNumber ) [private]
```

Definition at line 242 of file rob_timer.cc.

References DynamicMicroOp::getSequenceNumber(), LOG_ASSERT_ERROR, rob, CircularQueue< T >::size(), and RobTimer::RobEntry::uop.

Referenced by issueInstruction(), and simulate().

6.283.4.8 issueInstruction()

```
void RobTimer::issueInstruction (
            uint64_t idx,
            SubsecondTime & next_event ) [private]
```

Definition at line 567 of file rob_timer.cc.

References Core::accessMemory(), Memory::Access::address, RobTimer::RobEntry::addressReady, RobTimer::RobEntry::addressReadyMax, SubsecondTime::divideRounded(), RobContention::doIssue(), RobTimer::RobEntry::done, findEntryBySequenceNumber(), CircularQueue< T >::front(), frontend_stalled_until, Instruction::getAddress(), DynamicMicroOp::getAddress(), RobTimer::RobEntry::getAddressProducer(), ContentionModel::getCompletionTime(), DynamicMicroOp::getDCacheHitWhere(), RobTimer::RobEntry::getDependant(), DynamicMicroOp::getDependenciesLength(), ComponentTime::getElapsedTime(), DynamicMicroOp::getExecLatency(), MicroOp::getInstruction(), MicroOp::getMemoryAccessSize(), DynamicMicroOp::getMicroOp(), RobTimer::RobEntry::getNumAddressProducers(), RobTimer::RobEntry::getNumDependants(), ComponentTime::getPeriod(), SubsecondTime::getPS(), DynamicMicroOp::getSequenceNumber(), MemoryResult::hit_where, MicroOp::isBranch(), DynamicMicroOp::isBranchMispredicted(), MicroOp::isLoad(), MicroOp::isStore(), RobTimer::RobEntry::issued, last_store_done, MemoryResult::latency, load_queue, LOG_ASSERT_ERROR, m_core, m_loads_count, m_loads_latency, m_rob_contention, m_rs_entries_used, m_stores_count, m_stores_latency, SubsecondTime::MaxTime(), Core::MEM_MODELED_RETURN, misprediction_penalty, Core::NONE, now, Core::READ, RobTimer::RobEntry::ready, RobTimer::RobEntry::readyMax, DynamicMicroOp::removeDependency(), rob, DynamicMicroOp::setDCacheHitWhere(), DynamicMicroOp::setExecLatency(), store_queue, MicroOp::toShortString(), HitWhere::UNKNOWN, RobTimer::RobEntry::uop, and Core::WRITE.

Referenced by doIssue().

6.283.4.9 printRob()

```
void RobTimer::printRob ( ) [private]
```

Definition at line 976 of file rob_timer.cc.

References Memory::Access::address, CircularQueue< T >::at(), RobTimer::RobEntry::done, frontend_stalled_until, Instruction::getAddress(), DynamicMicroOp::getAddress(), DynamicMicroOp::getDependenciesLength(), DynamicMicroOp::getDependency(), Instruction::getDisassembly(), DynamicMicroOp::getExecLatency(), MicroOp::getInstruction(), DynamicMicroOp::getMicroOp(), ContentionModel::getNumUsed(), DynamicMicroOp::getSequenceNumber(), in_icache_miss, MicroOp::isLoad(), MicroOp::isStore(), load_queue, m_num_in_rob, m_rs_entries_used, SubsecondTime::MaxTime(), now, RobTimer::RobEntry::ready, rob, CircularQueue< T >::size(), store_queue, and RobTimer::RobEntry::uop.

Referenced by execute().

6.283.4.10 simulate()

```
boost::tuple< uint64_t, SubsecondTime > RobTimer::simulate (
            const std::vector< DynamicMicroOp * > & insts )
```

Definition at line 253 of file rob_timer.cc.

References RobTimer::RobEntry::addAddressProducer(), RobTimer::RobEntry::addDependant(), DynamicMicroOp::addDependency(), RobTimer::RobEntry::addressReady, RobTimer::RobEntry::addressReadyMax, RobTimer::RobEntry::done, execute(), findEntryBySequenceNumber(), CircularQueue< T >::front(), RobTimer::RobEntry::getAddressProducer(), MicroOp::getAddressRegister(), RobTimer::RobEntry::getDependant(), DynamicMicroOp::getDependenciesLength(), DynamicMicroOp::getDependency(), DynamicMicroOp::getMicroOp(), RobTimer::RobEntry::getNumAddressProducers(), RobTimer::RobEntry::getNumDependants(), DynamicMicroOp::getSequenceNumber(), RobTimer::RobEntry::init(), INVALID_SEQNR, MicroOp::isLoad(), MicroOp::isStore(), LOG_ASSERT_ERROR, LOG_PRINT_WARNING_ONCE, m_producerInsDistance, m_store_to_load_forwarding, m_totalConsumers, m_totalProducerInsDistance, m_uop_type_count, m_uops_pause, m_uops_total, m_uops_x87, SubsecondTime::MaxTime(), memoryDependencies, CircularQueue< T >::next(), nextSequenceNumber, RegisterDependencies::peekProducer(), RobTimer::RobEntry::ready, RobTimer::RobEntry::readyMax, registerDependencies, DynamicMicroOp::removeDependency(), rob, RegisterDependencies::setDependencies(), MemoryDependencies::setDependencies(), CircularQueue< T >::size(), MicroOp::toShortString(), RobTimer::RobEntry::uop, and SubsecondTime::Zero().

Referenced by RobPerformanceModel::simulate().

6.283.4.11 synchronize()

```
void RobTimer::synchronize (
              SubsecondTime time )
```

Definition at line 406 of file rob_timer.cc.

References now, and ComponentTime::setElapsedTime().

Referenced by RobPerformanceModel::notifyElapsedTimeUpdate().

6.283.5 Member Data Documentation

6.283.5.1 addressMask

```
int RobTimer::addressMask  [private]
```

Definition at line 79 of file rob_timer.h.

6.283.5.2 commitWidth

```
const uint64_t RobTimer::commitWidth  [private]
```

Definition at line 49 of file rob_timer.h.

Referenced by doCommit().

6.283.5.3 dispatchWidth

```
const uint64_t RobTimer::dispatchWidth  [private]
```

Definition at line 48 of file rob_timer.h.

Referenced by doDispatch(), doIssue(), and execute().

6.283.5.4 frontend_stalled_until

```
SubsecondTime RobTimer::frontend_stalled_until  [private]
```

Definition at line 66 of file rob_timer.h.

Referenced by doDispatch(), execute(), issueInstruction(), and printRob().

6.283.5.5 in_icache_miss

```
bool RobTimer::in_icache_miss  [private]
```

Definition at line 67 of file rob_timer.h.

Referenced by doDispatch(), and printRob().

6.283.5.6 inorder

```
const bool RobTimer::inorder  [private]
```

Definition at line 55 of file rob_timer.h.

Referenced by doIssue().

6.283.5.7 last_store_done

SubsecondTime RobTimer::last_store_done [private]

Definition at line 68 of file rob_timer.h.

Referenced by doIssue(), and issueInstruction().

6.283.5.8 load_queue

ContentionModel RobTimer::load_queue [private]

Definition at line 69 of file rob_timer.h.

Referenced by doIssue(), issueInstruction(), and printRob().

6.283.5.9 m_core

Core* RobTimer::m_core [private]

Definition at line 57 of file rob_timer.h.

Referenced by doCommit(), and issueInstruction().

6.283.5.10 m_cpiBase

SubsecondTime RobTimer::m_cpiBase [private]

Definition at line 123 of file rob_timer.h.

Referenced by doDispatch(), and RobTimer().

6.283.5.11 m_cpiBranchPredictor

SubsecondTime RobTimer::m_cpiBranchPredictor [private]

Definition at line 124 of file rob_timer.h.

Referenced by doDispatch(), and RobTimer().

6.283.5.12 m_cpiCurrentFrontEndStall

```
SubsecondTime* RobTimer::m_cpiCurrentFrontEndStall [private]
```

Definition at line 131 of file rob_timer.h.

Referenced by doDispatch().

6.283.5.13 m_cpiDataCache

```
std::vector< SubsecondTime> RobTimer::m_cpiDataCache [private]
```

Definition at line 129 of file rob_timer.h.

Referenced by findCpiComponent(), and RobTimer().

6.283.5.14 m_cpiInstructionCache

```
std::vector< SubsecondTime> RobTimer::m_cpiInstructionCache [private]
```

Definition at line 128 of file rob_timer.h.

Referenced by doDispatch(), and RobTimer().

6.283.5.15 m_cpiRSFull

```
SubsecondTime RobTimer::m_cpiRSFull [private]
```

Definition at line 126 of file rob_timer.h.

Referenced by doDispatch(), and RobTimer().

6.283.5.16 m_cpiSerialization

```
SubsecondTime RobTimer::m_cpiSerialization [private]
```

Definition at line 125 of file rob_timer.h.

Referenced by findCpiComponent(), and RobTimer().

6.283.5.17 m_lastAccountedMemoryCycle

SubsecondTime RobTimer::m_lastAccountedMemoryCycle [private]

Definition at line 102 of file rob_timer.h.

Referenced by doIssue(), and RobTimer().

6.283.5.18 m_loads_count

uint64_t RobTimer::m_loads_count [private]

Definition at line 104 of file rob_timer.h.

Referenced by issueInstruction(), and RobTimer().

6.283.5.19 m_loads_latency

SubsecondTime RobTimer::m_loads_latency [private]

Definition at line 105 of file rob_timer.h.

Referenced by issueInstruction(), and RobTimer().

6.283.5.20 m_mlp_histogram

const bool RobTimer::m_mlp_histogram [private]

Definition at line 133 of file rob_timer.h.

Referenced by execute(), and RobTimer().

6.283.5.21 m_no_address_disambiguation

const bool RobTimer::m_no_address_disambiguation [private]

Definition at line 54 of file rob_timer.h.

Referenced by doIssue().

6.283.5.22 m_num_in_rob

```
uint64_t RobTimer::m_num_in_rob  [private]
```

Definition at line 61 of file rob_timer.h.

Referenced by countOutstandingMemop(), doCommit(), doDispatch(), doIssue(), execute(), findCpiComponent(), and printRob().

6.283.5.23 m_numBPredOverlapped

```
uint64_t RobTimer::m_numBPredOverlapped  [private]
```

Definition at line 87 of file rob_timer.h.

6.283.5.24 m_numDCacheOverlapped

```
uint64_t RobTimer::m_numDCacheOverlapped  [private]
```

Definition at line 88 of file rob_timer.h.

6.283.5.25 m_numHiddenLongerDCacheLatency

```
uint64_t RobTimer::m_numHiddenLongerDCacheLatency  [private]
```

Definition at line 98 of file rob_timer.h.

6.283.5.26 m_numICacheOverlapped

```
uint64_t RobTimer::m_numICacheOverlapped  [private]
```

Definition at line 86 of file rob_timer.h.

6.283.5.27 m_numLongLatencyLoads

```
uint64_t RobTimer::m_numLongLatencyLoads  [private]
```

Definition at line 90 of file rob_timer.h.

6.283.5.28 m_numMfenceInsns

```
uint64_t RobTimer::m_numMfenceInsns  [private]
```

Definition at line 119 of file rob_timer.h.

Referenced by RobTimer().

6.283.5.29 m_numSerializationInsns

```
uint64_t RobTimer::m_numSerializationInsns  [private]
```

Definition at line 93 of file rob_timer.h.

Referenced by RobTimer().

6.283.5.30 m_numTotalLongLatencyLoadLatency

```
uint64_t RobTimer::m_numTotalLongLatencyLoadLatency  [private]
```

Definition at line 91 of file rob_timer.h.

6.283.5.31 m_outstandingLoads

```
std::vector<std::vector< SubsecondTime> > RobTimer::m_outstandingLoads  [private]
```

Definition at line 135 of file rob_timer.h.

Referenced by countOutstandingMemop(), and RobTimer().

6.283.5.32 m_outstandingLoadsAll

```
std::vector< SubsecondTime> RobTimer::m_outstandingLoadsAll  [private]
```

Definition at line 136 of file rob_timer.h.

Referenced by countOutstandingMemop(), and RobTimer().

6.283.5.33 m_outstandingLongLatencyCycles

SubsecondTime RobTimer::m_outstandingLongLatencyCycles [private]

Definition at line 101 of file rob_timer.h.

Referenced by doIssue(), and RobTimer().

6.283.5.34 m_outstandingLongLatencyInsns

SubsecondTime RobTimer::m_outstandingLongLatencyInsns [private]

Definition at line 100 of file rob_timer.h.

Referenced by doIssue(), and RobTimer().

6.283.5.35 m_producerInsDistance

std::vector<uint64_t> RobTimer::m_producerInsDistance [private]

Definition at line 111 of file rob_timer.h.

Referenced by RobTimer(), and simulate().

6.283.5.36 m_rob_contention

RobContention* RobTimer::m_rob_contention [private]

Definition at line 63 of file rob_timer.h.

Referenced by doIssue(), and issueInstruction().

6.283.5.37 m_rs_entries_used

uint64_t RobTimer::m_rs_entries_used [private]

Definition at line 62 of file rob_timer.h.

Referenced by doDispatch(), issueInstruction(), and printRob().

6.283.5.38 m_store_to_load_forwarding

```
const bool RobTimer::m_store_to_load_forwarding  [private]
```

Definition at line 53 of file rob_timer.h.

Referenced by simulate().

6.283.5.39 m_stores_count

```
uint64_t RobTimer::m_stores_count  [private]
```

Definition at line 106 of file rob_timer.h.

Referenced by issueInstruction(), and RobTimer().

6.283.5.40 m_stores_latency

```
SubsecondTime RobTimer::m_stores_latency  [private]
```

Definition at line 107 of file rob_timer.h.

Referenced by issueInstruction(), and RobTimer().

6.283.5.41 m_totalConsumers

```
uint64_t RobTimer::m_totalConsumers  [private]
```

Definition at line 110 of file rob_timer.h.

Referenced by RobTimer(), and simulate().

6.283.5.42 m_totalHiddenDCacheLatency

```
uint64_t RobTimer::m_totalHiddenDCacheLatency  [private]
```

Definition at line 96 of file rob_timer.h.

Referenced by RobTimer().

6.283.5.43 m_totalHiddenLongerDCacheLatency

```
uint64_t RobTimer::m_totalHiddenLongerDCacheLatency  [private]
```

Definition at line 97 of file rob_timer.h.

6.283.5.44 m_totalMfenceLatency

```
uint64_t RobTimer::m_totalMfenceLatency  [private]
```

Definition at line 120 of file rob_timer.h.

Referenced by RobTimer().

6.283.5.45 m_totalProducerInsDistance

```
uint64_t RobTimer::m_totalProducerInsDistance  [private]
```

Definition at line 109 of file rob_timer.h.

Referenced by RobTimer(), and simulate().

6.283.5.46 m_totalSerializationLatency

```
uint64_t RobTimer::m_totalSerializationLatency  [private]
```

Definition at line 94 of file rob_timer.h.

Referenced by RobTimer().

6.283.5.47 m_uop_type_count

```
UInt64 RobTimer::m_uop_type_count[ MicroOp::UOP_SUBTYPE_SIZE]  [private]
```

Definition at line 81 of file rob_timer.h.

Referenced by RobTimer(), and simulate().

6.283.5.48 m_uops_pause

UInt64 RobTimer::m_uops_pause [private]

Definition at line 84 of file rob_timer.h.

Referenced by RobTimer(), and simulate().

6.283.5.49 m_uops_total

UInt64 RobTimer::m_uops_total [private]

Definition at line 82 of file rob_timer.h.

Referenced by RobTimer(), and simulate().

6.283.5.50 m_uops_x87

UInt64 RobTimer::m_uops_x87 [private]

Definition at line 83 of file rob_timer.h.

Referenced by RobTimer(), and simulate().

6.283.5.51 MAX_OUTSTANDING

const unsigned int RobTimer::MAX_OUTSTANDING = 32 [static], [private]

Definition at line 134 of file rob_timer.h.

Referenced by countOutstandingMemop(), and RobTimer().

6.283.5.52 memoryDependencies

MemoryDependencies* const RobTimer::memoryDependencies [private]

Definition at line 77 of file rob_timer.h.

Referenced by simulate().

6.283.5.53 mispredict_penalty

```
const uint64_t RobTimer::mispredict_penalty  [private]
```

Definition at line 52 of file rob_timer.h.

Referenced by issueInstruction().

6.283.5.54 nextSequenceNumber

```
uint64_t RobTimer::nextSequenceNumber  [private]
```

Definition at line 72 of file rob_timer.h.

Referenced by simulate().

6.283.5.55 now

```
ComponentTime RobTimer::now  [private]
```

Definition at line 65 of file rob_timer.h.

Referenced by countOutstandingMemop(), doCommit(), doDispatch(), doIssue(), execute(), findCpiComponent(), issueInstruction(), printRob(), and synchronize().

6.283.5.56 perf

```
PerformanceModel* RobTimer::perf  [private]
```

Definition at line 113 of file rob_timer.h.

6.283.5.57 registerDependencies

```
RegisterDependencies* const RobTimer::registerDependencies  [private]
```

Definition at line 76 of file rob_timer.h.

Referenced by simulate().

6.283.5.58 rob

Rob RobTimer::rob [private]

Definition at line 60 of file rob_timer.h.

Referenced by countOutstandingMemop(), doCommit(), doDispatch(), doIssue(), execute(), findCpiComponent(), findEntryBySequenceNumber(), issueInstruction(), printRob(), simulate(), and ~RobTimer().

6.283.5.59 rsEntries

const uint64_t RobTimer::rsEntries [private]

Definition at line 51 of file rob_timer.h.

Referenced by doDispatch().

6.283.5.60 store_queue

ContentionModel RobTimer::store_queue [private]

Definition at line 70 of file rob_timer.h.

Referenced by doIssue(), issueInstruction(), and printRob().

6.283.5.61 time_skipped

SubsecondTime RobTimer::time_skipped [private]

Definition at line 74 of file rob_timer.h.

Referenced by execute(), and RobTimer().

6.283.5.62 will_skip

bool RobTimer::will_skip [private]

Definition at line 73 of file rob_timer.h.

Referenced by doCommit(), doDispatch(), doIssue(), and execute().

6.283.5.63 windowSize

```
const uint64_t RobTimer::windowSize  [private]
```

Definition at line 50 of file rob_timer.h.

Referenced by doDispatch(), and RobTimer().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/rob_performance_model/ **rob_timer.h**
- common/performance_model/performance_models/rob_performance_model/ **rob_timer.cc**

6.284 RoutineTracer::Routine Class Reference

```
#include <routine_tracer.h>
```

Inheritance diagram for RoutineTracer::Routine:

Public Member Functions

- **Routine** (**IntPtr** eip, const char ∗name, const char ∗imgname, **IntPtr** offset, int column, int line, const char ∗filename)
- void **updateLocation** (const char ∗name, const char ∗imgname, **IntPtr** offset, int column, int line, const char ∗filename)

Public Attributes

- const **IntPtr** **m_eip**
- char ∗ **m_name**
- char ∗ **m_imgname**
- char ∗ **m_filename**
- char ∗ **m_location**
- **IntPtr** **m_offset**
- int **m_column**
- int **m_line**

6.284.1 Detailed Description

Definition at line 64 of file routine_tracer.h.

6.284.2 Constructor & Destructor Documentation

6.284.2.1 Routine()

```
RoutineTracer::Routine::Routine (
             IntPtr eip,
            const char * name,
            const char * imgname,
             IntPtr offset,
            int column,
            int line,
            const char * filename )
```

Definition at line 176 of file routine_tracer.cc.

References updateLocation().

6.284.3 Member Function Documentation

6.284.3.1 updateLocation()

```
void RoutineTracer::Routine::updateLocation (
            const char * name,
            const char * imgname,
             IntPtr offset,
            int column,
            int line,
            const char * filename )
```

Definition at line 186 of file routine_tracer.cc.

Referenced by Routine().

6.284.4 Member Data Documentation

6.284.4.1 m_column

```
int RoutineTracer::Routine::m_column
```

Definition at line 70 of file routine_tracer.h.

6.284.4.2 m_eip

```
const  IntPtr RoutineTracer::Routine::m_eip
```

Definition at line 67 of file routine_tracer.h.

Referenced by RoutineTracerFunctionStats::RtnMaster::ce_notify_evict().

6.284.4.3 m_filename

```
char * RoutineTracer::Routine::m_filename
```

Definition at line 68 of file routine_tracer.h.

6.284.4.4 m_imgname

```
char * RoutineTracer::Routine::m_imgname
```

Definition at line 68 of file routine_tracer.h.

6.284.4.5 m_line

```
int RoutineTracer::Routine::m_line
```

Definition at line 70 of file routine_tracer.h.

Referenced by RoutineTracerPrint::RtnThread::functionEnter().

6.284.4.6 m_location

```
char * RoutineTracer::Routine::m_location
```

Definition at line 68 of file routine_tracer.h.

Referenced by MemoryTracker::~MemoryTracker().

6.284.4.7 m_name

```
char* RoutineTracer::Routine::m_name
```

Definition at line 68 of file routine_tracer.h.

Referenced by RoutineTracerPrint::RtnThread::functionEnter(), RoutineTracerOndemand::RtnThread::printStack(), and MemoryTracker::~MemoryTracker().

6.284.4.8 m_offset

```
IntPtr RoutineTracer::Routine::m_offset
```

Definition at line 69 of file routine_tracer.h.

The documentation for this class was generated from the following files:

- common/system/ **routine_tracer.h**
- common/system/ **routine_tracer.cc**

6.285 RoutineTracerFunctionStats::Routine Class Reference

```
#include <routine_tracer_funcstats.h>
```

Inheritance diagram for RoutineTracerFunctionStats::Routine:

Public Member Functions

- **Routine** (**IntPtr** eip, const char *name, const char *imgname, **IntPtr** offset, int column, int line, const char *filename)
- bool **isProvisional** () const
- void **setProvisional** (bool provisional)
- **Routine** (const **Routine** &r)

Public Attributes

- bool **m_provisional**
- **UInt64** **m_calls**
- **RtnValues** **m_values**
- **UInt64** **m_bits_used**
- **UInt64** **m_bits_total**

6.285.1 Detailed Description

Definition at line 16 of file routine_tracer_funcstats.h.

6.285.2 Constructor & Destructor Documentation

6.285.2.1 Routine() [1/2]

```
RoutineTracerFunctionStats::Routine::Routine (
             IntPtr eip,
            const char * name,
            const char * imgname,
             IntPtr offset,
            int column,
            int line,
            const char * filename )  [inline]
```

Definition at line 24 of file routine_tracer_funcstats.h.

6.285.2.2 Routine() [2/2]

```
RoutineTracerFunctionStats::Routine::Routine (
            const  Routine & r )  [inline]
```

Definition at line 33 of file routine_tracer_funcstats.h.

6.285.3 Member Function Documentation

6.285.3.1 isProvisional()

```
bool RoutineTracerFunctionStats::Routine::isProvisional ( ) const  [inline]
```

Definition at line 29 of file routine_tracer_funcstats.h.

References m_provisional.

6.285.3.2 setProvisional()

```
void RoutineTracerFunctionStats::Routine::setProvisional (
            bool provisional )  [inline]
```

Definition at line 30 of file routine_tracer_funcstats.h.

References m_provisional.

6.285.4 Member Data Documentation

6.285.4.1 m_bits_total

UInt64 RoutineTracerFunctionStats::Routine::m_bits_total

Definition at line 22 of file routine_tracer_funcstats.h.

Referenced by RoutineTracerFunctionStats::RtnMaster::ce_notify_evict().

6.285.4.2 m_bits_used

UInt64 RoutineTracerFunctionStats::Routine::m_bits_used

Definition at line 22 of file routine_tracer_funcstats.h.

Referenced by RoutineTracerFunctionStats::RtnMaster::ce_notify_evict().

6.285.4.3 m_calls

UInt64 RoutineTracerFunctionStats::Routine::m_calls

Definition at line 20 of file routine_tracer_funcstats.h.

Referenced by RoutineTracerFunctionStats::RtnMaster::updateRoutineFull().

6.285.4.4 m_provisional

bool RoutineTracerFunctionStats::Routine::m_provisional

Definition at line 19 of file routine_tracer_funcstats.h.

Referenced by isProvisional(), and setProvisional().

6.285.4.5 m_values

RtnValues RoutineTracerFunctionStats::Routine::m_values

Definition at line 21 of file routine_tracer_funcstats.h.

Referenced by RoutineTracerFunctionStats::RtnMaster::updateRoutineFull().

The documentation for this class was generated from the following file:

- common/system/ **routine_tracer_funcstats.h**

6.286 MemoryTracker::RoutineTracer Class Reference

#include <memory_tracker.h>

Inheritance diagram for MemoryTracker::RoutineTracer:

Public Member Functions

- **RoutineTracer** ()
- virtual ~**RoutineTracer** ()
- virtual **RoutineTracerThread** * **getThreadHandler** (**Thread** *thread)
- virtual void **addRoutine** (**IntPtr** eip, const char *name, const char *imgname, **IntPtr** offset, int column, int line, const char *filename)
- virtual bool **hasRoutine** (**IntPtr** eip)
- virtual const **Routine** * **getRoutineInfo** (**IntPtr** eip)

Private Types

- typedef std::unordered_map< **IntPtr**, **RoutineTracer::Routine** * > **RoutineMap**

Private Attributes

- **Lock m_lock**
- **RoutineMap m_routines**

Additional Inherited Members

6.286.1 Detailed Description

Definition at line 33 of file memory_tracker.h.

6.286.2 Member Typedef Documentation

6.286.2.1 RoutineMap

```
typedef std::unordered_map< IntPtr, RoutineTracer::Routine* > MemoryTracker::RoutineTracer
::RoutineMap [private]
```

Definition at line 47 of file memory_tracker.h.

6.286.3 Constructor & Destructor Documentation

6.286.3.1 RoutineTracer()

```
MemoryTracker::RoutineTracer::RoutineTracer ( )
```

Definition at line 177 of file memory_tracker.cc.

References MemoryTracker::MemoryTracker().

6.286.3.2 ~RoutineTracer()

```
MemoryTracker::RoutineTracer::~RoutineTracer ( ) [virtual]
```

Reimplemented from **RoutineTracer** (p. 1073).

Definition at line 182 of file memory_tracker.cc.

6.286.4 Member Function Documentation

6.286.4.1 addRoutine()

```
void MemoryTracker::RoutineTracer::addRoutine (
             IntPtr eip,
            const char * name,
            const char * imgname,
             IntPtr offset,
            int column,
            int line,
            const char * filename ) [virtual]
```

Implements **RoutineTracer** (p. 1073).

Definition at line 187 of file memory_tracker.cc.

References MemoryTracker::m_lock.

6.286.4.2 getRoutineInfo()

```
virtual const  Routine* MemoryTracker::RoutineTracer::getRoutineInfo (
              IntPtr eip )  [inline], [virtual]
```

Reimplemented from **RoutineTracer** (p. 1073).

Definition at line 43 of file memory_tracker.h.

References m_routines.

6.286.4.3 getThreadHandler()

```
virtual  RoutineTracerThread* MemoryTracker::RoutineTracer::getThreadHandler (
              Thread * thread )  [inline], [virtual]
```

Implements **RoutineTracer** (p. 1074).

Definition at line 39 of file memory_tracker.h.

6.286.4.4 hasRoutine()

```
bool MemoryTracker::RoutineTracer::hasRoutine (
              IntPtr eip )  [virtual]
```

Implements **RoutineTracer** (p. 1074).

Definition at line 197 of file memory_tracker.cc.

References MemoryTracker::m_lock.

6.286.5 Member Data Documentation

6.286.5.1 m_lock

```
Lock MemoryTracker::RoutineTracer::m_lock  [private]
```

Definition at line 46 of file memory_tracker.h.

6.286.5.2 m_routines

RoutineMap MemoryTracker::RoutineTracer::m_routines [private]

Definition at line 48 of file memory_tracker.h.

Referenced by getRoutineInfo().

The documentation for this class was generated from the following files:

- common/system/ **memory_tracker.h**
- common/system/ **memory_tracker.cc**

6.287 RoutineTracer Class Reference

#include <routine_tracer.h>

Inheritance diagram for RoutineTracer:

Classes

- class **Routine**

Public Member Functions

- **RoutineTracer** ()
- virtual ~**RoutineTracer** ()
- virtual void **addRoutine** (**IntPtr** eip, const char *name, const char *imgname, **IntPtr** offset, int column, int line, const char *filename)=0
- virtual bool **hasRoutine** (**IntPtr** eip)=0
- virtual **RoutineTracerThread** * **getThreadHandler** (**Thread** *thread)=0
- virtual const **Routine** * **getRoutineInfo** (**IntPtr** eip)

Static Public Member Functions

- static **RoutineTracer** * **create** ()

6.287.1 Detailed Description

Definition at line 61 of file routine_tracer.h.

6.287.2 Constructor & Destructor Documentation

6.287.2.1 RoutineTracer()

```
RoutineTracer::RoutineTracer ( )
```

Definition at line 211 of file routine_tracer.cc.

6.287.2.2 ~RoutineTracer()

```
RoutineTracer::~RoutineTracer ( )  [virtual]
```

Reimplemented in **MemoryTracker::RoutineTracer** (p. 1070).

Definition at line 215 of file routine_tracer.cc.

6.287.3 Member Function Documentation

6.287.3.1 addRoutine()

```
virtual void RoutineTracer::addRoutine (
             IntPtr eip,
            const char * name,
            const char * imgname,
             IntPtr offset,
            int column,
            int line,
            const char * filename )  [pure virtual]
```

Implemented in **RoutineTracerFunctionStats::RtnMaster** (p. 1087), **MemoryTracker::RoutineTracer** (p. 1070), **RoutineTracerOndemand::RtnMaster** (p. 1092), and **RoutineTracerPrint::RtnMaster** (p. 1094).

6.287.3.2 create()

```
RoutineTracer * RoutineTracer::create ( )  [static]
```

Definition at line 219 of file routine_tracer.cc.

References LOG_PRINT_ERROR.

Referenced by Simulator::start().

6.287.3.3 getRoutineInfo()

```
virtual const  Routine* RoutineTracer::getRoutineInfo (
              IntPtr eip )  [inline], [virtual]
```

Reimplemented in **MemoryTracker::RoutineTracer** (p. 1070).

Definition at line 85 of file routine_tracer.h.

6.287.3.4 getThreadHandler()

```
virtual  RoutineTracerThread* RoutineTracer::getThreadHandler (
              Thread * thread )  [pure virtual]
```

Implemented in **RoutineTracerFunctionStats::RtnMaster** (p. 1088), **MemoryTracker::RoutineTracer** (p. 1071), **RoutineTracerOndemand::RtnMaster** (p. 1092), and **RoutineTracerPrint::RtnMaster** (p. 1095).

6.287.3.5 hasRoutine()

```
virtual bool RoutineTracer::hasRoutine (
              IntPtr eip )  [pure virtual]
```

Implemented in **RoutineTracerFunctionStats::RtnMaster** (p. 1088), **MemoryTracker::RoutineTracer** (p. 1071), **RoutineTracerOndemand::RtnMaster** (p. 1092), and **RoutineTracerPrint::RtnMaster** (p. 1095).

The documentation for this class was generated from the following files:

- common/system/ **routine_tracer.h**
- common/system/ **routine_tracer.cc**

6.288 RoutineTracerFunctionStats Class Reference

```
#include <routine_tracer_funcstats.h>
```

Classes

- class **Routine**
- class **RtnMaster**
- class **RtnThread**
- class **ThreadStatAggregates**
- class **ThreadStatCpiMem**

Public Types

- typedef std::unordered_map< **ThreadStatsManager::ThreadStatType**, **UInt64** > **RtnValues**

6.288.1 Detailed Description

Definition at line 12 of file routine_tracer_funcstats.h.

6.288.2 Member Typedef Documentation

6.288.2.1 RtnValues

typedef std::unordered_map< **ThreadStatsManager::ThreadStatType**, **UInt64**> **RoutineTracer↩ FunctionStats::RtnValues**

Definition at line 15 of file routine_tracer_funcstats.h.

The documentation for this class was generated from the following file:

- common/system/ **routine_tracer_funcstats.h**

6.289 RoutineTracerOndemand Class Reference

#include <routine_tracer_ondemand.h>

Classes

- class **RtnMaster**
- class **RtnThread**

6.289.1 Detailed Description

Definition at line 8 of file routine_tracer_ondemand.h.

The documentation for this class was generated from the following file:

- common/system/ **routine_tracer_ondemand.h**

6.290 RoutineTracerPrint Class Reference

#include <routine_tracer_print.h>

Classes

- class **RtnMaster**
- class **RtnThread**

6.290.1 Detailed Description

Definition at line 8 of file routine_tracer_print.h.

The documentation for this class was generated from the following file:

- common/system/ **routine_tracer_print.h**

6.291 MemoryTracker::RoutineTracerThread Class Reference

`#include <memory_tracker.h>`

Inheritance diagram for MemoryTracker::RoutineTracerThread:

Public Member Functions

- **RoutineTracerThread** (**Thread** ∗thread)
- const **CallStack** & **getCallsiteStack** () const

Protected Member Functions

- virtual void **functionEnter** (**IntPtr** eip, **IntPtr** callEip)
- virtual void **functionExit** (**IntPtr** eip)
- virtual void **functionChildEnter** (**IntPtr** eip, **IntPtr** eip_child)
- virtual void **functionChildExit** (**IntPtr** eip, **IntPtr** eip_child)

Private Attributes

- **CallStack m_callsite_stack**

Additional Inherited Members

6.291.1 Detailed Description

Definition at line 20 of file memory_tracker.h.

6.291.2 Constructor & Destructor Documentation

6.291.2.1 RoutineTracerThread()

```
MemoryTracker::RoutineTracerThread::RoutineTracerThread (
              Thread * thread )  [inline]
```

Definition at line 23 of file memory_tracker.h.

6.291.3 Member Function Documentation

6.291.3.1 functionChildEnter()

```
virtual void MemoryTracker::RoutineTracerThread::functionChildEnter (
              IntPtr eip,
              IntPtr eip_child )  [inline], [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1080).

Definition at line 28 of file memory_tracker.h.

6.291.3.2 functionChildExit()

```
virtual void MemoryTracker::RoutineTracerThread::functionChildExit (
              IntPtr eip,
              IntPtr eip_child )  [inline], [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1080).

Definition at line 29 of file memory_tracker.h.

6.291.3.3 functionEnter()

```
void MemoryTracker::RoutineTracerThread::functionEnter (
              IntPtr eip,
              IntPtr callEip )  [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1081).

Definition at line 204 of file memory_tracker.cc.

6.291.3.4 functionExit()

```
void MemoryTracker::RoutineTracerThread::functionExit (
              IntPtr eip )  [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1081).

Definition at line 209 of file memory_tracker.cc.

6.291.3.5 getCallsiteStack()

```
const  CallStack& MemoryTracker::RoutineTracerThread::getCallsiteStack ( ) const  [inline]
```

Definition at line 24 of file memory_tracker.h.

References m_callsite_stack.

6.291.4 Member Data Documentation

6.291.4.1 m_callsite_stack

```
 CallStack MemoryTracker::RoutineTracerThread::m_callsite_stack  [private]
```

Definition at line 31 of file memory_tracker.h.

Referenced by getCallsiteStack().

The documentation for this class was generated from the following files:

- common/system/ **memory_tracker.h**
- common/system/ **memory_tracker.cc**

6.292 RoutineTracerThread Class Reference

```
#include <routine_tracer.h>
```

Inheritance diagram for RoutineTracerThread:

Public Member Functions

- **RoutineTracerThread** (**Thread** *thread)
- virtual ~**RoutineTracerThread** ()
- void **routineEnter** (**IntPtr** eip, **IntPtr** esp, **IntPtr** returnEip)
- void **routineExit** (**IntPtr** eip, **IntPtr** esp)
- void **routineAssert** (**IntPtr** eip, **IntPtr** esp)
- const **CallStack** & **getCallStack** () const

Protected Attributes

- **Lock m_lock**
- **Thread** * **m_thread**
- **CallStack m_stack**
- **IntPtr m_last_esp**

Private Member Functions

- bool **unwindTo** (**IntPtr** eip)
- void **routineEnter_unlocked** (**IntPtr** eip, **IntPtr** esp, **IntPtr** callEip)
- virtual void **functionEnter** (**IntPtr** eip, **IntPtr** callEip)=0
- virtual void **functionExit** (**IntPtr** eip)=0
- virtual void **functionChildEnter** (**IntPtr** eip, **IntPtr** eip_child)=0
- virtual void **functionChildExit** (**IntPtr** eip, **IntPtr** eip_child)=0
- void **hookRoiBegin** ()
- void **hookRoiEnd** ()

Static Private Member Functions

- static **SInt64** **__hook_roi_begin** (**UInt64** user, **UInt64** arg)
- static **SInt64** **__hook_roi_end** (**UInt64** user, **UInt64** arg)

6.292.1 Detailed Description

Definition at line 27 of file routine_tracer.h.

6.292.2 Constructor & Destructor Documentation

6.292.2.1 RoutineTracerThread()

```
RoutineTracerThread::RoutineTracerThread (
            Thread * thread )
```

Definition at line 12 of file routine_tracer.cc.

References __hook_roi_begin(), __hook_roi_end(), HookType::HOOK_ROI_BEGIN, and HookType::HOOK_ROI↩_END.

6.292.2.2 ~RoutineTracerThread()

```
RoutineTracerThread::~RoutineTracerThread ( )  [virtual]
```

Definition at line 19 of file routine_tracer.cc.

6.292.3 Member Function Documentation

6.292.3.1 __hook_roi_begin()

```
static  SInt64 RoutineTracerThread::__hook_roi_begin (
              UInt64 user,
              UInt64 arg )  [inline], [static], [private]
```

Definition at line 57 of file routine_tracer.h.

Referenced by RoutineTracerThread().

6.292.3.2 __hook_roi_end()

```
static  SInt64 RoutineTracerThread::__hook_roi_end (
              UInt64 user,
              UInt64 arg )  [inline], [static], [private]
```

Definition at line 58 of file routine_tracer.h.

Referenced by RoutineTracerThread().

6.292.3.3 functionChildEnter()

```
virtual void RoutineTracerThread::functionChildEnter (
              IntPtr eip,
              IntPtr eip_child )  [private], [pure virtual]
```

Implemented in **RoutineTracerFunctionStats::RtnThread** (p. 1098), **RoutineTracerOndemand::RtnThread** (p. 1104), **RoutineTracerPrint::RtnThread** (p. 1101), and **MemoryTracker::RoutineTracerThread** (p. 1077).

Referenced by hookRoiBegin(), and routineEnter_unlocked().

6.292.3.4 functionChildExit()

```
virtual void RoutineTracerThread::functionChildExit (
              IntPtr eip,
              IntPtr eip_child )  [private], [pure virtual]
```

Implemented in **RoutineTracerFunctionStats::RtnThread** (p. 1098), **RoutineTracerOndemand::RtnThread** (p. 1104), **RoutineTracerPrint::RtnThread** (p. 1101), and **MemoryTracker::RoutineTracerThread** (p. 1077).

Referenced by hookRoiEnd(), routineExit(), and unwindTo().

6.292.3.5 functionEnter()

```
virtual void RoutineTracerThread::functionEnter (
              IntPtr eip,
              IntPtr callEip )  [private], [pure virtual]
```

Implemented in **RoutineTracerFunctionStats::RtnThread** (p. 1099), **RoutineTracerOndemand::RtnThread** (p. 1104), **RoutineTracerPrint::RtnThread** (p. 1102), and **MemoryTracker::RoutineTracerThread** (p. 1077).

Referenced by hookRoiBegin(), and routineEnter_unlocked().

6.292.3.6 functionExit()

```
virtual void RoutineTracerThread::functionExit (
              IntPtr eip )  [private], [pure virtual]
```

Implemented in **RoutineTracerFunctionStats::RtnThread** (p. 1099), **RoutineTracerOndemand::RtnThread** (p. 1104), **RoutineTracerPrint::RtnThread** (p. 1102), and **MemoryTracker::RoutineTracerThread** (p. 1077).

Referenced by hookRoiEnd(), routineExit(), and unwindTo().

6.292.3.7 getCallStack()

```
const  CallStack& RoutineTracerThread::getCallStack ( ) const  [inline]
```

Definition at line 37 of file routine_tracer.h.

References m_stack.

6.292.3.8 hookRoiBegin()

```
void RoutineTracerThread::hookRoiBegin ( )  [private]
```

Definition at line 136 of file routine_tracer.cc.

References functionChildEnter(), functionEnter(), m_lock, and m_stack.

6.292.3.9 hookRoiEnd()

```
void RoutineTracerThread::hookRoiEnd ( )  [private]
```

Definition at line 150 of file routine_tracer.cc.

References functionChildExit(), functionExit(), m_lock, and m_stack.

6.292.3.10 routineAssert()

```
void RoutineTracerThread::routineAssert (
              IntPtr eip,
              IntPtr esp )
```

Definition at line 76 of file routine_tracer.cc.

References LOG_ASSERT_ERROR, m_last_esp, m_lock, m_stack, routineEnter_unlocked(), and unwindTo().

Referenced by TraceThread::handleRoutineChangeFunc().

6.292.3.11 routineEnter()

```
void RoutineTracerThread::routineEnter (
              IntPtr eip,
              IntPtr esp,
              IntPtr returnEip )
```

Definition at line 23 of file routine_tracer.cc.

References m_lock, and routineEnter_unlocked().

Referenced by TraceThread::handleRoutineChangeFunc().

6.292.3.12 routineEnter_unlocked()

```
void RoutineTracerThread::routineEnter_unlocked (
              IntPtr eip,
              IntPtr esp,
              IntPtr callEip )  [private]
```

Definition at line 30 of file routine_tracer.cc.

References functionChildEnter(), functionEnter(), m_last_esp, and m_stack.

Referenced by routineAssert(), and routineEnter().

6.292.3.13 routineExit()

```
void RoutineTracerThread::routineExit (
              IntPtr eip,
              IntPtr esp )
```

Definition at line 43 of file routine_tracer.cc.

References functionChildExit(), functionExit(), m_last_esp, m_lock, m_stack, and unwindTo().

Referenced by TraceThread::handleRoutineChangeFunc().

6.292.3.14 unwindTo()

```
bool RoutineTracerThread::unwindTo (
              IntPtr eip )  [private]
```

Definition at line 115 of file routine_tracer.cc.

References functionChildExit(), functionExit(), and m_stack.

Referenced by routineAssert(), and routineExit().

6.292.4 Member Data Documentation

6.292.4.1 m_last_esp

```
IntPtr RoutineTracerThread::m_last_esp  [protected]
```

Definition at line 43 of file routine_tracer.h.

Referenced by routineAssert(), routineEnter_unlocked(), and routineExit().

6.292.4.2 m_lock

```
Lock RoutineTracerThread::m_lock  [protected]
```

Definition at line 40 of file routine_tracer.h.

Referenced by hookRoiBegin(), hookRoiEnd(), routineAssert(), routineEnter(), and routineExit().

6.292.4.3 m_stack

```
CallStack RoutineTracerThread::m_stack  [protected]
```

Definition at line 42 of file routine_tracer.h.

Referenced by getCallStack(), hookRoiBegin(), hookRoiEnd(), RoutineTracerOndemand::RtnThread::printStack(), routineAssert(), routineEnter_unlocked(), routineExit(), and unwindTo().

6.292.4.4 m_thread

```
Thread* RoutineTracerThread::m_thread  [protected]
```

Definition at line 41 of file routine_tracer.h.

Referenced by RoutineTracerOndemand::RtnThread::printStack().

The documentation for this class was generated from the following files:

- common/system/ **routine_tracer.h**
- common/system/ **routine_tracer.cc**

6.293 RoutineTracerFunctionStats::RtnMaster Class Reference

```
#include <routine_tracer_funcstats.h>
```

Inheritance diagram for RoutineTracerFunctionStats::RtnMaster:

Public Member Functions

- **RtnMaster** ()
- virtual ~**RtnMaster** ()
- virtual **RoutineTracerThread** * **getThreadHandler** (**Thread** *thread)
- virtual void **addRoutine** (**IntPtr** eip, const char *name, const char *imgname, **IntPtr** offset, int column, int line, const char *filename)
- virtual bool **hasRoutine** (**IntPtr** eip)
- void **updateRoutine** (**IntPtr** eip, **UInt64** calls, **RtnValues** values)
- void **updateRoutineFull** (const **CallStack** &stack, **UInt64** calls, **RtnValues** values)
- void **updateRoutineFull** (**RoutineTracerFunctionStats::Routine** *rtn, **UInt64** calls, **RtnValues** values)
- **RoutineTracerFunctionStats::Routine** * **getRoutineFullPtr** (const **CallStack** &stack)

Public Attributes

- std::unordered_map< **thread_id_t**, **RtnThread** * > **m_threads**

Private Types

- typedef std::unordered_map< **IntPtr**, **RoutineTracerFunctionStats::Routine** * > **RoutineMap**
- typedef std::unordered_map< **CallStack**, **RoutineTracerFunctionStats::Routine** * > **RoutineMapFull**

Private Member Functions

- **UInt64** **ce_get_owner** (**core_id_t** core_id, **UInt64** address)
- void **ce_notify_evict** (bool on_roi_end, **UInt64** owner, **UInt64** evictor, **CacheBlockInfo::BitsUsedType** bits_used, **UInt32** bits_total)
- void **writeResults** (const char *filename)
- void **writeResultsFull** (const char *filename)

Static Private Member Functions

- static **UInt64** **__ce_get_owner** (**UInt64** user, **core_id_t** core_id, **UInt64** address)
- static void **__ce_notify_evict** (**UInt64** user, bool on_roi_end, **UInt64** owner, **UInt64** evictor, **CacheBlockInfo::BitsUsedType** bits_used, **UInt32** bits_total)

Private Attributes

- **Lock** **m_lock**
- **RoutineMap** **m_routines**
- **RoutineMapFull** **m_callstack_routines**

Additional Inherited Members

6.293.1 Detailed Description

Definition at line 40 of file routine_tracer_funcstats.h.

6.293.2 Member Typedef Documentation

6.293.2.1 RoutineMap

typedef std::unordered_map< **IntPtr**, **RoutineTracerFunctionStats::Routine**∗> **RoutineTracer↩FunctionStats::RtnMaster::RoutineMap** [private]

Definition at line 59 of file routine_tracer_funcstats.h.

6.293.2.2 RoutineMapFull

typedef std::unordered_map< **CallStack**, **RoutineTracerFunctionStats::Routine**∗> **RoutineTracer↩FunctionStats::RtnMaster::RoutineMapFull** [private]

Definition at line 62 of file routine_tracer_funcstats.h.

6.293.3 Constructor & Destructor Documentation

6.293.3.1 RtnMaster()

RoutineTracerFunctionStats::RtnMaster::RtnMaster ( )

Definition at line 107 of file routine_tracer_funcstats.cc.

References ThreadStatsManager::INVALID, RoutineTracerFunctionStats::ThreadStatCpiMem::registerStat(), ThreadStatNamedStat::registerStat(), and RoutineTracerFunctionStats::ThreadStatAggregates::registerStats().

6.293.3.2 ∼RtnMaster()

RoutineTracerFunctionStats::RtnMaster::∼RtnMaster ( ) [virtual]

Definition at line 122 of file routine_tracer_funcstats.cc.

6.293.4 Member Function Documentation

6.293.4.1 __ce_get_owner()

```
static  UInt64 RoutineTracerFunctionStats::RtnMaster::__ce_get_owner (
              UInt64 user,
              core_id_t core_id,
              UInt64 address )  [inline], [static], [private]
```

Definition at line 68 of file routine_tracer_funcstats.h.

6.293.4.2 __ce_notify_evict()

```
static void RoutineTracerFunctionStats::RtnMaster::__ce_notify_evict (
              UInt64 user,
             bool on_roi_end,
              UInt64 owner,
              UInt64 evictor,
              CacheBlockInfo::BitsUsedType bits_used,
              UInt32 bits_total )  [inline], [static], [private]
```

Definition at line 70 of file routine_tracer_funcstats.h.

6.293.4.3 addRoutine()

```
void RoutineTracerFunctionStats::RtnMaster::addRoutine (
              IntPtr eip,
             const char * name,
             const char * imgname,
              IntPtr offset,
             int column,
             int line,
             const char * filename )  [virtual]
```

Implements **RoutineTracer** (p. 1073).

Definition at line 162 of file routine_tracer_funcstats.cc.

6.293.4.4 ce_get_owner()

```
 UInt64 RoutineTracerFunctionStats::RtnMaster::ce_get_owner (
              core_id_t core_id,
              UInt64 address )  [private]
```

Definition at line 128 of file routine_tracer_funcstats.cc.

References Thread::getId().

6.293.4.5 ce_notify_evict()

```
void RoutineTracerFunctionStats::RtnMaster::ce_notify_evict (
            bool on_roi_end,
             UInt64 owner,
             UInt64 evictor,
             CacheBlockInfo::BitsUsedType bits_used,
             UInt32 bits_total )  [private]
```

Definition at line 137 of file routine_tracer_funcstats.cc.

References countBits(), LOG_ASSERT_ERROR, RoutineTracerFunctionStats::Routine::m_bits_total, RoutineTracerFunctionStats::Routine::m_bits_used, and RoutineTracer::Routine::m_eip.

6.293.4.6 getRoutineFullPtr()

```
 RoutineTracerFunctionStats::Routine * RoutineTracerFunctionStats::RtnMaster::getRoutineFullPtr (
            const  CallStack & stack )
```

Definition at line 206 of file routine_tracer_funcstats.cc.

6.293.4.7 getThreadHandler()

```
 RoutineTracerThread * RoutineTracerFunctionStats::RtnMaster::getThreadHandler (
             Thread * thread )  [virtual]
```

Implements **RoutineTracer** (p. 1074).

Definition at line 155 of file routine_tracer_funcstats.cc.

References Thread::getId().

6.293.4.8 hasRoutine()

```
bool RoutineTracerFunctionStats::RtnMaster::hasRoutine (
             IntPtr eip )  [virtual]
```

Implements **RoutineTracer** (p. 1074).

Definition at line 177 of file routine_tracer_funcstats.cc.

6.293.4.9 updateRoutine()

```
void RoutineTracerFunctionStats::RtnMaster::updateRoutine (
              IntPtr eip,
              UInt64 calls,
              RtnValues values )
```

Definition at line 184 of file routine_tracer_funcstats.cc.

References LOG_ASSERT_ERROR.

6.293.4.10 updateRoutineFull() [1/2]

```
void RoutineTracerFunctionStats::RtnMaster::updateRoutineFull (
             const  CallStack & stack,
              UInt64 calls,
              RtnValues values )
```

Definition at line 224 of file routine_tracer_funcstats.cc.

6.293.4.11 updateRoutineFull() [2/2]

```
void RoutineTracerFunctionStats::RtnMaster::updateRoutineFull (
              RoutineTracerFunctionStats::Routine * rtn,
              UInt64 calls,
              RtnValues values )
```

Definition at line 229 of file routine_tracer_funcstats.cc.

References RoutineTracerFunctionStats::Routine::m_calls, and RoutineTracerFunctionStats::Routine::m_values.

6.293.4.12 writeResults()

```
void RoutineTracerFunctionStats::RtnMaster::writeResults (
             const char * filename )  [private]
```

Definition at line 240 of file routine_tracer_funcstats.cc.

6.293.4.13 writeResultsFull()

```
void RoutineTracerFunctionStats::RtnMaster::writeResultsFull (
             const char * filename )  [private]
```

Definition at line 263 of file routine_tracer_funcstats.cc.

6.293.5 Member Data Documentation

6.293.5.1 m_callstack_routines

RoutineMapFull RoutineTracerFunctionStats::RtnMaster::m_callstack_routines [private]

Definition at line 63 of file routine_tracer_funcstats.h.

6.293.5.2 m_lock

Lock RoutineTracerFunctionStats::RtnMaster::m_lock [private]

Definition at line 57 of file routine_tracer_funcstats.h.

6.293.5.3 m_routines

RoutineMap RoutineTracerFunctionStats::RtnMaster::m_routines [private]

Definition at line 60 of file routine_tracer_funcstats.h.

6.293.5.4 m_threads

std::unordered_map< **thread_id_t**, **RtnThread***> RoutineTracerFunctionStats::RtnMaster::m_threads

Definition at line 44 of file routine_tracer_funcstats.h.

The documentation for this class was generated from the following files:

- common/system/ **routine_tracer_funcstats.h**
- common/system/ **routine_tracer_funcstats.cc**

6.294 RoutineTracerOndemand::RtnMaster Class Reference

#include <routine_tracer_ondemand.h>

Inheritance diagram for RoutineTracerOndemand::RtnMaster:

Public Member Functions

- **RtnMaster** ()
- virtual ~**RtnMaster** ()
- virtual **RoutineTracerThread** * **getThreadHandler** (**Thread** *thread)
- virtual void **addRoutine** (**IntPtr** eip, const char *name, const char *imgname, **IntPtr** offset, int column, int line, const char *filename)
- virtual bool **hasRoutine** (**IntPtr** eip)
- **RoutineTracer::Routine** * **getRoutine** (**IntPtr** eip)

Static Private Member Functions

- static **SInt64** **signalHandler** (**UInt64**, **UInt64**)

Private Attributes

- **Lock** **m_lock**
- std::unordered_map< **IntPtr**, **RoutineTracer::Routine** * > **m_routines**

Additional Inherited Members

6.294.1 Detailed Description

Definition at line 11 of file routine_tracer_ondemand.h.

6.294.2 Constructor & Destructor Documentation

6.294.2.1 RtnMaster()

```
RoutineTracerOndemand::RtnMaster::RtnMaster ( )
```

Definition at line 28 of file routine_tracer_ondemand.cc.

References HookType::HOOK_SIGUSR1.

6.294.2.2 ~RtnMaster()

```
virtual RoutineTracerOndemand::RtnMaster::~RtnMaster ( )  [inline], [virtual]
```

Definition at line 15 of file routine_tracer_ondemand.h.

6.294.3 Member Function Documentation

6.294.3.1 addRoutine()

```
void RoutineTracerOndemand::RtnMaster::addRoutine (
              IntPtr eip,
             const char * name,
             const char * imgname,
              IntPtr offset,
             int column,
             int line,
             const char * filename )  [virtual]
```

Implements **RoutineTracer** (p. 1073).

Definition at line 50 of file routine_tracer_ondemand.cc.

6.294.3.2 getRoutine()

```
RoutineTracer::Routine * RoutineTracerOndemand::RtnMaster::getRoutine (
             IntPtr eip )
```

Definition at line 66 of file routine_tracer_ondemand.cc.

Referenced by RoutineTracerOndemand::RtnThread::printStack().

6.294.3.3 getThreadHandler()

```
virtual  RoutineTracerThread* RoutineTracerOndemand::RtnMaster::getThreadHandler (
             Thread * thread )  [inline], [virtual]
```

Implements **RoutineTracer** (p. 1074).

Definition at line 17 of file routine_tracer_ondemand.h.

6.294.3.4 hasRoutine()

```
bool RoutineTracerOndemand::RtnMaster::hasRoutine (
             IntPtr eip )  [virtual]
```

Implements **RoutineTracer** (p. 1074).

Definition at line 60 of file routine_tracer_ondemand.cc.

6.294.3.5 signalHandler()

```
SInt64 RoutineTracerOndemand::RtnMaster::signalHandler (
            UInt64 ,
            UInt64  )  [static], [private]
```

Definition at line 33 of file routine_tracer_ondemand.cc.

References Thread::getRoutineTracer(), LOG_ASSERT_ERROR, and RoutineTracerOndemand::RtnThread↩ ::printStack().

6.294.4 Member Data Documentation

6.294.4.1 m_lock

```
Lock RoutineTracerOndemand::RtnMaster::m_lock  [private]
```

Definition at line 25 of file routine_tracer_ondemand.h.

6.294.4.2 m_routines

```
std::unordered_map< IntPtr,  RoutineTracer::Routine*> RoutineTracerOndemand::RtnMaster::m_↩
routines  [private]
```

Definition at line 26 of file routine_tracer_ondemand.h.

The documentation for this class was generated from the following files:

- common/system/ **routine_tracer_ondemand.h**
- common/system/ **routine_tracer_ondemand.cc**

6.295 RoutineTracerPrint::RtnMaster Class Reference

```
#include <routine_tracer_print.h>
```

Inheritance diagram for RoutineTracerPrint::RtnMaster:

Public Member Functions

- **RtnMaster** ()
- virtual ∼**RtnMaster** ()
- virtual **RoutineTracerThread** ∗ **getThreadHandler** (**Thread** ∗thread)
- virtual void **addRoutine** (**IntPtr** eip, const char ∗name, const char ∗imgname, **IntPtr** offset, int column, int line, const char ∗filename)
- virtual bool **hasRoutine** (**IntPtr** eip)
- **RoutineTracer::Routine** ∗ **getRoutine** (**IntPtr** eip)

Private Attributes

- **Lock m_lock**
- std::unordered_map< **IntPtr**, **RoutineTracer::Routine** ∗ > **m_routines**

Additional Inherited Members

6.295.1 Detailed Description

Definition at line 11 of file routine_tracer_print.h.

6.295.2 Constructor & Destructor Documentation

6.295.2.1 RtnMaster()

```
RoutineTracerPrint::RtnMaster::RtnMaster ( )  [inline]
```

Definition at line 14 of file routine_tracer_print.h.

6.295.2.2 ∼RtnMaster()

```
virtual RoutineTracerPrint::RtnMaster::~RtnMaster ( )  [inline], [virtual]
```

Definition at line 15 of file routine_tracer_print.h.

6.295.3 Member Function Documentation

6.295.3.1 addRoutine()

```
void RoutineTracerPrint::RtnMaster::addRoutine (
              IntPtr eip,
             const char * name,
             const char * imgname,
              IntPtr offset,
             int column,
             int line,
             const char * filename )  [virtual]
```

Implements **RoutineTracer** (p. 1073).

Definition at line 34 of file routine_tracer_print.cc.

6.295.3.2 getRoutine()

```
RoutineTracer::Routine * RoutineTracerPrint::RtnMaster::getRoutine (
              IntPtr eip )
```

Definition at line 51 of file routine_tracer_print.cc.

6.295.3.3 getThreadHandler()

```
RoutineTracerThread * RoutineTracerPrint::RtnMaster::getThreadHandler (
              Thread * thread )  [virtual]
```

Implements **RoutineTracer** (p. 1074).

Definition at line 29 of file routine_tracer_print.cc.

6.295.3.4 hasRoutine()

```
bool RoutineTracerPrint::RtnMaster::hasRoutine (
              IntPtr eip )  [virtual]
```

Implements **RoutineTracer** (p. 1074).

Definition at line 44 of file routine_tracer_print.cc.

6.295.4 Member Data Documentation

6.295.4.1 m_lock

Lock RoutineTracerPrint::RtnMaster::m_lock [private]

Definition at line 23 of file routine_tracer_print.h.

6.295.4.2 m_routines

std::unordered_map< **IntPtr**, **RoutineTracer::Routine***> RoutineTracerPrint::RtnMaster::m_routines [private]

Definition at line 24 of file routine_tracer_print.h.

The documentation for this class was generated from the following files:

- common/system/ **routine_tracer_print.h**
- common/system/ **routine_tracer_print.cc**

6.296 RoutineTracerFunctionStats::RtnThread Class Reference

#include <routine_tracer_funcstats.h>

Inheritance diagram for RoutineTracerFunctionStats::RtnThread:

Public Member Functions

- **RtnThread** (**RtnMaster** *master, **Thread** *thread)
- **UInt64** **getCurrentRoutineId** ()

Protected Member Functions

- virtual void **functionEnter** (**IntPtr** eip, **IntPtr** callEip)
- virtual void **functionExit** (**IntPtr** eip)
- virtual void **functionChildEnter** (**IntPtr** eip, **IntPtr** eip_child)
- virtual void **functionChildExit** (**IntPtr** eip, **IntPtr** eip_child)

Private Member Functions

- void **functionBegin** (**IntPtr** eip)
- void **functionEnd** (**IntPtr** eip, bool is_function_start)
- void **functionBeginHelper** (**IntPtr** eip, **RtnValues** &)
- void **functionEndHelper** (**IntPtr** eip, **UInt64** count)
- void **functionEndFullHelper** (const **CallStack** &stack, **UInt64** count)
- **UInt64 getThreadStat** (**ThreadStatsManager::ThreadStatType** type)

Private Attributes

- **RtnMaster** ∗ **m_master**
- **IntPtr m_current_eip**
- **RtnValues m_values_start**
- std::unordered_map< **CallStack**, **RtnValues** > **m_values_start_full**

Additional Inherited Members

6.296.1 Detailed Description

Definition at line 77 of file routine_tracer_funcstats.h.

6.296.2 Constructor & Destructor Documentation

6.296.2.1 RtnThread()

```
RoutineTracerFunctionStats::RtnThread::RtnThread (
              RoutineTracerFunctionStats::RtnMaster * master,
              Thread * thread )
```

Definition at line 14 of file routine_tracer_funcstats.cc.

6.296.3 Member Function Documentation

6.296.3.1 functionBegin()

```
void RoutineTracerFunctionStats::RtnThread::functionBegin (
              IntPtr eip )  [private]
```

Definition at line 73 of file routine_tracer_funcstats.cc.

6.296.3.2 functionBeginHelper()

```
void RoutineTracerFunctionStats::RtnThread::functionBeginHelper (
              IntPtr eip,
              RtnValues & values_start )  [private]
```

Definition at line 41 of file routine_tracer_funcstats.cc.

6.296.3.3 functionChildEnter()

```
void RoutineTracerFunctionStats::RtnThread::functionChildEnter (
              IntPtr eip,
              IntPtr eip_child )  [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1080).

Definition at line 31 of file routine_tracer_funcstats.cc.

6.296.3.4 functionChildExit()

```
void RoutineTracerFunctionStats::RtnThread::functionChildExit (
              IntPtr eip,
              IntPtr eip_child )  [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1080).

Definition at line 36 of file routine_tracer_funcstats.cc.

6.296.3.5 functionEnd()

```
void RoutineTracerFunctionStats::RtnThread::functionEnd (
              IntPtr eip,
             bool is_function_start )  [private]
```

Definition at line 83 of file routine_tracer_funcstats.cc.

6.296.3.6 functionEndFullHelper()

```
void RoutineTracerFunctionStats::RtnThread::functionEndFullHelper (
             const  CallStack & stack,
              UInt64 count )  [private]
```

Definition at line 62 of file routine_tracer_funcstats.cc.

6.296.3.7 functionEndHelper()

```
void RoutineTracerFunctionStats::RtnThread::functionEndHelper (
              IntPtr eip,
              UInt64 count )  [private]
```

Definition at line 51 of file routine_tracer_funcstats.cc.

6.296.3.8 functionEnter()

```
void RoutineTracerFunctionStats::RtnThread::functionEnter (
              IntPtr eip,
              IntPtr callEip )  [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1081).

Definition at line 21 of file routine_tracer_funcstats.cc.

6.296.3.9 functionExit()

```
void RoutineTracerFunctionStats::RtnThread::functionExit (
              IntPtr eip )  [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1081).

Definition at line 26 of file routine_tracer_funcstats.cc.

6.296.3.10 getCurrentRoutineId()

```
UInt64 RoutineTracerFunctionStats::RtnThread::getCurrentRoutineId ( )
```

Definition at line 97 of file routine_tracer_funcstats.cc.

6.296.3.11 getThreadStat()

```
UInt64 RoutineTracerFunctionStats::RtnThread::getThreadStat (
              ThreadStatsManager::ThreadStatType type )  [private]
```

Definition at line 92 of file routine_tracer_funcstats.cc.

6.296.4 Member Data Documentation

6.296.4.1 m_current_eip

```
IntPtr RoutineTracerFunctionStats::RtnThread::m_current_eip  [private]
```

Definition at line 86 of file routine_tracer_funcstats.h.

6.296.4.2 m_master

```
RtnMaster* RoutineTracerFunctionStats::RtnThread::m_master  [private]
```

Definition at line 84 of file routine_tracer_funcstats.h.

6.296.4.3 m_values_start

```
RtnValues RoutineTracerFunctionStats::RtnThread::m_values_start  [private]
```

Definition at line 87 of file routine_tracer_funcstats.h.

6.296.4.4 m_values_start_full

```
std::unordered_map< CallStack, RtnValues> RoutineTracerFunctionStats::RtnThread::m_values_↩
start_full  [private]
```

Definition at line 88 of file routine_tracer_funcstats.h.

The documentation for this class was generated from the following files:

- common/system/ **routine_tracer_funcstats.h**
- common/system/ **routine_tracer_funcstats.cc**

6.297 RoutineTracerPrint::RtnThread Class Reference

```
#include <routine_tracer_print.h>
```

Inheritance diagram for RoutineTracerPrint::RtnThread:

Public Member Functions

- **RtnThread** (**RtnMaster** ∗master, **Thread** ∗thread)

Protected Member Functions

- virtual void **functionEnter** (**IntPtr** eip, **IntPtr** callEip)
- virtual void **functionExit** (**IntPtr** eip)
- virtual void **functionChildEnter** (**IntPtr** eip, **IntPtr** eip_child)
- virtual void **functionChildExit** (**IntPtr** eip, **IntPtr** eip_child)

Private Attributes

- **RtnMaster** ∗ **m_master**
- **UInt64 m_depth**

Additional Inherited Members

6.297.1 Detailed Description

Definition at line 27 of file routine_tracer_print.h.

6.297.2 Constructor & Destructor Documentation

6.297.2.1 RtnThread()

```
RoutineTracerPrint::RtnThread::RtnThread (
              RoutineTracerPrint::RtnMaster * master,
              Thread * thread )
```

Definition at line 4 of file routine_tracer_print.cc.

6.297.3 Member Function Documentation

6.297.3.1 functionChildEnter()

```
virtual void RoutineTracerPrint::RtnThread::functionChildEnter (
              IntPtr eip,
              IntPtr eip_child )  [inline], [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1080).

Definition at line 39 of file routine_tracer_print.h.

6.297.3.2 functionChildExit()

```
virtual void RoutineTracerPrint::RtnThread::functionChildExit (
              IntPtr eip,
              IntPtr eip_child )  [inline], [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1080).

Definition at line 40 of file routine_tracer_print.h.

6.297.3.3 functionEnter()

```
void RoutineTracerPrint::RtnThread::functionEnter (
              IntPtr eip,
              IntPtr callEip )  [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1081).

Definition at line 11 of file routine_tracer_print.cc.

References RoutineTracer::Routine::m_line, and RoutineTracer::Routine::m_name.

6.297.3.4 functionExit()

```
void RoutineTracerPrint::RtnThread::functionExit (
              IntPtr eip )  [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1081).

Definition at line 24 of file routine_tracer_print.cc.

6.297.4 Member Data Documentation

6.297.4.1 m_depth

```
UInt64 RoutineTracerPrint::RtnThread::m_depth  [private]
```

Definition at line 34 of file routine_tracer_print.h.

6.297.4.2 m_master

```
RtnMaster* RoutineTracerPrint::RtnThread::m_master  [private]
```

Definition at line 33 of file routine_tracer_print.h.

The documentation for this class was generated from the following files:

- common/system/ **routine_tracer_print.h**
- common/system/ **routine_tracer_print.cc**

6.298 RoutineTracerOndemand::RtnThread Class Reference

```
#include <routine_tracer_ondemand.h>
```

Inheritance diagram for RoutineTracerOndemand::RtnThread:

Public Member Functions

- **RtnThread** (**RtnMaster** *master, **Thread** *thread)
- void **printStack** ()

Protected Member Functions

- virtual void **functionEnter** (**IntPtr** eip, **IntPtr** callEip)
- virtual void **functionExit** (**IntPtr** eip)
- virtual void **functionChildEnter** (**IntPtr** eip, **IntPtr** eip_child)
- virtual void **functionChildExit** (**IntPtr** eip, **IntPtr** eip_child)

Private Attributes

- **RtnMaster** * **m_master**

Additional Inherited Members

6.298.1 Detailed Description

Definition at line 29 of file routine_tracer_ondemand.h.

6.298.2 Constructor & Destructor Documentation

6.298.2.1 RtnThread()

```
RoutineTracerOndemand::RtnThread::RtnThread (
              RtnMaster * master,
              Thread * thread )  [inline]
```

Definition at line 32 of file routine_tracer_ondemand.h.

6.298.3 Member Function Documentation

6.298.3.1 functionChildEnter()

```
virtual void RoutineTracerOndemand::RtnThread::functionChildEnter (
              IntPtr eip,
              IntPtr eip_child )  [inline], [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1080).

Definition at line 42 of file routine_tracer_ondemand.h.

6.298.3.2 functionChildExit()

```
virtual void RoutineTracerOndemand::RtnThread::functionChildExit (
              IntPtr eip,
              IntPtr eip_child )  [inline], [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1080).

Definition at line 43 of file routine_tracer_ondemand.h.

6.298.3.3 functionEnter()

```
virtual void RoutineTracerOndemand::RtnThread::functionEnter (
              IntPtr eip,
              IntPtr callEip )  [inline], [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1081).

Definition at line 40 of file routine_tracer_ondemand.h.

6.298.3.4 functionExit()

```
virtual void RoutineTracerOndemand::RtnThread::functionExit (
              IntPtr eip )  [inline], [protected], [virtual]
```

Implements **RoutineTracerThread** (p. 1081).

Definition at line 41 of file routine_tracer_ondemand.h.

6.298.3.5 printStack()

```
void RoutineTracerOndemand::RtnThread::printStack ( )
```

Definition at line 10 of file routine_tracer_ondemand.cc.

References Core::CoreStateString(), SyscallMdl::formatSyscall(), Thread::getAppId(), Thread::getCore(), Thread::getId(), Core::getId(), RoutineTracerOndemand::RtnMaster::getRoutine(), Thread::getSyscallMdl(), SyscallMdl::inSyscall(), m_master, RoutineTracer::Routine::m_name, RoutineTracerThread::m_stack, RoutineTracerThread::m_thread, ThreadManager::stall_type_names, and Core::STALLED.

Referenced by RoutineTracerOndemand::RtnMaster::signalHandler().

6.298.4 Member Data Documentation

6.298.4.1 m_master

```
RtnMaster* RoutineTracerOndemand::RtnThread::m_master  [private]
```

Definition at line 37 of file routine_tracer_ondemand.h.

Referenced by printStack().

The documentation for this class was generated from the following files:

- common/system/ **routine_tracer_ondemand.h**
- common/system/ **routine_tracer_ondemand.cc**

6.299 Runnable Class Reference

```
#include <_thread.h>
```

Inheritance diagram for Runnable:

Public Member Functions

- virtual ∼**Runnable** ()
- virtual void **run** ()=0

Static Public Member Functions

- static void **threadFunc** (void ∗vpRunnable)

6.299.1 Detailed Description

Definition at line 4 of file _thread.h.

6.299.2 Constructor & Destructor Documentation

6.299.2.1 ∼Runnable()

```
virtual Runnable::∼Runnable ( )  [inline], [virtual]
```

Definition at line 7 of file _thread.h.

6.299.3 Member Function Documentation

6.299.3.1 run()

```
virtual void Runnable::run ( )  [pure virtual]
```

Implemented in **TraceThread** (p. 1454), **TraceManager::Monitor** (p. 801), **CoreThread** (p. 415), and **Sim↩ Thread** (p. 1223).

Referenced by threadFunc().

6.299.3.2 threadFunc()

```
static void Runnable::threadFunc (
            void * vpRunnable )  [inline], [static]
```

Definition at line 9 of file _thread.h.

References run().

Referenced by _Thread::create().

The documentation for this class was generated from the following file:

- common/misc/ **_thread.h**

6.300 SamplingAlgorithm Class Reference

`#include <sampling_algorithm.h>`

Inheritance diagram for SamplingAlgorithm:

Public Member Functions

- **SamplingAlgorithm** (**SamplingManager** ∗sampling_manager)
- virtual ∼**SamplingAlgorithm** ()
- virtual void **callbackDetailed** (**SubsecondTime** now)=0
- virtual void **callbackFastForward** (**SubsecondTime** now, bool in_warmup)=0

Static Public Member Functions

- static **SamplingAlgorithm** ∗ **create** (**SamplingManager** ∗sampling_manager)

Protected Attributes

- **SamplingManager** ∗ **m_sampling_manager**

6.300.1 Detailed Description

Definition at line 16 of file sampling_algorithm.h.

6.300.2 Constructor & Destructor Documentation

6.300.2.1 SamplingAlgorithm()

```
SamplingAlgorithm::SamplingAlgorithm (
            SamplingManager * sampling_manager )  [inline]
```

Definition at line 21 of file sampling_algorithm.h.

6.300.2.2 ~SamplingAlgorithm()

```
virtual SamplingAlgorithm::~SamplingAlgorithm ( )  [inline], [virtual]
```

Definition at line 22 of file sampling_algorithm.h.

6.300.3 Member Function Documentation

6.300.3.1 callbackDetailed()

```
virtual void SamplingAlgorithm::callbackDetailed (
              SubsecondTime now )  [pure virtual]
```

Implemented in **PeriodicSampling** (p. 922).

Referenced by SamplingManager::periodic().

6.300.3.2 callbackFastForward()

```
virtual void SamplingAlgorithm::callbackFastForward (
              SubsecondTime now,
             bool in_warmup )  [pure virtual]
```

Implemented in **PeriodicSampling** (p. 922).

Referenced by SamplingManager::periodic().

6.300.3.3 create()

```
SamplingAlgorithm * SamplingAlgorithm::create (
              SamplingManager * sampling_manager )  [static]
```

Definition at line 8 of file sampling_algorithm.cc.

References LOG_PRINT_ERROR.

Referenced by SamplingManager::SamplingManager().

6.300.4 Member Data Documentation

6.300.4.1 m_sampling_manager

```
SamplingManager* SamplingAlgorithm::m_sampling_manager  [protected]
```

Definition at line 19 of file sampling_algorithm.h.

Referenced by PeriodicSampling::callbackDetailed(), PeriodicSampling::callbackFastForward(), and PeriodicSampling::stepFastForward().

The documentation for this class was generated from the following files:

- common/sampling/ **sampling_algorithm.h**
- common/sampling/ **sampling_algorithm.cc**

6.301 SamplingManager Class Reference

`#include <sampling_manager.h>`

Public Member Functions

- **SamplingManager** ()
- ~**SamplingManager** ()
- void **enableFastForward** (**SubsecondTime** until, bool warmup, bool detailed_sync)
- void **disableFastForward** ()
- **SamplingProvider** * **getSamplingProvider** ()
- **SubsecondTime** **getCoreHistoricCPI** (**Core** *core, bool non_idle, **SubsecondTime** min_nonidle_time) const
- void **resetCoreHistoricCPIs** ()

Protected Member Functions

- void **recalibrateInstructionsCallback** (**core_id_t** core_id)

Private Member Functions

- void **setInstrumentationMode** (**InstMode::inst_mode_t** mode)
- void **periodic** (**SubsecondTime** time)
- void **instr_count** (**core_id_t** core_id)

Static Private Member Functions

- static void **hook_instr_count** (**SamplingManager** *self, **core_id_t** core_id)
- static void **hook_periodic** (**SamplingManager** *self, **subsecond_time_t** time)

Private Attributes

- bool **m_sampling_enabled**
- bool **m_uncoordinated**
- bool **m_fastforward**
- bool **m_warmup**
- **SubsecondTime m_target_ffend**
- **SamplingProvider** * **m_sampling_provider**
- **SamplingAlgorithm** * **m_sampling_algorithm**
- std::vector< **UInt64** > **m_instructions**
- std::vector< **SubsecondTime** > **m_time_total**
- std::vector< **SubsecondTime** > **m_time_nonidle**

Friends

- class **FastforwardPerformanceModel**

6.301.1 Detailed Description

Definition at line 15 of file sampling_manager.h.

6.301.2 Constructor & Destructor Documentation

6.301.2.1 SamplingManager()

```
SamplingManager::SamplingManager ( )
```

Definition at line 14 of file sampling_manager.cc.

References SamplingAlgorithm::create(), SamplingProvider::create(), HookType::HOOK_INSTR_COUNT, hook↩ _instr_count(), HookType::HOOK_PERIODIC, hook_periodic(), LOG_ASSERT_ERROR, m_sampling_algorithm, m_sampling_enabled, m_sampling_provider, m_uncoordinated, and Config::PINTOOL.

6.301.2.2 ~SamplingManager()

```
SamplingManager::~SamplingManager ( )
```

Definition at line 39 of file sampling_manager.cc.

References m_sampling_algorithm, and m_sampling_provider.

6.301.3 Member Function Documentation

6.301.3.1 disableFastForward()

```
void SamplingManager::disableFastForward ( )
```

Definition at line 76 of file sampling_manager.cc.

References InstMode::DETAILED, Core::disableInstructionsCallback(), PerformanceModel::getElapsed↩ Time(), PerformanceModel::getFastforwardPerformanceModel(), Core::getPerformanceModel(), Fastforward↩ PerformanceModel::incrementElapsedTime(), m_fastforward, m_uncoordinated, m_warmup, SubsecondTime::↩ MaxTime(), PerformanceModel::setFastForward(), setInstrumentationMode(), and SubsecondTime::Zero().

Referenced by PeriodicSampling::callbackFastForward().

6.301.3.2 enableFastForward()

```
void SamplingManager::enableFastForward (
              SubsecondTime until,
             bool warmup,
             bool detailed_sync )
```

Definition at line 115 of file sampling_manager.cc.

References InstMode::CACHE_ONLY, InstMode::FAST_FORWARD, PerformanceModel::getElapsedTime(), m_↩ fastforward, m_sampling_provider, m_target_ffend, m_warmup, recalibrateInstructionsCallback(), Performance↩ Model::setFastForward(), setInstrumentationMode(), SamplingProvider::startSampling(), and SubsecondTime::↩ Zero().

Referenced by PeriodicSampling::stepFastForward().

6.301.3.3 getCoreHistoricCPI()

```
SubsecondTime SamplingManager::getCoreHistoricCPI (
              Core * core,
             bool non_idle,
              SubsecondTime min_nonidle_time ) const
```

Definition at line 146 of file sampling_manager.cc.

References PerformanceModel::getElapsedTime(), Core::getId(), PerformanceModel::getInstructionCount(), PerformanceModel::getNonIdleElapsedTime(), Core::getPerformanceModel(), m_instructions, m_time_nonidle, m_time_total, SubsecondTime::MaxTime(), and SubsecondTime::Zero().

Referenced by PeriodicSampling::callbackDetailed().

6.301.3.4 getSamplingProvider()

```
SamplingProvider* SamplingManager::getSamplingProvider ( )  [inline]
```

Definition at line 52 of file sampling_manager.h.

References m_sampling_provider.

6.301.3.5 hook_instr_count()

```
static void SamplingManager::hook_instr_count (
              SamplingManager * self,
              core_id_t core_id )  [inline], [static], [private]
```

Definition at line 36 of file sampling_manager.h.

Referenced by SamplingManager().

6.301.3.6 hook_periodic()

```
static void SamplingManager::hook_periodic (
              SamplingManager * self,
              subsecond_time_t time )  [inline], [static], [private]
```

Definition at line 37 of file sampling_manager.h.

Referenced by SamplingManager().

6.301.3.7 instr_count()

```
void SamplingManager::instr_count (
              core_id_t core_id )  [private]
```

Definition at line 202 of file sampling_manager.cc.

References m_fastforward, m_uncoordinated, and SubsecondTime::Zero().

6.301.3.8 periodic()

```
void SamplingManager::periodic (
              SubsecondTime time )  [private]
```

Definition at line 48 of file sampling_manager.cc.

References SamplingAlgorithm::callbackDetailed(), SamplingAlgorithm::callbackFastForward(), m_fastforward, m↩
_sampling_algorithm, m_warmup, and Timer::now().

6.301.3.9 recalibrateInstructionsCallback()

```
void SamplingManager::recalibrateInstructionsCallback (
              core_id_t core_id )  [protected]
```

Definition at line 184 of file sampling_manager.cc.

References SubsecondTime::divideRounded(), FastforwardPerformanceModel::getCurrentCPI(), PerformanceModel::getElapsedTime(), PerformanceModel::getFastforwardPerformanceModel(), Core::getPerformanceModel(), m_target_ffend, Core::setInstructionsCallback(), and SubsecondTime::Zero().

Referenced by enableFastForward().

6.301.3.10 resetCoreHistoricCPIs()

```
void SamplingManager::resetCoreHistoricCPIs ( )
```

Definition at line 172 of file sampling_manager.cc.

References PerformanceModel::getElapsedTime(), Core::getId(), PerformanceModel::getInstructionCount(), PerformanceModel::getNonIdleElapsedTime(), Core::getPerformanceModel(), m_instructions, m_time_nonidle, and m_time_total.

Referenced by PeriodicSampling::callbackDetailed(), and PeriodicSampling::callbackFastForward().

6.301.3.11 setInstrumentationMode()

```
void SamplingManager::setInstrumentationMode (
              InstMode::inst_mode_t mode )  [private]
```

Definition at line 68 of file sampling_manager.cc.

Referenced by disableFastForward(), and enableFastForward().

6.301.4 Friends And Related Function Documentation

6.301.4.1 FastforwardPerformanceModel

```
friend class  FastforwardPerformanceModel  [friend]
```

Definition at line 40 of file sampling_manager.h.

6.301.5 Member Data Documentation

6.301.5.1 m_fastforward

```
bool SamplingManager::m_fastforward  [private]
```

Definition at line 21 of file sampling_manager.h.

Referenced by disableFastForward(), enableFastForward(), instr_count(), and periodic().

6.301.5.2 m_instructions

```
std::vector< UInt64> SamplingManager::m_instructions  [private]
```

Definition at line 28 of file sampling_manager.h.

Referenced by getCoreHistoricCPI(), and resetCoreHistoricCPIs().

6.301.5.3 m_sampling_algorithm

```
SamplingAlgorithm* SamplingManager::m_sampling_algorithm  [private]
```

Definition at line 26 of file sampling_manager.h.

Referenced by periodic(), SamplingManager(), and ~SamplingManager().

6.301.5.4 m_sampling_enabled

```
bool SamplingManager::m_sampling_enabled  [private]
```

Definition at line 18 of file sampling_manager.h.

Referenced by SamplingManager().

6.301.5.5 m_sampling_provider

```
SamplingProvider* SamplingManager::m_sampling_provider  [private]
```

Definition at line 25 of file sampling_manager.h.

Referenced by enableFastForward(), getSamplingProvider(), SamplingManager(), and ~SamplingManager().

6.301.5.6 m_target_ffend

```
SubsecondTime SamplingManager::m_target_ffend [private]
```

Definition at line 23 of file sampling_manager.h.

Referenced by enableFastForward(), and recalibrateInstructionsCallback().

6.301.5.7 m_time_nonidle

```
std::vector< SubsecondTime> SamplingManager::m_time_nonidle [private]
```

Definition at line 30 of file sampling_manager.h.

Referenced by getCoreHistoricCPI(), and resetCoreHistoricCPIs().

6.301.5.8 m_time_total

```
std::vector< SubsecondTime> SamplingManager::m_time_total [private]
```

Definition at line 29 of file sampling_manager.h.

Referenced by getCoreHistoricCPI(), and resetCoreHistoricCPIs().

6.301.5.9 m_uncoordinated

```
bool SamplingManager::m_uncoordinated [private]
```

Definition at line 19 of file sampling_manager.h.

Referenced by disableFastForward(), instr_count(), and SamplingManager().

6.301.5.10 m_warmup

```
bool SamplingManager::m_warmup [private]
```

Definition at line 22 of file sampling_manager.h.

Referenced by disableFastForward(), enableFastForward(), and periodic().

The documentation for this class was generated from the following files:

- common/sampling/ **sampling_manager.h**
- common/sampling/ **sampling_manager.cc**

6.302 SamplingProvider Class Reference

`#include <sampling_provider.h>`

Inheritance diagram for SamplingProvider:

Public Member Functions

- virtual ∼**SamplingProvider** ()
- virtual void **startSampling** (**SubsecondTime** until)=0
- virtual int32_t **registerSignal** ()
- virtual **InstrumentLevel::Level** **requestedInstrumentation** ()=0

Static Public Member Functions

- static **SamplingProvider** ∗ **create** ()

6.302.1 Detailed Description

Definition at line 17 of file sampling_provider.h.

6.302.2 Constructor & Destructor Documentation

6.302.2.1 ∼SamplingProvider()

`virtual SamplingProvider::∼SamplingProvider ( ) [inline], [virtual]`

Definition at line 20 of file sampling_provider.h.

6.302.3 Member Function Documentation

6.302.3.1 create()

```
SamplingProvider * SamplingProvider::create ( )  [static]
```

Definition at line 8 of file sampling_provider.cc.

References LOG_PRINT_ERROR.

Referenced by SamplingManager::SamplingManager().

6.302.3.2 registerSignal()

```
virtual int32_t SamplingProvider::registerSignal ( )  [inline], [virtual]
```

Definition at line 22 of file sampling_provider.h.

6.302.3.3 requestedInstrumentation()

```
virtual  InstrumentLevel::Level SamplingProvider::requestedInstrumentation ( )  [pure virtual]
```

Implemented in **InstrCountSampling** (p. 620).

6.302.3.4 startSampling()

```
virtual void SamplingProvider::startSampling (
              SubsecondTime until )  [pure virtual]
```

Implemented in **InstrCountSampling** (p. 620).

Referenced by SamplingManager::enableFastForward().

The documentation for this class was generated from the following files:

- common/sampling/ **sampling_provider.h**
- common/sampling/ **sampling_provider.cc**

6.303 SaturatingPredictor< n > Class Template Reference

```
#include <saturating_predictor.h>
```

Public Member Functions

- **SaturatingPredictor** (**UInt32** initial_value)
- bool **predict** ()
- void **reset** (bool prediction=0)
- void **update** (bool actual)
- **SaturatingPredictor** & **operator++** ()
- **SaturatingPredictor** & **operator--** ()

Private Attributes

- int8_t **m_counter**

6.303.1 Detailed Description

template<unsigned n>
class SaturatingPredictor< n >

Definition at line 23 of file saturating_predictor.h.

6.303.2 Constructor & Destructor Documentation

6.303.2.1 SaturatingPredictor()

```
template<unsigned n>
SaturatingPredictor< n >:: SaturatingPredictor (
              UInt32 initial_value ) [inline]
```

Definition at line 27 of file saturating_predictor.h.

References SaturatingPredictor< n >::m_counter.

6.303.3 Member Function Documentation

6.303.3.1 operator++()

```
template<unsigned n>
SaturatingPredictor& SaturatingPredictor< n >::operator++ ( ) [inline]
```

Definition at line 68 of file saturating_predictor.h.

References SaturatingPredictor< n >::m_counter.

6.303.3.2 operator--()

```
template<unsigned n>
SaturatingPredictor&  SaturatingPredictor<  n >::operator-- ( )  [inline]
```

Definition at line 84 of file saturating_predictor.h.

References SaturatingPredictor< n >::m_counter.

6.303.3.3 predict()

```
template<unsigned n>
bool  SaturatingPredictor<  n >::predict ( )  [inline]
```

Definition at line 32 of file saturating_predictor.h.

References SaturatingPredictor< n >::m_counter.

6.303.3.4 reset()

```
template<unsigned n>
void  SaturatingPredictor<  n >::reset (
            bool prediction = 0 )  [inline]
```

Definition at line 37 of file saturating_predictor.h.

References SaturatingPredictor< n >::m_counter, and n.

6.303.3.5 update()

```
template<unsigned n>
void  SaturatingPredictor<  n >::update (
            bool actual )  [inline]
```

Definition at line 54 of file saturating_predictor.h.

6.303.4 Member Data Documentation

6.303.4.1 m_counter

```
template<unsigned n>
int8_t SaturatingPredictor< n >::m_counter [private]
```

Definition at line 102 of file saturating_predictor.h.

Referenced by SaturatingPredictor< n >::operator++(), SaturatingPredictor< n >::operator--(), SaturatingPredictor< n >::predict(), SaturatingPredictor< n >::reset(), and SaturatingPredictor< n >::SaturatingPredictor().

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **saturating_predictor.h**

6.304 config::SaveError Class Reference

```
#include <config_exceptions.hpp>
```

Inheritance diagram for config::SaveError:

Public Member Functions

- **SaveError** (const String &error_str)
- virtual ~**SaveError** () throw ()
- virtual const char * **what** () const throw ()

Private Attributes

- String **m_error**

6.304.1 Detailed Description

Definition at line 63 of file config_exceptions.hpp.

6.304.2 Constructor & Destructor Documentation

6.304.2.1 SaveError()

```
config::SaveError::SaveError (
            const String & error_str )  [inline]
```

Definition at line 66 of file config_exceptions.hpp.

6.304.2.2 ~SaveError()

```
virtual config::SaveError::~SaveError ( ) throw ( )   [inline], [virtual]
```

Definition at line 72 of file config_exceptions.hpp.

6.304.3 Member Function Documentation

6.304.3.1 what()

```
virtual const char* config::SaveError::what ( ) const throw ( )   [inline], [virtual]
```

Definition at line 74 of file config_exceptions.hpp.

References m_error.

6.304.4 Member Data Documentation

6.304.4.1 m_error

```
String config::SaveError::m_error  [private]
```

Definition at line 80 of file config_exceptions.hpp.

Referenced by what().

The documentation for this class was generated from the following file:

- common/config/ **config_exceptions.hpp**

6.305 Scheduler Class Reference

`#include <scheduler.h>`

Inheritance diagram for Scheduler:

Public Member Functions

- **Scheduler** (**ThreadManager** ∗thread_manager)
- virtual ∼**Scheduler** ()
- virtual **core_id_t threadCreate** (**thread_id_t** thread_id)=0
- virtual void **threadYield** (**thread_id_t** thread_id)
- virtual bool **threadSetAffinity** (**thread_id_t** calling_thread_id, **thread_id_t** thread_id, size_t cpusetsize, const cpu_set_t ∗mask)
- virtual bool **threadGetAffinity** (**thread_id_t** thread_id, size_t cpusetsize, cpu_set_t ∗mask)

Static Public Member Functions

- static **Scheduler** ∗ **create** (**ThreadManager** ∗thread_manager)

Protected Member Functions

- **core_id_t findFirstFreeCore** ()
- void **printMapping** ()

Protected Attributes

- **ThreadManager** ∗ **m_thread_manager**

6.305.1 Detailed Description

Definition at line 7 of file scheduler.h.

6.305.2 Constructor & Destructor Documentation

6.305.2.1 Scheduler()

```
Scheduler::Scheduler (
            ThreadManager * thread_manager )
```

Definition at line 31 of file scheduler.cc.

6.305.2.2 ~Scheduler()

```
virtual Scheduler::~Scheduler ( )  [inline], [virtual]
```

Definition at line 13 of file scheduler.h.

6.305.3 Member Function Documentation

6.305.3.1 create()

```
Scheduler * Scheduler::create (
            ThreadManager * thread_manager )  [static]
```

Definition at line 13 of file scheduler.cc.

References LOG_PRINT_ERROR.

6.305.3.2 findFirstFreeCore()

```
core_id_t Scheduler::findFirstFreeCore ( )  [protected]
```

Definition at line 36 of file scheduler.cc.

References Core::IDLE, and INVALID_CORE_ID.

6.305.3.3 printMapping()

```
void Scheduler::printMapping ( )  [protected]
```

Definition at line 49 of file scheduler.cc.

References Core::BROKEN, Core::IDLE, Core::INITIALIZING, Core::NUM_STATES, Core::RUNNING, Core::SL↩
EEPING, Core::STALLED, and Core::WAKING_UP.

6.305.3.4 threadCreate()

```
virtual  core_id_t Scheduler::threadCreate (
              thread_id_t thread_id )  [pure virtual]
```

Implemented in **SchedulerDynamic** (p. 1135), **SchedulerStatic** (p. 1156), and **SchedulerPinnedBase** (p. 1142).

Referenced by ThreadManager::createThread_unlocked().

6.305.3.5 threadGetAffinity()

```
virtual bool Scheduler::threadGetAffinity (
              thread_id_t thread_id,
             size_t cpusetsize,
             cpu_set_t * mask )  [inline], [virtual]
```

Reimplemented in **SchedulerPinnedBase** (p. 1142).

Definition at line 18 of file scheduler.h.

6.305.3.6 threadSetAffinity()

```
virtual bool Scheduler::threadSetAffinity (
              thread_id_t calling_thread_id,
              thread_id_t thread_id,
             size_t cpusetsize,
             const cpu_set_t * mask )  [inline], [virtual]
```

Reimplemented in **SchedulerPinnedBase** (p. 1143).

Definition at line 17 of file scheduler.h.

6.305.3.7 threadYield()

```
virtual void Scheduler::threadYield (
              thread_id_t thread_id )  [inline], [virtual]
```

Reimplemented in **SchedulerPinnedBase** (p. 1144).

Definition at line 16 of file scheduler.h.

6.305.4 Member Data Documentation

6.305.4.1 m_thread_manager

```
ThreadManager* Scheduler::m_thread_manager  [protected]
```

Definition at line 21 of file scheduler.h.

Referenced by SchedulerDynamic::moveThread().

The documentation for this class was generated from the following files:

- common/scheduler/ **scheduler.h**
- common/scheduler/ **scheduler.cc**

6.306 SchedulerBigSmall Class Reference

```
#include <scheduler_big_small.h>
```

Inheritance diagram for SchedulerBigSmall:

Public Member Functions

- **SchedulerBigSmall** (**ThreadManager** *thread_manager)
- virtual void **threadSetInitialAffinity** (**thread_id_t** thread_id)
- virtual void **threadStall** (**thread_id_t** thread_id, **ThreadManager::stall_type_t** reason, **SubsecondTime** time)
- virtual void **threadExit** (**thread_id_t** thread_id, **SubsecondTime** time)
- virtual void **periodic** (**SubsecondTime** time)

Private Member Functions

- void **moveToBig** (**thread_id_t** thread_id)
- void **moveToSmall** (**thread_id_t** thread_id)
- void **pickBigThread** ()

Private Attributes

- const bool **m_debug_output**
- **UInt64 m_num_big_cores**
- cpu_set_t **m_mask_big**
- cpu_set_t **m_mask_small**
- **SubsecondTime m_last_reshuffle**
- **UInt64 m_rng**
- std::unordered_map< **thread_id_t**, bool > **m_thread_isbig**

Additional Inherited Members

6.306.1 Detailed Description

Definition at line 8 of file scheduler_big_small.h.

6.306.2 Constructor & Destructor Documentation

6.306.2.1 SchedulerBigSmall()

```
SchedulerBigSmall::SchedulerBigSmall (
            ThreadManager * thread_manager )
```

Definition at line 22 of file scheduler_big_small.cc.

References m_mask_big, m_mask_small, and m_num_big_cores.

6.306.3 Member Function Documentation

6.306.3.1 moveToBig()

```
void SchedulerBigSmall::moveToBig (
            thread_id_t thread_id )  [private]
```

Definition at line 132 of file scheduler_big_small.cc.

References INVALID_THREAD_ID, m_mask_big, m_thread_isbig, and SchedulerPinnedBase::threadSetAffinity().

Referenced by pickBigThread().

6.306.3.2 moveToSmall()

```
void SchedulerBigSmall::moveToSmall (
            thread_id_t thread_id ) [private]
```

Definition at line 126 of file scheduler_big_small.cc.

References INVALID_THREAD_ID, m_mask_small, m_thread_isbig, and SchedulerPinnedBase::threadSetAffinity().

Referenced by periodic(), threadSetInitialAffinity(), and threadStall().

6.306.3.3 periodic()

```
void SchedulerBigSmall::periodic (
            SubsecondTime time ) [virtual]
```

Reimplemented from **SchedulerPinnedBase** (p. 1141).

Definition at line 97 of file scheduler_big_small.cc.

References m_debug_output, m_last_reshuffle, m_num_big_cores, SchedulerPinnedBase::m_quantum, m_thread_isbig, moveToSmall(), SchedulerPinnedBase::periodic(), pickBigThread(), and SchedulerPinnedBase::printState().

6.306.3.4 pickBigThread()

```
void SchedulerBigSmall::pickBigThread ( ) [private]
```

Definition at line 138 of file scheduler_big_small.cc.

References m_debug_output, m_rng, m_thread_isbig, SchedulerDynamic::m_threads_runnable, moveToBig(), and rng_next().

Referenced by periodic(), threadExit(), and threadStall().

6.306.3.5 threadExit()

```
void SchedulerBigSmall::threadExit (
            thread_id_t thread_id,
            SubsecondTime time ) [virtual]
```

Reimplemented from **SchedulerPinnedBase** (p. 1142).

Definition at line 77 of file scheduler_big_small.cc.

References m_debug_output, m_thread_isbig, pickBigThread(), SchedulerPinnedBase::printState(), and SchedulerPinnedBase::threadExit().

6.306.3.6 threadSetInitialAffinity()

```
void SchedulerBigSmall::threadSetInitialAffinity (
              thread_id_t thread_id )  [virtual]
```

Implements **SchedulerPinnedBase** (p. 1143).

Definition at line 49 of file scheduler_big_small.cc.

References moveToSmall().

6.306.3.7 threadStall()

```
void SchedulerBigSmall::threadStall (
              thread_id_t thread_id,
              ThreadManager::stall_type_t reason,
              SubsecondTime time )  [virtual]
```

Reimplemented from **SchedulerPinnedBase** (p. 1144).

Definition at line 55 of file scheduler_big_small.cc.

References m_debug_output, m_thread_isbig, moveToSmall(), pickBigThread(), SchedulerPinnedBase::print↩State(), and SchedulerPinnedBase::threadStall().

6.306.4 Member Data Documentation

6.306.4.1 m_debug_output

```
const bool SchedulerBigSmall::m_debug_output  [private]
```

Definition at line 19 of file scheduler_big_small.h.

Referenced by periodic(), pickBigThread(), threadExit(), and threadStall().

6.306.4.2 m_last_reshuffle

```
SubsecondTime SchedulerBigSmall::m_last_reshuffle  [private]
```

Definition at line 26 of file scheduler_big_small.h.

Referenced by periodic().

6.306.4.3 m_mask_big

```
cpu_set_t SchedulerBigSmall::m_mask_big  [private]
```

Definition at line 23 of file scheduler_big_small.h.

Referenced by moveToBig(), and SchedulerBigSmall().

6.306.4.4 m_mask_small

```
cpu_set_t SchedulerBigSmall::m_mask_small  [private]
```

Definition at line 24 of file scheduler_big_small.h.

Referenced by moveToSmall(), and SchedulerBigSmall().

6.306.4.5 m_num_big_cores

```
UInt64 SchedulerBigSmall::m_num_big_cores  [private]
```

Definition at line 22 of file scheduler_big_small.h.

Referenced by periodic(), and SchedulerBigSmall().

6.306.4.6 m_rng

```
UInt64 SchedulerBigSmall::m_rng  [private]
```

Definition at line 27 of file scheduler_big_small.h.

Referenced by pickBigThread().

6.306.4.7 m_thread_isbig

```
std::unordered_map< thread_id_t, bool> SchedulerBigSmall::m_thread_isbig  [private]
```

Definition at line 28 of file scheduler_big_small.h.

Referenced by moveToBig(), moveToSmall(), periodic(), pickBigThread(), threadExit(), and threadStall().

The documentation for this class was generated from the following files:

- common/scheduler/ **scheduler_big_small.h**
- common/scheduler/ **scheduler_big_small.cc**

6.307 SchedulerDynamic Class Reference

`#include <scheduler_dynamic.h>`

Inheritance diagram for SchedulerDynamic:

Public Member Functions

- **SchedulerDynamic** (**ThreadManager** *thread_manager)
- virtual ~**SchedulerDynamic** ()
- virtual **core_id_t** **threadCreate** (**thread_id_t**)=0
- virtual void **periodic** (**SubsecondTime** time)
- virtual void **threadStart** (**thread_id_t** thread_id, **SubsecondTime** time)
- virtual void **threadStall** (**thread_id_t** thread_id, **ThreadManager::stall_type_t** reason, **SubsecondTime** time)
- virtual void **threadResume** (**thread_id_t** thread_id, **thread_id_t** thread_by, **SubsecondTime** time)
- virtual void **threadExit** (**thread_id_t** thread_id, **SubsecondTime** time)

Protected Member Functions

- void **moveThread** (**thread_id_t** thread_id, **core_id_t** core_id, **SubsecondTime** time)

Protected Attributes

- std::vector< bool > **m_threads_runnable**

Private Member Functions

- void **__periodic** (**SubsecondTime** time)
- void **__roi_begin** ()
- void **__roi_end** ()
- void **__threadStart** (**thread_id_t** thread_id, **SubsecondTime** time)
- void **__threadStall** (**thread_id_t** thread_id, **ThreadManager::stall_type_t** reason, **SubsecondTime** time)
- void **__threadResume** (**thread_id_t** thread_id, **thread_id_t** thread_by, **SubsecondTime** time)
- void **__threadExit** (**thread_id_t** thread_id, **SubsecondTime** time)

Static Private Member Functions

- static **SInt64 hook_periodic** (**UInt64** ptr, **UInt64** time)
- static **SInt64 hook_thread_start** (**UInt64** ptr, **UInt64** _args)
- static **SInt64 hook_thread_stall** (**UInt64** ptr, **UInt64** _args)
- static **SInt64 hook_thread_resume** (**UInt64** ptr, **UInt64** _args)
- static **SInt64 hook_thread_exit** (**UInt64** ptr, **UInt64** _args)

Private Attributes

- bool **m_in_periodic**

Additional Inherited Members

6.307.1 Detailed Description

Definition at line 9 of file scheduler_dynamic.h.

6.307.2 Constructor & Destructor Documentation

6.307.2.1 SchedulerDynamic()

```
SchedulerDynamic::SchedulerDynamic (
            ThreadManager * thread_manager )
```

Definition at line 6 of file scheduler_dynamic.cc.

References HookType::HOOK_PERIODIC, hook_periodic(), HookType::HOOK_THREAD_EXIT, hook_thread_exit(), HookType::HOOK_THREAD_RESUME, hook_thread_resume(), HookType::HOOK_THREAD_STALL, hook_thread_stall(), HookType::HOOK_THREAD_START, hook_thread_start(), and HooksManager::ORDER_ACTION.

6.307.2.2 ~SchedulerDynamic()

```
SchedulerDynamic::~SchedulerDynamic ( )  [virtual]
```

Definition at line 18 of file scheduler_dynamic.cc.

6.307.3 Member Function Documentation

6.307.3.1 __periodic()

```
void SchedulerDynamic::__periodic (
              SubsecondTime time )  [private]
```

Definition at line 22 of file scheduler_dynamic.cc.

References m_in_periodic, and periodic().

6.307.3.2 __roi_begin()

```
void SchedulerDynamic::__roi_begin ( )  [private]
```

6.307.3.3 __roi_end()

```
void SchedulerDynamic::__roi_end ( )  [private]
```

6.307.3.4 __threadExit()

```
void SchedulerDynamic::__threadExit (
              thread_id_t thread_id,
              SubsecondTime time )  [private]
```

Definition at line 55 of file scheduler_dynamic.cc.

References m_threads_runnable, and threadExit().

6.307.3.5 __threadResume()

```
void SchedulerDynamic::__threadResume (
              thread_id_t thread_id,
              thread_id_t thread_by,
              SubsecondTime time )  [private]
```

Definition at line 49 of file scheduler_dynamic.cc.

References m_threads_runnable, and threadResume().

6.307.3.6 __threadStall()

```
void SchedulerDynamic::__threadStall (
              thread_id_t thread_id,
              ThreadManager::stall_type_t reason,
              SubsecondTime time )  [private]
```

Definition at line 40 of file scheduler_dynamic.cc.

References m_threads_runnable, ThreadManager::STALL_UNSCHEDULED, and threadStall().

6.307.3.7 __threadStart()

```
void SchedulerDynamic::__threadStart (
              thread_id_t thread_id,
              SubsecondTime time )  [private]
```

Definition at line 31 of file scheduler_dynamic.cc.

References m_threads_runnable, and threadStart().

6.307.3.8 hook_periodic()

```
static  SInt64 SchedulerDynamic::hook_periodic (
              UInt64 ptr,
              UInt64 time )  [inline], [static], [private]
```

Definition at line 39 of file scheduler_dynamic.h.

Referenced by SchedulerDynamic().

6.307.3.9 hook_thread_exit()

```
static  SInt64 SchedulerDynamic::hook_thread_exit (
              UInt64 ptr,
              UInt64 _args )  [inline], [static], [private]
```

Definition at line 59 of file scheduler_dynamic.h.

References HooksManager::ThreadTime::thread_id, and HooksManager::ThreadTime::time.

Referenced by SchedulerDynamic().

6.307.3.10 hook_thread_resume()

```
static SInt64 SchedulerDynamic::hook_thread_resume (
              UInt64 ptr,
              UInt64 _args ) [inline], [static], [private]
```

Definition at line 53 of file scheduler_dynamic.h.

References HooksManager::ThreadResume::thread_by, HooksManager::ThreadResume::thread_id, and HooksManager::ThreadResume::time.

Referenced by SchedulerDynamic().

6.307.3.11 hook_thread_stall()

```
static SInt64 SchedulerDynamic::hook_thread_stall (
              UInt64 ptr,
              UInt64 _args ) [inline], [static], [private]
```

Definition at line 47 of file scheduler_dynamic.h.

References HooksManager::ThreadStall::reason, HooksManager::ThreadStall::thread_id, and HooksManager::ThreadStall::time.

Referenced by SchedulerDynamic().

6.307.3.12 hook_thread_start()

```
static SInt64 SchedulerDynamic::hook_thread_start (
              UInt64 ptr,
              UInt64 _args ) [inline], [static], [private]
```

Definition at line 41 of file scheduler_dynamic.h.

References HooksManager::ThreadTime::thread_id, and HooksManager::ThreadTime::time.

Referenced by SchedulerDynamic().

6.307.3.13 moveThread()

```
void SchedulerDynamic::moveThread (
              thread_id_t thread_id,
              core_id_t core_id,
              SubsecondTime time ) [protected]
```

Definition at line 61 of file scheduler_dynamic.cc.

References LOG_ASSERT_ERROR, m_in_periodic, Scheduler::m_thread_manager, m_threads_runnable, and ThreadManager::moveThread().

Referenced by SchedulerPinnedBase::reschedule().

6.307.3.14 periodic()

```
virtual void SchedulerDynamic::periodic (
              SubsecondTime time )  [inline], [virtual]
```

Reimplemented in **SchedulerPinnedBase** (p. 1141), and **SchedulerBigSmall** (p. 1127).

Definition at line 16 of file scheduler_dynamic.h.

Referenced by __periodic().

6.307.3.15 threadCreate()

```
virtual  core_id_t SchedulerDynamic::threadCreate (
              thread_id_t  )  [pure virtual]
```

Implements **Scheduler** (p. 1123).

Implemented in **SchedulerPinnedBase** (p. 1142).

6.307.3.16 threadExit()

```
virtual void SchedulerDynamic::threadExit (
              thread_id_t thread_id,
              SubsecondTime time )  [inline], [virtual]
```

Reimplemented in **SchedulerSequential** (p. 1150), **SchedulerPinnedBase** (p. 1142), and **SchedulerBigSmall** (p. 1127).

Definition at line 20 of file scheduler_dynamic.h.

Referenced by __threadExit().

6.307.3.17 threadResume()

```
virtual void SchedulerDynamic::threadResume (
              thread_id_t thread_id,
              thread_id_t thread_by,
              SubsecondTime time )  [inline], [virtual]
```

Reimplemented in **SchedulerPinnedBase** (p. 1143).

Definition at line 19 of file scheduler_dynamic.h.

Referenced by __threadResume().

6.307.3.18 threadStall()

```
virtual void SchedulerDynamic::threadStall (
              thread_id_t thread_id,
              ThreadManager::stall_type_t reason,
              SubsecondTime time ) [inline], [virtual]
```

Reimplemented in **SchedulerPinnedBase** (p. 1144), and **SchedulerBigSmall** (p. 1128).

Definition at line 18 of file scheduler_dynamic.h.

Referenced by __threadStall().

6.307.3.19 threadStart()

```
virtual void SchedulerDynamic::threadStart (
              thread_id_t thread_id,
              SubsecondTime time ) [inline], [virtual]
```

Reimplemented in **SchedulerSequential** (p. 1151), and **SchedulerPinnedBase** (p. 1144).

Definition at line 17 of file scheduler_dynamic.h.

Referenced by __threadStart().

6.307.4 Member Data Documentation

6.307.4.1 m_in_periodic

```
bool SchedulerDynamic::m_in_periodic [private]
```

Definition at line 28 of file scheduler_dynamic.h.

Referenced by __periodic(), and moveThread().

6.307.4.2 m_threads_runnable

```
std::vector<bool> SchedulerDynamic::m_threads_runnable [protected]
```

Definition at line 23 of file scheduler_dynamic.h.

Referenced by __threadExit(), __threadResume(), __threadStall(), __threadStart(), moveThread(), Scheduler↩BigSmall::pickBigThread(), SchedulerPinnedBase::printState(), SchedulerPinnedBase::reschedule(), Scheduler↩Sequential::threadExit(), SchedulerPinnedBase::threadSetAffinity(), and SchedulerSequential::threadStart().

The documentation for this class was generated from the following files:

- common/scheduler/ **scheduler_dynamic.h**
- common/scheduler/ **scheduler_dynamic.cc**

6.308 SchedulerPinned Class Reference

`#include <scheduler_pinned.h>`

Inheritance diagram for SchedulerPinned:

Public Member Functions

- **SchedulerPinned** (**ThreadManager** *thread_manager)
- virtual void **threadSetInitialAffinity** (**thread_id_t** thread_id)

Private Member Functions

- **core_id_t getNextCore** (**core_id_t** core_first)
- **core_id_t getFreeCore** (**core_id_t** core_first)

Private Attributes

- const int **m_interleaving**
- std::vector< bool > **m_core_mask**
- **core_id_t m_next_core**

Additional Inherited Members

6.308.1 Detailed Description

Definition at line 6 of file scheduler_pinned.h.

6.308.2 Constructor & Destructor Documentation

6.308.2.1 SchedulerPinned()

```
SchedulerPinned::SchedulerPinned (
            ThreadManager * thread_manager )
```

Definition at line 4 of file scheduler_pinned.cc.

References m_core_mask.

6.308.3 Member Function Documentation

6.308.3.1 getFreeCore()

```
core_id_t SchedulerPinned::getFreeCore (
            core_id_t core_first )  [private]
```

Definition at line 33 of file scheduler_pinned.cc.

References getNextCore(), INVALID_THREAD_ID, m_core_mask, and SchedulerPinnedBase::m_core_thread_↩ running.

Referenced by threadSetInitialAffinity().

6.308.3.2 getNextCore()

```
core_id_t SchedulerPinned::getNextCore (
            core_id_t core_first )  [private]
```

Definition at line 17 of file scheduler_pinned.cc.

References m_core_mask, and m_interleaving.

Referenced by getFreeCore(), and threadSetInitialAffinity().

6.308.3.3 threadSetInitialAffinity()

```
void SchedulerPinned::threadSetInitialAffinity (
            thread_id_t thread_id )  [virtual]
```

Implements **SchedulerPinnedBase** (p. 1143).

Definition at line 49 of file scheduler_pinned.cc.

References getFreeCore(), getNextCore(), m_next_core, and SchedulerPinnedBase::m_thread_info.

6.308.4 Member Data Documentation

6.308.4.1 m_core_mask

```
std::vector<bool> SchedulerPinned::m_core_mask  [private]
```

Definition at line 18 of file scheduler_pinned.h.

Referenced by getFreeCore(), getNextCore(), and SchedulerPinned().

6.308.4.2 m_interleaving

```
const int SchedulerPinned::m_interleaving  [private]
```

Definition at line 17 of file scheduler_pinned.h.

Referenced by getNextCore().

6.308.4.3 m_next_core

```
core_id_t SchedulerPinned::m_next_core  [private]
```

Definition at line 20 of file scheduler_pinned.h.

Referenced by threadSetInitialAffinity().

The documentation for this class was generated from the following files:

- common/scheduler/ **scheduler_pinned.h**
- common/scheduler/ **scheduler_pinned.cc**

6.309 SchedulerPinnedBase Class Reference

```
#include <scheduler_pinned_base.h>
```

Inheritance diagram for SchedulerPinnedBase:

Classes

- class **ThreadInfo**

Public Member Functions

- **SchedulerPinnedBase** (**ThreadManager** ∗thread_manager, **SubsecondTime** quantum)
- virtual **core_id_t threadCreate** (**thread_id_t**)
- virtual void **threadYield** (**thread_id_t** thread_id)
- virtual bool **threadSetAffinity** (**thread_id_t** calling_thread_id, **thread_id_t** thread_id, size_t cpusetsize, const cpu_set_t ∗mask)
- virtual bool **threadGetAffinity** (**thread_id_t** thread_id, size_t cpusetsize, cpu_set_t ∗mask)
- virtual void **periodic** (**SubsecondTime** time)
- virtual void **threadStart** (**thread_id_t** thread_id, **SubsecondTime** time)
- virtual void **threadStall** (**thread_id_t** thread_id, **ThreadManager::stall_type_t** reason, **SubsecondTime** time)
- virtual void **threadResume** (**thread_id_t** thread_id, **thread_id_t** thread_by, **SubsecondTime** time)
- virtual void **threadExit** (**thread_id_t** thread_id, **SubsecondTime** time)

Protected Member Functions

- virtual void **threadSetInitialAffinity** (**thread_id_t** thread_id)=0
- **core_id_t findFreeCoreForThread** (**thread_id_t** thread_id)
- void **reschedule** (**SubsecondTime** time, **core_id_t** core_id, bool is_periodic)
- void **printState** ()

Protected Attributes

- const **SubsecondTime m_quantum**
- **SubsecondTime m_last_periodic**
- std::vector< **ThreadInfo** > **m_thread_info**
- std::vector< **thread_id_t** > **m_core_thread_running**
- std::vector< **SubsecondTime** > **m_quantum_left**

Additional Inherited Members

6.309.1 Detailed Description

Definition at line 7 of file scheduler_pinned_base.h.

6.309.2 Constructor & Destructor Documentation

6.309.2.1 SchedulerPinnedBase()

```
SchedulerPinnedBase::SchedulerPinnedBase (
            ThreadManager * thread_manager,
            SubsecondTime quantum )
```

Definition at line 14 of file scheduler_pinned_base.cc.

6.309.3 Member Function Documentation

6.309.3.1 findFreeCoreForThread()

```
core_id_t SchedulerPinnedBase::findFreeCoreForThread (
            thread_id_t thread_id )  [protected]
```

Definition at line 23 of file scheduler_pinned_base.cc.

References INVALID_CORE_ID, INVALID_THREAD_ID, m_core_thread_running, and m_thread_info.

Referenced by threadCreate(), threadResume(), threadSetAffinity(), threadStart(), and threadYield().

6.309.3.2 periodic()

```
void SchedulerPinnedBase::periodic (
            SubsecondTime time )  [virtual]
```

Reimplemented from **SchedulerDynamic** (p. 1134).

Reimplemented in **SchedulerBigSmall** (p. 1127).

Definition at line 202 of file scheduler_pinned_base.cc.

References INVALID_THREAD_ID, m_core_thread_running, m_last_periodic, m_quantum_left, and reschedule().

Referenced by SchedulerBigSmall::periodic().

6.309.3.3 printState()

```
void SchedulerPinnedBase::printState ( )  [protected]
```

Definition at line 308 of file scheduler_pinned_base.cc.

References Core::BROKEN, Core::IDLE, Core::INITIALIZING, INVALID_THREAD_ID, m_core_thread_running, m_thread_info, SchedulerDynamic::m_threads_runnable, Core::NUM_STATES, Core::RUNNING, Core::SLEEP↩ING, Core::STALLED, and Core::WAKING_UP.

Referenced by SchedulerBigSmall::periodic(), SchedulerBigSmall::threadExit(), and SchedulerBigSmall::thread↩Stall().

6.309.3.4 reschedule()

```
void SchedulerPinnedBase::reschedule (
            SubsecondTime time,
            core_id_t core_id,
            bool is_periodic ) [protected]
```

Definition at line 221 of file scheduler_pinned_base.cc.

References SubsecondTime::getPS(), Core::INITIALIZING, INVALID_CORE_ID, INVALID_THREAD_ID, m_↩ core_thread_running, m_quantum, m_quantum_left, m_thread_info, SchedulerDynamic::m_threads_runnable, SchedulerDynamic::moveThread(), and SubsecondTime::PS().

Referenced by periodic(), threadExit(), SchedulerSequential::threadExit(), threadResume(), threadSetAffinity(), threadStall(), threadStart(), and threadYield().

6.309.3.5 threadCreate()

```
core_id_t SchedulerPinnedBase::threadCreate (
            thread_id_t thread_id ) [virtual]
```

Implements **SchedulerDynamic** (p. 1135).

Definition at line 35 of file scheduler_pinned_base.cc.

References findFreeCoreForThread(), INVALID_CORE_ID, m_core_thread_running, m_quantum, m_quantum_↩ left, m_thread_info, and threadSetInitialAffinity().

6.309.3.6 threadExit()

```
void SchedulerPinnedBase::threadExit (
            thread_id_t thread_id,
            SubsecondTime time ) [virtual]
```

Reimplemented from **SchedulerDynamic** (p. 1135).

Reimplemented in **SchedulerSequential** (p. 1150), and **SchedulerBigSmall** (p. 1127).

Definition at line 195 of file scheduler_pinned_base.cc.

References m_thread_info, and reschedule().

Referenced by SchedulerBigSmall::threadExit().

6.309.3.7 threadGetAffinity()

```
bool SchedulerPinnedBase::threadGetAffinity (
              thread_id_t thread_id,
             size_t cpusetsize,
             cpu_set_t * mask )  [virtual]
```

Reimplemented from **Scheduler** (p. 1124).

Definition at line 149 of file scheduler_pinned_base.cc.

References CPU_SET_S, CPU_ZERO_S, and m_thread_info.

6.309.3.8 threadResume()

```
void SchedulerPinnedBase::threadResume (
              thread_id_t thread_id,
              thread_id_t thread_by,
              SubsecondTime time )  [virtual]
```

Reimplemented from **SchedulerDynamic** (p. 1135).

Definition at line 187 of file scheduler_pinned_base.cc.

References findFreeCoreForThread(), INVALID_THREAD_ID, and reschedule().

6.309.3.9 threadSetAffinity()

```
bool SchedulerPinnedBase::threadSetAffinity (
              thread_id_t calling_thread_id,
              thread_id_t thread_id,
             size_t cpusetsize,
             const cpu_set_t * mask )  [virtual]
```

Reimplemented from **Scheduler** (p. 1124).

Definition at line 89 of file scheduler_pinned_base.cc.

References CPU_ISSET_S, findFreeCoreForThread(), PerformanceModel::getElapsedTime(), Core::get↩ PerformanceModel(), INVALID_THREAD_ID, LOG_ASSERT_ERROR, m_quantum_left, m_thread_info, SchedulerDynamic::m_threads_runnable, reschedule(), threadYield(), and SubsecondTime::Zero().

Referenced by SchedulerBigSmall::moveToBig(), and SchedulerBigSmall::moveToSmall().

6.309.3.10 threadSetInitialAffinity()

```
virtual void SchedulerPinnedBase::threadSetInitialAffinity (
              thread_id_t thread_id )  [protected], [pure virtual]
```

Implemented in **SchedulerSequential** (p. 1151), **SchedulerBigSmall** (p. 1127), **SchedulerPinned** (p. 1138), and **SchedulerRoaming** (p. 1147).

Referenced by threadCreate().

6.309.3.11 threadStall()

```
void SchedulerPinnedBase::threadStall (
              thread_id_t thread_id,
              ThreadManager::stall_type_t reason,
              SubsecondTime time )  [virtual]
```

Reimplemented from **SchedulerDynamic** (p. 1135).

Reimplemented in **SchedulerBigSmall** (p. 1128).

Definition at line 180 of file scheduler_pinned_base.cc.

References m_thread_info, and reschedule().

Referenced by SchedulerBigSmall::threadStall().

6.309.3.12 threadStart()

```
void SchedulerPinnedBase::threadStart (
              thread_id_t thread_id,
              SubsecondTime time )  [virtual]
```

Reimplemented from **SchedulerDynamic** (p. 1136).

Reimplemented in **SchedulerSequential** (p. 1151).

Definition at line 172 of file scheduler_pinned_base.cc.

References findFreeCoreForThread(), INVALID_THREAD_ID, and reschedule().

6.309.3.13 threadYield()

```
void SchedulerPinnedBase::threadYield (
              thread_id_t thread_id )  [virtual]
```

Reimplemented from **Scheduler** (p. 1124).

Definition at line 65 of file scheduler_pinned_base.cc.

References findFreeCoreForThread(), PerformanceModel::getElapsedTime(), Core::getPerformanceModel(), IN↩ VALID_CORE_ID, m_quantum_left, m_thread_info, reschedule(), and SubsecondTime::Zero().

Referenced by threadSetAffinity().

6.309.4 Member Data Documentation

6.309.4.1 m_core_thread_running

```
std::vector< thread_id_t> SchedulerPinnedBase::m_core_thread_running  [protected]
```

Definition at line 77 of file scheduler_pinned_base.h.

Referenced by findFreeCoreForThread(), SchedulerPinned::getFreeCore(), periodic(), printState(), reschedule(), threadCreate(), and SchedulerSequential::threadStart().

6.309.4.2 m_last_periodic

```
SubsecondTime SchedulerPinnedBase::m_last_periodic  [protected]
```

Definition at line 73 of file scheduler_pinned_base.h.

Referenced by periodic().

6.309.4.3 m_quantum

```
const  SubsecondTime SchedulerPinnedBase::m_quantum  [protected]
```

Definition at line 71 of file scheduler_pinned_base.h.

Referenced by SchedulerBigSmall::periodic(), reschedule(), and threadCreate().

6.309.4.4 m_quantum_left

```
std::vector< SubsecondTime> SchedulerPinnedBase::m_quantum_left  [protected]
```

Definition at line 78 of file scheduler_pinned_base.h.

Referenced by periodic(), reschedule(), threadCreate(), threadSetAffinity(), and threadYield().

6.309.4.5 m_thread_info

```
std::vector< ThreadInfo> SchedulerPinnedBase::m_thread_info  [protected]
```

Definition at line 75 of file scheduler_pinned_base.h.

Referenced by findFreeCoreForThread(), printState(), reschedule(), SchedulerSequential::results_on_screen(), threadCreate(), threadExit(), SchedulerSequential::threadExit(), threadGetAffinity(), threadSetAffinity(), SchedulerRoaming::threadSetInitialAffinity(), SchedulerPinned::threadSetInitialAffinity(), SchedulerSequential::threadSetInitialAffinity(), threadStall(), SchedulerSequential::threadStart(), and threadYield().

The documentation for this class was generated from the following files:

- common/scheduler/ **scheduler_pinned_base.h**
- common/scheduler/ **scheduler_pinned_base.cc**

6.310 SchedulerRoaming Class Reference

```
#include <scheduler_roaming.h>
```

Inheritance diagram for SchedulerRoaming:

Public Member Functions

- **SchedulerRoaming** (**ThreadManager** *thread_manager)
- virtual void **threadSetInitialAffinity** (**thread_id_t** thread_id)

Private Attributes

- std::vector< bool > **m_core_mask**

Additional Inherited Members

6.310.1 Detailed Description

Definition at line 6 of file scheduler_roaming.h.

6.310.2 Constructor & Destructor Documentation

6.310.2.1 SchedulerRoaming()

```
SchedulerRoaming::SchedulerRoaming (
            ThreadManager * thread_manager )
```

Definition at line 4 of file scheduler_roaming.cc.

References m_core_mask.

6.310.3 Member Function Documentation

6.310.3.1 threadSetInitialAffinity()

```
void SchedulerRoaming::threadSetInitialAffinity (
            thread_id_t thread_id )  [virtual]
```

Implements **SchedulerPinnedBase** (p. 1143).

Definition at line 15 of file scheduler_roaming.cc.

References m_core_mask, and SchedulerPinnedBase::m_thread_info.

6.310.4 Member Data Documentation

6.310.4.1 m_core_mask

```
std::vector<bool> SchedulerRoaming::m_core_mask  [private]
```

Definition at line 14 of file scheduler_roaming.h.

Referenced by SchedulerRoaming(), and threadSetInitialAffinity().

The documentation for this class was generated from the following files:

- common/scheduler/ **scheduler_roaming.h**
- common/scheduler/ **scheduler_roaming.cc**

6.311 SchedulerSequential Class Reference

`#include <scheduler_sequential.h>`

Inheritance diagram for SchedulerSequential:

Public Member Functions

- **SchedulerSequential** (**ThreadManager** *thread_manager)
- virtual void **threadSetInitialAffinity** (**thread_id_t** thread_id)
- virtual void **threadStart** (**thread_id_t** thread_id, **SubsecondTime** time)
- virtual void **threadExit** (**thread_id_t** thread_id, **SubsecondTime** time)

Private Member Functions

- void **print_message** (**thread_id_t** thread_id, const char *message)
- void **String2IntVector** (std::vector< String > from, std::vector< int > to)
- void **__sim_end** (**SubsecondTime** time)
- void **results_on_screen** ()
- void **results_on_file** ()

Static Private Member Functions

- static **SInt64** **hook_sim_end** (**UInt64** ptr, **UInt64** time)

Private Attributes

- std::vector< std::priority_queue< **thread_id_t**, std::vector< **thread_id_t** >, std::greater< **thread_id_t** > > > **core_waiting_threads**
- **thread_id_t** **last_thread**
- int **current_pinball_set**
- std::vector< int > **seqs**
- std::vector< int > **next_thread_to_execute**
- bool **verbose**
- unsigned int **total_pinballs**
- unsigned int **cores_working**
- String **outfile**
- **ThreadStatsManager::ThreadStatType** **l1d_load_miss_stat**
- **ThreadStatsManager::ThreadStatType** **l1d_store_miss_stat**
- **ThreadStatsManager::ThreadStatType** **l1i_load_miss_stat**
- **ThreadStatsManager::ThreadStatType** **l1i_store_miss_stat**
- **ThreadStatsManager::ThreadStatType** **l2_load_miss_stat**
- **ThreadStatsManager::ThreadStatType** **l2_store_miss_stat**
- **ThreadStatsManager::ThreadStatType** **l3_load_miss_stat**
- **ThreadStatsManager::ThreadStatType** **l3_store_miss_stat**
- std::ostringstream **aux_mm**
- const **SubsecondTime** **period**

Additional Inherited Members

6.311.1 Detailed Description

Definition at line 11 of file scheduler_sequential.h.

6.311.2 Constructor & Destructor Documentation

6.311.2.1 SchedulerSequential()

```
SchedulerSequential::SchedulerSequential (
            ThreadManager * thread_manager )
```

Definition at line 46 of file scheduler_sequential.cc.

References core_waiting_threads, current_pinball_set, HookType::HOOK_SIM_END, hook_sim_end(), ThreadStatsManager::INVALID, l1d_load_miss_stat, l1d_store_miss_stat, l1i_load_miss_stat, l2_load_miss_stat, l2_store_miss_stat, l3_load_miss_stat, l3_store_miss_stat, LOG_ASSERT_ERROR, next_thread_to_execute, HooksManager::ORDER_ACTION, outfile, ThreadStatNamedStat::registerStat(), seqs, total_pinballs, and verbose.

6.311.3 Member Function Documentation

6.311.3.1 __sim_end()

```
void SchedulerSequential::__sim_end (
            SubsecondTime time )  [private]
```

Definition at line 169 of file scheduler_sequential.cc.

References outfile, results_on_file(), and results_on_screen().

6.311.3.2 hook_sim_end()

```
static  SInt64 SchedulerSequential::hook_sim_end (
            UInt64 ptr,
            UInt64 time )  [inline], [static], [private]
```

Definition at line 59 of file scheduler_sequential.h.

Referenced by SchedulerSequential().

6.311.3.3 print_message()

```
void SchedulerSequential::print_message (
             thread_id_t thread_id,
            const char * message )  [private]
```

Definition at line 231 of file scheduler_sequential.cc.

References verbose.

Referenced by threadExit(), and threadStart().

6.311.3.4 results_on_file()

```
void SchedulerSequential::results_on_file ( )  [private]
```

Definition at line 207 of file scheduler_sequential.cc.

References SubsecondTime::divideRounded(), ThreadStatsManager::ELAPSED_NONIDLE_TIME, SubsecondTime::getNS(), ThreadStatsManager::getThreadStatistic(), ThreadStatsManager::INSTRUCTIONS, l1d_load_miss_stat, l1d_store_miss_stat, l2_load_miss_stat, l2_store_miss_stat, l3_load_miss_stat, l3_store_miss_stat, outfile, period, SubsecondTime::setInternalDataForced(), and total_pinballs.

Referenced by __sim_end().

6.311.3.5 results_on_screen()

```
void SchedulerSequential::results_on_screen ( )  [private]
```

Definition at line 175 of file scheduler_sequential.cc.

References SubsecondTime::divideRounded(), ThreadStatsManager::ELAPSED_NONIDLE_TIME, SubsecondTime::getNS(), ThreadStatsManager::getThreadStatistic(), ThreadStatsManager::INSTRUCTIONS, l1d_load_miss_stat, l1d_store_miss_stat, l1i_load_miss_stat, l2_load_miss_stat, l2_store_miss_stat, l3_load_miss_stat, l3_store_miss_stat, SchedulerPinnedBase::m_thread_info, period, SubsecondTime::setInternalDataForced(), and total_pinballs.

Referenced by __sim_end().

6.311.3.6 String2IntVector()

```
void SchedulerSequential::String2IntVector (
            std::vector< String > from,
            std::vector< int > to )  [private]
```

6.311.3.7 threadExit()

```
void SchedulerSequential::threadExit (
              thread_id_t thread_id,
              SubsecondTime time )  [virtual]
```

Reimplemented from **SchedulerPinnedBase** (p. 1142).

Definition at line 126 of file scheduler_sequential.cc.

References core_waiting_threads, SchedulerPinnedBase::m_thread_info, SchedulerDynamic::m_threads_↩ runnable, next_thread_to_execute, print_message(), and SchedulerPinnedBase::reschedule().

6.311.3.8 threadSetInitialAffinity()

```
void SchedulerSequential::threadSetInitialAffinity (
              thread_id_t thread_id )  [virtual]
```

Implements **SchedulerPinnedBase** (p. 1143).

Definition at line 158 of file scheduler_sequential.cc.

References current_pinball_set, SchedulerPinnedBase::m_thread_info, and seqs.

6.311.3.9 threadStart()

```
void SchedulerSequential::threadStart (
              thread_id_t thread_id,
              SubsecondTime time )  [virtual]
```

Reimplemented from **SchedulerPinnedBase** (p. 1144).

Definition at line 99 of file scheduler_sequential.cc.

References aux_mm, core_waiting_threads, SchedulerPinnedBase::m_core_thread_running, SchedulerPinned↩ Base::m_thread_info, SchedulerDynamic::m_threads_runnable, next_thread_to_execute, print_message(), and total_pinballs.

6.311.4 Member Data Documentation

6.311.4.1 aux_mm

```
std::ostringstream SchedulerSequential::aux_mm  [private]
```

Definition at line 47 of file scheduler_sequential.h.

Referenced by threadStart().

6.311.4.2 core_waiting_threads

```
std::vector<std::priority_queue< thread_id_t, std::vector< thread_id_t>, std::greater< thread_id_t> > > SchedulerSequential::core_waiting_threads [private]
```

Definition at line 28 of file scheduler_sequential.h.

Referenced by SchedulerSequential(), threadExit(), and threadStart().

6.311.4.3 cores_working

```
unsigned int SchedulerSequential::cores_working [private]
```

Definition at line 35 of file scheduler_sequential.h.

6.311.4.4 current_pinball_set

```
int SchedulerSequential::current_pinball_set [private]
```

Definition at line 30 of file scheduler_sequential.h.

Referenced by SchedulerSequential(), and threadSetInitialAffinity().

6.311.4.5 l1d_load_miss_stat

```
ThreadStatsManager::ThreadStatType SchedulerSequential::l1d_load_miss_stat [private]
```

Definition at line 38 of file scheduler_sequential.h.

Referenced by results_on_file(), results_on_screen(), and SchedulerSequential().

6.311.4.6 l1d_store_miss_stat

```
ThreadStatsManager::ThreadStatType SchedulerSequential::l1d_store_miss_stat [private]
```

Definition at line 39 of file scheduler_sequential.h.

Referenced by results_on_file(), results_on_screen(), and SchedulerSequential().

6.311.4.7 l1i_load_miss_stat

```
ThreadStatsManager::ThreadStatType SchedulerSequential::l1i_load_miss_stat  [private]
```

Definition at line 40 of file scheduler_sequential.h.

Referenced by results_on_screen(), and SchedulerSequential().

6.311.4.8 l1i_store_miss_stat

```
ThreadStatsManager::ThreadStatType SchedulerSequential::l1i_store_miss_stat  [private]
```

Definition at line 41 of file scheduler_sequential.h.

6.311.4.9 l2_load_miss_stat

```
ThreadStatsManager::ThreadStatType SchedulerSequential::l2_load_miss_stat  [private]
```

Definition at line 42 of file scheduler_sequential.h.

Referenced by results_on_file(), results_on_screen(), and SchedulerSequential().

6.311.4.10 l2_store_miss_stat

```
ThreadStatsManager::ThreadStatType SchedulerSequential::l2_store_miss_stat  [private]
```

Definition at line 43 of file scheduler_sequential.h.

Referenced by results_on_file(), results_on_screen(), and SchedulerSequential().

6.311.4.11 l3_load_miss_stat

```
ThreadStatsManager::ThreadStatType SchedulerSequential::l3_load_miss_stat  [private]
```

Definition at line 44 of file scheduler_sequential.h.

Referenced by results_on_file(), results_on_screen(), and SchedulerSequential().

6.311.4.12 l3_store_miss_stat

ThreadStatsManager::ThreadStatType SchedulerSequential::l3_store_miss_stat [private]

Definition at line 45 of file scheduler_sequential.h.

Referenced by results_on_file(), results_on_screen(), and SchedulerSequential().

6.311.4.13 last_thread

thread_id_t SchedulerSequential::last_thread [private]

Definition at line 29 of file scheduler_sequential.h.

6.311.4.14 next_thread_to_execute

std::vector<int> SchedulerSequential::next_thread_to_execute [private]

Definition at line 32 of file scheduler_sequential.h.

Referenced by SchedulerSequential(), threadExit(), and threadStart().

6.311.4.15 outfile

String SchedulerSequential::outfile [private]

Definition at line 36 of file scheduler_sequential.h.

Referenced by __sim_end(), results_on_file(), and SchedulerSequential().

6.311.4.16 period

const **SubsecondTime** SchedulerSequential::period [private]

Definition at line 49 of file scheduler_sequential.h.

Referenced by results_on_file(), and results_on_screen().

6.311.4.17 seqs

```
std::vector<int> SchedulerSequential::seqs  [private]
```

Definition at line 31 of file scheduler_sequential.h.

Referenced by SchedulerSequential(), and threadSetInitialAffinity().

6.311.4.18 total_pinballs

```
unsigned int SchedulerSequential::total_pinballs  [private]
```

Definition at line 34 of file scheduler_sequential.h.

Referenced by results_on_file(), results_on_screen(), SchedulerSequential(), and threadStart().

6.311.4.19 verbose

```
bool SchedulerSequential::verbose  [private]
```

Definition at line 33 of file scheduler_sequential.h.

Referenced by print_message(), and SchedulerSequential().

The documentation for this class was generated from the following files:

- common/scheduler/ **scheduler_sequential.h**
- common/scheduler/ **scheduler_sequential.cc**

6.312 SchedulerStatic Class Reference

```
#include <scheduler_static.h>
```

Inheritance diagram for SchedulerStatic:

Public Member Functions

- **SchedulerStatic** (**ThreadManager** ∗thread_manager)
- **core_id_t threadCreate** (**thread_id_t**)

Private Member Functions

- **core_id_t findFirstFreeMaskedCore** ()

Private Attributes

- std::vector< bool > **m_core_mask**

Additional Inherited Members

6.312.1 Detailed Description

Definition at line 8 of file scheduler_static.h.

6.312.2 Constructor & Destructor Documentation

6.312.2.1 SchedulerStatic()

```
SchedulerStatic::SchedulerStatic (
            ThreadManager * thread_manager )
```

Definition at line 8 of file scheduler_static.cc.

References m_core_mask.

6.312.3 Member Function Documentation

6.312.3.1 findFirstFreeMaskedCore()

```
core_id_t SchedulerStatic::findFirstFreeMaskedCore ( )  [private]
```

Definition at line 19 of file scheduler_static.cc.

References Core::IDLE, INVALID_CORE_ID, and m_core_mask.

Referenced by threadCreate().

6.312.3.2 threadCreate()

```
core_id_t SchedulerStatic::threadCreate (
            thread_id_t thread_id )  [virtual]
```

Implements **Scheduler** (p. 1123).

Definition at line 31 of file scheduler_static.cc.

References findFirstFreeMaskedCore(), INVALID_CORE_ID, LOG_ASSERT_ERROR, and LOG_PRINT.

6.312.4 Member Data Documentation

6.312.4.1 m_core_mask

```
std::vector<bool> SchedulerStatic::m_core_mask  [private]
```

Definition at line 16 of file scheduler_static.h.

Referenced by findFirstFreeMaskedCore(), and SchedulerStatic().

The documentation for this class was generated from the following files:

- common/scheduler/ **scheduler_static.h**
- common/scheduler/ **scheduler_static.cc**

6.313 ScopedFxsave Class Reference

```
#include <fxsupport.h>
```

Public Member Functions

- **ScopedFxsave** ()
- ∼**ScopedFxsave** ()

6.313.1 Detailed Description

Definition at line 29 of file fxsupport.h.

6.313.2 Constructor & Destructor Documentation

6.313.2.1 ScopedFxsave()

```
ScopedFxsave::ScopedFxsave ( )  [inline]
```

Definition at line 32 of file fxsupport.h.

References Fxsupport::fxsave(), and Fxsupport::getSingleton().

6.313.2.2 ~ScopedFxsave()

```
ScopedFxsave::~ScopedFxsave ( )  [inline]
```

Definition at line 33 of file fxsupport.h.

References Fxsupport::fxrstor(), and Fxsupport::getSingleton().

The documentation for this class was generated from the following file:

- common/misc/ **fxsupport.h**

6.314 ScopedLock Class Reference

```
#include <lock.h>
```

Public Member Functions

- **ScopedLock** (**BaseLock** &lock)
- ~**ScopedLock** ()

Private Attributes

- **BaseLock** & **_lock**

6.314.1 Detailed Description

Definition at line 127 of file lock.h.

6.314.2 Constructor & Destructor Documentation

6.314.2.1 ScopedLock()

```
ScopedLock::ScopedLock (
              BaseLock & lock )  [inline]
```

Definition at line 132 of file lock.h.

References _lock, and BaseLock::acquire().

6.314.2.2 ~ScopedLock()

```
ScopedLock::~ScopedLock ( )  [inline]
```

Definition at line 138 of file lock.h.

References _lock, and BaseLock::release().

6.314.3 Member Data Documentation

6.314.3.1 _lock

```
BaseLock& ScopedLock::_lock  [private]
```

Definition at line 129 of file lock.h.

Referenced by ScopedLock(), and ~ScopedLock().

The documentation for this class was generated from the following file:

- common/misc/ **lock.h**

6.315 ScopedReadLock Class Reference

`#include <lock.h>`

Public Member Functions

- **ScopedReadLock** (**BaseLock** &lock)
- ~**ScopedReadLock** ()

Private Attributes

- **BaseLock** & **_lock**

6.315.1 Detailed Description

Definition at line 145 of file lock.h.

6.315.2 Constructor & Destructor Documentation

6.315.2.1 ScopedReadLock()

```
ScopedReadLock::ScopedReadLock (
              BaseLock & lock )  [inline]
```

Definition at line 150 of file lock.h.

References _lock, and BaseLock::acquire_read().

6.315.2.2 ~ScopedReadLock()

```
ScopedReadLock::~ScopedReadLock ( )  [inline]
```

Definition at line 156 of file lock.h.

References _lock, and BaseLock::release_read().

6.315.3 Member Data Documentation

6.315.3.1 _lock

```
BaseLock& ScopedReadLock::_lock  [private]
```

Definition at line 147 of file lock.h.

Referenced by ScopedReadLock(), and ~ScopedReadLock().

The documentation for this class was generated from the following file:

- common/misc/ **lock.h**

6.316 ScopedTimer Class Reference

```
#include <timer.h>
```

Public Member Functions

- **ScopedTimer** (**TotalTimer** &_total)
- ∼**ScopedTimer** ()

Private Attributes

- **TotalTimer** & **total**
- **Timer** **timer**

6.316.1 Detailed Description

Definition at line 66 of file timer.h.

6.316.2 Constructor & Destructor Documentation

6.316.2.1 ScopedTimer()

```
ScopedTimer::ScopedTimer (
              TotalTimer & _total )  [inline]
```

Definition at line 73 of file timer.h.

6.316.2.2 ∼ScopedTimer()

```
ScopedTimer::∼ScopedTimer ( )  [inline]
```

Definition at line 78 of file timer.h.

References TotalTimer::add(), Timer::getTime(), Timer::switched, timer, and total.

6.316.3 Member Data Documentation

6.316.3.1 timer

```
Timer ScopedTimer::timer  [private]
```

Definition at line 70 of file timer.h.

Referenced by ∼ScopedTimer().

6.316.3.2 total

```
TotalTimer& ScopedTimer::total  [private]
```

Definition at line 69 of file timer.h.

Referenced by ∼ScopedTimer().

The documentation for this class was generated from the following file:

- common/misc/ **timer.h**

6.317 SpinLoopDetector::SdtEntry Struct Reference

Public Member Functions

- **SdtEntry** (uint64_t _eip, uint8_t _id, uint64_t _icount, **SubsecondTime** _tcount)

Public Attributes

- uint64_t **eip**
- uint8_t **id**
- uint64_t **icount**
- **SubsecondTime** **tcount**

6.317.1 Detailed Description

Definition at line 18 of file spin_loop_detector.h.

6.317.2 Constructor & Destructor Documentation

6.317.2.1 SdtEntry()

```
SpinLoopDetector::SdtEntry::SdtEntry (
            uint64_t _eip,
            uint8_t _id,
            uint64_t _icount,
             SubsecondTime _tcount )  [inline]
```

Definition at line 20 of file spin_loop_detector.h.

6.317.3 Member Data Documentation

6.317.3.1 eip

```
uint64_t SpinLoopDetector::SdtEntry::eip
```

Definition at line 21 of file spin_loop_detector.h.

6.317.3.2 icount

```
uint64_t SpinLoopDetector::SdtEntry::icount
```

Definition at line 23 of file spin_loop_detector.h.

6.317.3.3 id

```
uint8_t SpinLoopDetector::SdtEntry::id
```

Definition at line 22 of file spin_loop_detector.h.

6.317.3.4 tcount

```
SubsecondTime SpinLoopDetector::SdtEntry::tcount
```

Definition at line 24 of file spin_loop_detector.h.

The documentation for this struct was generated from the following file:

- common/misc/ **spin_loop_detector.h**

6.318 config::Section Class Reference

Section (p. 1163): A configuration section entry This class is used to hold a given section from a configuration. It contains both a list of subsections as well as a list of keys. The root of a configuration tree is of this class type.

```
#include <section.hpp>
```

Public Member Functions

- **Section** (const String &name, bool case_sensitive)

  Section() *(p. 1165) Constructor for a root section.*
- **Section** (**Section** const &parent, const String &name, bool case_sensitive)

  Section() *(p. 1165) Constructor for a subsection.*
- ∼**Section** ()

  ∼***Section()*** *(p. 1165) Destructor*
- bool **isRoot** () const

  Determine if this section is the root of the given configuration tree.
- bool **hasKey** (const String &name, **UInt64** index) const

  returns true if the given name is a key within this section
- bool **hasSection** (const String &name) const

  returns true if the given name is a subsection within this section
- const **SectionList** & **getSubsections** () const

  SectionList() *(p. 36) returns the list of subsections.*
- const **KeyList** & **getKeys** () const

  Returns the list of keys.
- const **KeyArrayList** & **getArrayKeys** () const

  Returns the list of keys.
- String **getName** () const

  Returns the name of this section.
- const **Key** & **getKey** (const String &name, uint64_t index=UINT64_MAX)

  getKey() *(p. 1168) returns a key with the given name*
- **Section** & **addSubsection** (const String &name)
- const **Key** & **addKey** (const String &name, const String &value, **UInt64** index=UINT64_MAX)
- const **Key** & **addKey** (const String &name, const **SInt64** &value, **UInt64** index=UINT64_MAX)
- const **Key** & **addKey** (const String &name, const double &value, **UInt64** index=UINT64_MAX)
- const **Section** & **getSection** (const String &name)
- const String **getFullPath** () const

  getFullPath() *(p. 1168) Returns the path from the root to this section*
- const **Section** & **getParent** () const

Private Member Functions

- template<class V >
  const **Key** & **addKeyInternal** (const String &name, const V &value, **UInt64** index=UINT64_MAX)
- void **clear** ()
- **Section** & **getSection_unsafe** (const String &name)

Private Attributes

- String **m_name**
- **SectionList m_subSections**

  A list of bl::config::Section subsections.
- **KeyList m_keys**

  A list of bl::config::Key keys in this current section.
- **KeyArrayList m_array_keys**
- const **Section** & **m_parent**

  A bl::config::Section parent.
- bool **m_isroot**

  Whether or not this section is the root section of the config.
- bool **m_case_sensitive**

  Whether or not key lookups are case-sensitive.

Friends

- class **Config**

6.318.1 Detailed Description

Section (p. 1163): A configuration section entry This class is used to hold a given section from a configuration. It contains both a list of subsections as well as a list of keys. The root of a configuration tree is of this class type.

Definition at line 29 of file section.hpp.

6.318.2 Constructor & Destructor Documentation

6.318.2.1 Section() [1/2]

```
config::Section::Section (
            const String & name,
            bool case_sensitive )
```

Section() (p. 1165) Constructor for a root section.

Definition at line 14 of file section.cpp.

Referenced by addSubsection().

6.318.2.2 Section() [2/2]

```
config::Section::Section (
             Section const & parent,
            const String & name,
            bool case_sensitive )
```

Section() (p. 1165) Constructor for a subsection.

Definition at line 26 of file section.cpp.

6.318.2.3 ~Section()

```
config::Section::~Section ( )
```

~**Section()** (p. 1165) Destructor

Definition at line 38 of file section.cpp.

References m_array_keys, m_keys, and m_subSections.

6.318.3 Member Function Documentation

6.318.3.1 addKey() [1/3]

```
const  Key& config::Section::addKey (
             const String & name,
             const double & value,
              UInt64 index = UINT64_MAX )  [inline]
```

Definition at line 83 of file section.hpp.

References addKeyInternal().

6.318.3.2 addKey() [2/3]

```
const  Key& config::Section::addKey (
             const String & name,
             const  SInt64 & value,
              UInt64 index = UINT64_MAX )  [inline]
```

Definition at line 82 of file section.hpp.

References addKeyInternal().

6.318.3.3 addKey() [3/3]

```
const  Key& config::Section::addKey (
             const String & name,
             const String & value,
              UInt64 index = UINT64_MAX )  [inline]
```

addKey() (p. 1166) Add a key with the given name

Parameters

name	The name of the new key
value	The value of the new key (as a string)

Returns

A reference to the newly created key

Definition at line 81 of file section.hpp.

References addKeyInternal().

Referenced by config::Config::addKeyInternal(), and config::ConfigFile::evalTree().

6.318.3.4 addKeyInternal()

```
template<class V >
const template  Key & config::Section::addKeyInternal (
            const String & name,
            const V & value,
             UInt64 index = UINT64_MAX )  [private]
```

Definition at line 57 of file section.cpp.

References getFullPath(), m_array_keys, m_case_sensitive, and m_keys.

Referenced by addKey().

6.318.3.5 addSubsection()

```
 Section & config::Section::addSubsection (
            const String & name )
```

addSubSection() Add a subsection with the given name

Parameters

name	The name of the new subsection

Returns

A reference to the newly created subsection

Definition at line 46 of file section.cpp.

References m_case_sensitive, m_subSections, and Section().

Referenced by config::Config::addSection(), and config::Config::getSection_unsafe().

6.318.3.6 clear()

```
void config::Section::clear ( )  [inline], [private]
```

clear() (p. 1167) Clears out any sub-sections, used during a (re)load

Definition at line 104 of file section.hpp.

References m_subSections.

Referenced by config::Config::clear().

6.318.3.7 getArrayKeys()

```
const  KeyArrayList& config::Section::getArrayKeys ( ) const  [inline]
```

Returns the list of keys.

Definition at line 62 of file section.hpp.

References m_array_keys.

Referenced by config::ConfigFile::SaveTreeAs().

6.318.3.8 getFullPath()

```
const String config::Section::getFullPath ( ) const
```

getFullPath() (p. 1168) Returns the path from the root to this section

Definition at line 188 of file section.cpp.

References getName(), getParent(), and isRoot().

Referenced by addKeyInternal(), and config::ConfigFile::SaveTreeAs().

6.318.3.9 getKey()

```
const  Key & config::Section::getKey (
             const String & name,
             uint64_t index = UINT64_MAX )
```

getKey() (p. 1168) returns a key with the given name

Definition at line 111 of file section.cpp.

References m_array_keys, m_case_sensitive, and m_keys.

Referenced by config::Config::getKey().

6.318.3.10 getKeys()

```
const  KeyList& config::Section::getKeys ( ) const  [inline]
```

Returns the list of keys.

Definition at line 59 of file section.hpp.

References m_keys.

Referenced by config::ConfigFile::SaveTreeAs(), and config::Config::showTree().

6.318.3.11 getName()

```
String config::Section::getName ( ) const  [inline]
```

Returns the name of this section.

Definition at line 65 of file section.hpp.

References m_name.

Referenced by getFullPath(), and config::Config::showTree().

6.318.3.12 getParent()

```
const  Section& config::Section::getParent ( ) const  [inline]
```

getParent() (p. 1169) Returns a reference to the parent section. Note: The **isRoot()** (p. 1171) method can be used to determine if a section is the root of the tree

Definition at line 97 of file section.hpp.

References m_parent.

Referenced by getFullPath().

6.318.3.13 getSection()

```
const  Section & config::Section::getSection (
             const String & name )
```

getSection() (p. 1169) Returns a reference to the section with the given name

Parameters

name	The name of the section we are trying to obtain

Returns

A reference to the named section, creating it if it doesn't exist

Definition at line 140 of file section.cpp.

References getSection_unsafe().

6.318.3.14 getSection_unsafe()

```
Section & config::Section::getSection_unsafe (
            const String & name )  [private]
```

Definition at line 146 of file section.cpp.

References m_case_sensitive, and m_subSections.

Referenced by getSection(), and config::Config::getSection_unsafe().

6.318.3.15 getSubsections()

```
const  SectionList& config::Section::getSubsections ( ) const  [inline]
```

SectionList() (p. 36) returns the list of subsections.

Definition at line 56 of file section.hpp.

References m_subSections.

Referenced by config::ConfigFile::SaveTreeAs(), config::Config::showTree(), and TagsManager::TagsManager().

6.318.3.16 hasKey()

```
bool config::Section::hasKey (
            const String & name,
             UInt64 index ) const
```

returns true if the given name is a key within this section

Parameters

name	The name of the key
case_sensitive	Whether or not the lookup should care about case

Returns

True if name is a key within this section

Definition at line 163 of file section.cpp.

References m_array_keys, m_case_sensitive, and m_keys.

Referenced by config::Config::getKey(), and config::Config::hasKey().

6.318.3.17 hasSection()

```
bool config::Section::hasSection (
            const String & name ) const
```

returns true if the given name is a subsection within this section

Parameters

name	The name of the subsection
case_sensitive	Whether or not the lookup should care about case

Returns

True if name is a subsection within this section

Definition at line 154 of file section.cpp.

References m_case_sensitive, and m_subSections.

Referenced by config::Config::getSection_unsafe().

6.318.3.18 isRoot()

```
bool config::Section::isRoot ( ) const  [inline]
```

Determine if this section is the root of the given configuration tree.

Definition at line 39 of file section.hpp.

References m_isroot.

Referenced by config::ConfigFile::evalTree(), getFullPath(), and config::ConfigFile::SaveTreeAs().

6.318.4 Friends And Related Function Documentation

6.318.4.1 Config

```
friend class Config [friend]
```

Definition at line 32 of file section.hpp.

6.318.5 Member Data Documentation

6.318.5.1 m_array_keys

KeyArrayList config::Section::m_array_keys [private]

Definition at line 115 of file section.hpp.

Referenced by addKeyInternal(), getArrayKeys(), getKey(), hasKey(), and ~Section().

6.318.5.2 m_case_sensitive

bool config::Section::m_case_sensitive [private]

Whether or not key lookups are case-sensitive.

Definition at line 124 of file section.hpp.

Referenced by addKeyInternal(), addSubsection(), getKey(), getSection_unsafe(), hasKey(), and hasSection().

6.318.5.3 m_isroot

bool config::Section::m_isroot [private]

Whether or not this section is the root section of the config.

Definition at line 121 of file section.hpp.

Referenced by isRoot().

6.318.5.4 m_keys

KeyList config::Section::m_keys [private]

A list of bl::config::Key keys in this current section.

Definition at line 113 of file section.hpp.

Referenced by addKeyInternal(), getKey(), getKeys(), hasKey(), and ~Section().

6.318.5.5 m_name

String config::Section::m_name [private]

Definition at line 107 of file section.hpp.

Referenced by getName().

6.318.5.6 m_parent

```
const Section& config::Section::m_parent [private]
```

A bl::config::Section parent.

Definition at line 118 of file section.hpp.

Referenced by getParent().

6.318.5.7 m_subSections

```
SectionList config::Section::m_subSections [private]
```

A list of bl::config::Section subsections.

Definition at line 110 of file section.hpp.

Referenced by addSubsection(), clear(), getSection_unsafe(), getSubsections(), hasSection(), and ~Section().

The documentation for this class was generated from the following files:

- common/config/ **section.hpp**
- common/config/ **section.cpp**

6.319 SELock Class Reference

```
#include <selock.h>
```

Inheritance diagram for SELock:

Public Member Functions

- **SELock** (void)
- void **acquire_exclusive** (void)
- void **release_exclusive** (void)
- void **acquire_shared** (void)
- void **release_shared** (void)
- void **upgrade** (void)
- void **downgrade** (void)

Private Attributes

- **Lock m_lock**
- bool **m_write**
- **UInt64 m_writers**
- **UInt64 m_readers**

6.319.1 Detailed Description

Definition at line 8 of file selock.h.

6.319.2 Constructor & Destructor Documentation

6.319.2.1 SELock()

```
SELock::SELock (
            void  )
```

Definition at line 4 of file selock.cc.

References TotalTimer::getTimerByStacktrace(), and itostr().

6.319.3 Member Function Documentation

6.319.3.1 acquire_exclusive()

```
void SELock::acquire_exclusive (
            void  )
```

Definition at line 24 of file selock.cc.

References m_readers, m_write, m_writers, and WAIT_WHILE.

6.319.3.2 acquire_shared()

```
void SELock::acquire_shared (
            void  )
```

Definition at line 62 of file selock.cc.

References m_readers, m_write, and WAIT_WHILE.

Referenced by _SELock::acquire_shared().

6.319.3.3 downgrade()

```
void SELock::downgrade (
            void  )
```

Definition at line 146 of file selock.cc.

References m_readers, and m_write.

Referenced by _SELock::downgrade().

6.319.3.4 release_exclusive()

```
void SELock::release_exclusive (
            void  )
```

Definition at line 53 of file selock.cc.

References m_write.

6.319.3.5 release_shared()

```
void SELock::release_shared (
            void  )
```

Definition at line 91 of file selock.cc.

References m_readers.

Referenced by _SELock::release_shared().

6.319.3.6 upgrade()

```
void SELock::upgrade (
            void  )
```

Definition at line 99 of file selock.cc.

References m_readers, m_write, m_writers, and WAIT_WHILE.

Referenced by _SELock::upgrade().

6.319.4 Member Data Documentation

6.319.4.1 m_lock

Lock SELock::m_lock [private]

Definition at line 20 of file selock.h.

6.319.4.2 m_readers

UInt64 SELock::m_readers [private]

Definition at line 23 of file selock.h.

Referenced by acquire_exclusive(), acquire_shared(), downgrade(), release_shared(), and upgrade().

6.319.4.3 m_write

bool SELock::m_write [private]

Definition at line 21 of file selock.h.

Referenced by acquire_exclusive(), acquire_shared(), downgrade(), release_exclusive(), and upgrade().

6.319.4.4 m_writers

UInt64 SELock::m_writers [private]

Definition at line 22 of file selock.h.

Referenced by acquire_exclusive(), and upgrade().

The documentation for this class was generated from the following files:

- common/misc/ **selock.h**
- common/misc/ **selock.cc**

6.320 Semaphore Class Reference

#include <semaphore.h>

Public Member Functions

- **Semaphore** (int count)
- **Semaphore** ()
- ∼**Semaphore** ()
- void **wait** ()
- void **signal** ()
- void **broadcast** ()

Private Attributes

- int **_count**
- int **_numWaiting**
- int **_futx**
- **Lock _lock**

6.320.1 Detailed Description

Definition at line 6 of file semaphore.h.

6.320.2 Constructor & Destructor Documentation

6.320.2.1 Semaphore() [1/2]

```
Semaphore::Semaphore (
            int count )
```

Definition at line 9 of file semaphore.cc.

6.320.2.2 Semaphore() [2/2]

```
Semaphore::Semaphore ( )
```

Definition at line 16 of file semaphore.cc.

6.320.2.3 ∼Semaphore()

```
Semaphore::∼Semaphore ( )
```

Definition at line 23 of file semaphore.cc.

6.320.3 Member Function Documentation

6.320.3.1 broadcast()

```
void Semaphore::broadcast ( )
```

Definition at line 63 of file semaphore.cc.

References _count, _futx, _lock, _numWaiting, TLock< T_LockCreator >::acquire(), FUTEX_PRIVATE_FLAG, and TLock< T_LockCreator >::release().

6.320.3.2 signal()

```
void Semaphore::signal ( )
```

Definition at line 49 of file semaphore.cc.

References _count, _futx, _lock, _numWaiting, TLock< T_LockCreator >::acquire(), FUTEX_PRIVATE_FLAG, and TLock< T_LockCreator >::release().

Referenced by TraceManager::stop(), and ParametricDramDirectoryMSI::CacheCntlr::wakeUpNetworkThread().

6.320.3.3 wait()

```
void Semaphore::wait ( )
```

Definition at line 27 of file semaphore.cc.

References _count, _futx, _lock, _numWaiting, TLock< T_LockCreator >::acquire(), FUTEX_PRIVATE_FLAG, and TLock< T_LockCreator >::release().

Referenced by TraceManager::wait(), and ParametricDramDirectoryMSI::CacheCntlr::waitForNetworkThread().

6.320.4 Member Data Documentation

6.320.4.1 _count

```
int Semaphore::_count  [private]
```

Definition at line 9 of file semaphore.h.

Referenced by broadcast(), signal(), and wait().

6.320.4.2 _futx

```
int Semaphore::_futx  [private]
```

Definition at line 11 of file semaphore.h.

Referenced by broadcast(), signal(), and wait().

6.320.4.3 _lock

```
Lock Semaphore::_lock  [private]
```

Definition at line 12 of file semaphore.h.

Referenced by broadcast(), signal(), and wait().

6.320.4.4 _numWaiting

```
int Semaphore::_numWaiting  [private]
```

Definition at line 10 of file semaphore.h.

Referenced by broadcast(), signal(), and wait().

The documentation for this class was generated from the following files:

- common/misc/ **semaphore.h**
- common/misc/ **semaphore.cc**

6.321 SharedCacheBlockInfo Class Reference

```
#include <shared_cache_block_info.h>
```

Inheritance diagram for SharedCacheBlockInfo:

Public Member Functions

- **SharedCacheBlockInfo** (**IntPtr** tag=∼0, **CacheState::cstate_t** cstate= **CacheState::INVALID**)
- ∼**SharedCacheBlockInfo** ()
- void **invalidate** ()
- void **clone** (**CacheBlockInfo** ∗cache_block_info)

Additional Inherited Members

6.321.1 Detailed Description

Definition at line 20 of file shared_cache_block_info.h.

6.321.2 Constructor & Destructor Documentation

6.321.2.1 SharedCacheBlockInfo()

```
SharedCacheBlockInfo::SharedCacheBlockInfo (
            IntPtr tag = ~0,
            CacheState::cstate_t cstate =  CacheState::INVALID )  [inline]
```

Definition at line 28 of file shared_cache_block_info.h.

6.321.2.2 ~SharedCacheBlockInfo()

```
SharedCacheBlockInfo::~SharedCacheBlockInfo ( )  [inline]
```

Definition at line 36 of file shared_cache_block_info.h.

6.321.3 Member Function Documentation

6.321.3.1 clone()

```
void SharedCacheBlockInfo::clone (
            CacheBlockInfo * cache_block_info )  [virtual]
```

Reimplemented from **CacheBlockInfo** (p. 152).

Definition at line 46 of file shared_cache_block_info.cc.

References CacheBlockInfo::clone().

6.321.3.2 invalidate()

```
void SharedCacheBlockInfo::invalidate ( )  [virtual]
```

Reimplemented from **CacheBlockInfo** (p. 154).

Definition at line 36 of file shared_cache_block_info.cc.

References CacheBlockInfo::invalidate().

Referenced by ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/cache/ **shared_cache_block_info.h**
- common/core/memory_subsystem/cache/ **shared_cache_block_info.cc**

6.322 PrL1PrL2DramDirectoryMSI::ShmemMsg Class Reference

```
#include <shmem_msg.h>
```

Public Types

- enum **msg_t** {
  INVALID_MSG_TYPE = 0, **MIN_MSG_TYPE**, **EX_REQ** = MIN_MSG_TYPE, **SH_REQ**,
  UPGRADE_REQ, **INV_REQ**, **FLUSH_REQ**, **WB_REQ**,
  EX_REP, **SH_REP**, **UPGRADE_REP**, **INV_REP**,
  FLUSH_REP, **WB_REP**, **NULLIFY_REQ**, **DRAM_READ_REQ**,
  DRAM_WRITE_REQ, **DRAM_READ_REP**, **MAX_MSG_TYPE** = NULLIFY_REQ, **NUM_MSG_TYPES** = MAX_MSG_TYPE - MIN_MSG_TYPE + 1 }

Public Member Functions

- **ShmemMsg** ()=delete
- **ShmemMsg** (**ShmemPerf** *perf)
- **ShmemMsg** (**msg_t** msg_type, **MemComponent::component_t** sender_mem_component, **MemComponent::component_t** receiver_mem_component, **core_id_t** requester, **IntPtr** address, **Byte** *data_buf, **UInt32** data_length, **ShmemPerf** *perf)
- **ShmemMsg** (**ShmemMsg** *shmem_msg)
- ~**ShmemMsg** ()
- **Byte** * **makeMsgBuf** ()
- **UInt32** **getMsgLen** ()
- **UInt32** **getModeledLength** ()
- **msg_t** **getMsgType** ()
- **MemComponent::component_t** **getSenderMemComponent** ()
- **MemComponent::component_t** **getReceiverMemComponent** ()
- **core_id_t** **getRequester** ()
- **IntPtr** **getAddress** ()
- **Byte** * **getDataBuf** ()
- **UInt32** **getDataLength** ()
- **HitWhere::where_t** **getWhere** ()
- void **setDataBuf** (**Byte** *data_buf)
- void **setWhere** (**HitWhere::where_t** where)
- **ShmemPerf** * **getPerf** ()

Static Public Member Functions

- static **ShmemMsg** * **getShmemMsg** (**Byte** *msg_buf, **ShmemPerf** *perf)

Private Attributes

- **msg_t m_msg_type**
- **MemComponent::component_t m_sender_mem_component**
- **MemComponent::component_t m_receiver_mem_component**
- **core_id_t m_requester**
- **HitWhere::where_t m_where**
- **IntPtr m_address**
- **Byte * m_data_buf**
- **UInt32 m_data_length**
- **ShmemPerf * m_perf**

6.322.1 Detailed Description

Definition at line 11 of file shmem_msg.h.

6.322.2 Member Enumeration Documentation

6.322.2.1 msg_t

enum **PrL1PrL2DramDirectoryMSI::ShmemMsg::msg_t**

Enumerator

INVALID_MSG_TYPE	
MIN_MSG_TYPE	
EX_REQ	
SH_REQ	
UPGRADE_REQ	
INV_REQ	
FLUSH_REQ	
WB_REQ	
EX_REP	
SH_REP	
UPGRADE_REP	
INV_REP	
FLUSH_REP	
WB_REP	
NULLIFY_REQ	
DRAM_READ_REQ	
DRAM_WRITE_REQ	
DRAM_READ_REP	
MAX_MSG_TYPE	
NUM_MSG_TYPES	

Definition at line 14 of file shmem_msg.h.

6.322.3 Constructor & Destructor Documentation

6.322.3.1 ShmemMsg() [1/4]

```
PrL1PrL2DramDirectoryMSI::ShmemMsg::ShmemMsg ( )  [delete]
```

Referenced by getShmemMsg().

6.322.3.2 ShmemMsg() [2/4]

```
PrL1PrL2DramDirectoryMSI::ShmemMsg::ShmemMsg (
              ShmemPerf * perf )
```

Definition at line 8 of file shmem_msg.cc.

6.322.3.3 ShmemMsg() [3/4]

```
PrL1PrL2DramDirectoryMSI::ShmemMsg::ShmemMsg (
              msg_t msg_type,
              MemComponent::component_t sender_mem_component,
              MemComponent::component_t receiver_mem_component,
              core_id_t requester,
              IntPtr address,
              Byte * data_buf,
              UInt32 data_length,
              ShmemPerf * perf )
```

Definition at line 20 of file shmem_msg.cc.

6.322.3.4 ShmemMsg() [4/4]

```
PrL1PrL2DramDirectoryMSI::ShmemMsg::ShmemMsg (
              ShmemMsg * shmem_msg )
```

Definition at line 39 of file shmem_msg.cc.

6.322.3.5 ~ShmemMsg()

```
PrL1PrL2DramDirectoryMSI::ShmemMsg::~ShmemMsg ( )
```

Definition at line 50 of file shmem_msg.cc.

6.322.4 Member Function Documentation

6.322.4.1 getAddress()

```
IntPtr PrL1PrL2DramDirectoryMSI::ShmemMsg::getAddress ( )  [inline]
```

Definition at line 80 of file shmem_msg.h.

References m_address.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), PrL1PrL2Dram↩
DirectoryMSI::DramDirectoryCntlr::handleMsgFromL2Cache(), ParametricDramDirectoryMSI::MemoryManager↩
::handleMsgFromNetwork(), DramCntlrInterface::handleMsgFromTagDirectory(), PrL1PrL2DramDirectoryMSI↩
::DramDirectoryCntlr::processDirectoryEntryAllocationReq(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr↩
::processDRAMReply(), ParametricDramDirectoryMSI::CacheCntlr::processExRepFromDramDirectory(), Pr↩
L1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2DramDirectoryMS↩
I::DramDirectoryCntlr::processFlushRepFromL2Cache(), ParametricDramDirectoryMSI::CacheCntlr::process↩
FlushReqFromDramDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache(),
ParametricDramDirectoryMSI::CacheCntlr::processInvReqFromDramDirectory(), PrL1PrL2DramDirectoryM↩
SI::DramDirectoryCntlr::processNullifyReq(), ParametricDramDirectoryMSI::CacheCntlr::processShRepFrom↩
DramDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processShReqFromL2Cache(), Parametric↩
DramDirectoryMSI::CacheCntlr::processUpgradeRepFromDramDirectory(), PrL1PrL2DramDirectoryMSI::Dram↩
DirectoryCntlr::processUpgradeReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::process↩
WbRepFromL2Cache(), ParametricDramDirectoryMSI::CacheCntlr::processWbReqFromDramDirectory(), and Pr↩
L1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache().

6.322.4.2 getDataBuf()

```
Byte* PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataBuf ( )  [inline]
```

Definition at line 81 of file shmem_msg.h.

References m_data_buf.

Referenced by ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork(), DramCntlrInterface↩
::handleMsgFromTagDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDRAMReply(), Parametric↩
DramDirectoryMSI::CacheCntlr::processExRepFromDramDirectory(), PrL1PrL2DramDirectoryMSI::Dram↩
DirectoryCntlr::processFlushRepFromL2Cache(), ParametricDramDirectoryMSI::CacheCntlr::processShRep↩
FromDramDirectory(), ParametricDramDirectoryMSI::CacheCntlr::processUpgradeRepFromDramDirectory(), Pr↩
L1PrL2DramDirectoryMSI::DramDirectoryCntlr::processUpgradeReqFromL2Cache(), PrL1PrL2DramDirectory↩
MSI::DramDirectoryCntlr::processWbRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr↩
::processWbReqFromL2Cache(), and PrL1PrL2DramDirectoryMSI::ShmemReq::ShmemReq().

6.322.4.3 getDataLength()

```
UInt32 PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataLength ( ) [inline]
```

Definition at line 82 of file shmem_msg.h.

References m_data_length.

Referenced by ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork().

6.322.4.4 getModeledLength()

```
UInt32 PrL1PrL2DramDirectoryMSI::ShmemMsg::getModeledLength ( )
```

Definition at line 87 of file shmem_msg.cc.

References DRAM_READ_REP, DRAM_READ_REQ, DRAM_WRITE_REQ, EX_REP, EX_REQ, FLUSH_REP, F↩
LUSH_REQ, INV_REP, INV_REQ, LOG_PRINT_ERROR, m_data_length, m_msg_type, SH_REP, SH_REQ, U↩
PGRADE_REP, UPGRADE_REQ, WB_REP, and WB_REQ.

6.322.4.5 getMsgLen()

```
UInt32 PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgLen ( )
```

Definition at line 81 of file shmem_msg.cc.

References m_data_length.

Referenced by ParametricDramDirectoryMSI::MemoryManager::broadcastMsg(), makeMsgBuf(), and Parametric↩
DramDirectoryMSI::MemoryManager::sendMsg().

6.322.4.6 getMsgType()

```
msg_t PrL1PrL2DramDirectoryMSI::ShmemMsg::getMsgType ( ) [inline]
```

Definition at line 76 of file shmem_msg.h.

References m_msg_type.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::handleMsgFromDRAM(), ParametricDram↩
DirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr↩
::handleMsgFromL2Cache(), ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork(), Dram↩
CntlrInterface::handleMsgFromTagDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDRA↩
MReply(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), PrL1PrL2Dram↩
DirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectory↩
Cntlr::processNextReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbRepFrom↩
L2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache(), and PrL1PrL2↩
DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache().

6.322.4.7 getPerf()

```
ShmemPerf* PrL1PrL2DramDirectoryMSI::ShmemMsg::getPerf ( ) [inline]
```

Definition at line 88 of file shmem_msg.h.

References m_perf.

Referenced by ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork(), DramCntlrInterface::handleMsgFromTagDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDRAMReply(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), ParametricDramDirectoryMSI::CacheCntlr::processFlushReqFromDramDirectory(), ParametricDramDirectoryMSI::CacheCntlr::processInvReqFromDramDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processShReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processUpgradeReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbRepFromL2Cache(), ParametricDramDirectoryMSI::CacheCntlr::processWbReqFromDramDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::updateShmemPerf().

6.322.4.8 getReceiverMemComponent()

```
MemComponent::component_t PrL1PrL2DramDirectoryMSI::ShmemMsg::getReceiverMemComponent ( )
[inline]
```

Definition at line 78 of file shmem_msg.h.

References m_receiver_mem_component.

Referenced by ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork().

6.322.4.9 getRequester()

```
core_id_t PrL1PrL2DramDirectoryMSI::ShmemMsg::getRequester ( ) [inline]
```

Definition at line 79 of file shmem_msg.h.

References m_requester.

Referenced by DramCntlrInterface::handleMsgFromTagDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDirectoryEntryAllocationReq(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDRAMReply(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), ParametricDramDirectoryMSI::CacheCntlr::processFlushReqFromDramDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache(), ParametricDramDirectoryMSI::CacheCntlr::processInvReqFromDramDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processNullifyReq(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processShReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processUpgradeReqFromL2Cache(), ParametricDramDirectoryMSI::CacheCntlr::processWbReqFromDramDirectory(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache().

6.322.4.10 getSenderMemComponent()

```
MemComponent::component_t PrL1PrL2DramDirectoryMSI::ShmemMsg::getSenderMemComponent ( )  [inline]
```

Definition at line 77 of file shmem_msg.h.

References m_sender_mem_component.

Referenced by ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork().

6.322.4.11 getShmemMsg()

```
ShmemMsg * PrL1PrL2DramDirectoryMSI::ShmemMsg::getShmemMsg (
            Byte * msg_buf,
            ShmemPerf * perf )  [static]
```

Definition at line 54 of file shmem_msg.cc.

References ShmemMsg().

Referenced by ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork().

6.322.4.12 getWhere()

```
HitWhere::where_t PrL1PrL2DramDirectoryMSI::ShmemMsg::getWhere ( )  [inline]
```

Definition at line 83 of file shmem_msg.h.

References m_where.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), and PrL1PrL2Dram↩DirectoryMSI::DramDirectoryCntlr::processDRAMReply().

6.322.4.13 makeMsgBuf()

```
Byte * PrL1PrL2DramDirectoryMSI::ShmemMsg::makeMsgBuf ( )
```

Definition at line 67 of file shmem_msg.cc.

References getMsgLen(), LOG_ASSERT_ERROR, m_data_buf, and m_data_length.

Referenced by ParametricDramDirectoryMSI::MemoryManager::broadcastMsg(), and ParametricDramDirectory↩MSI::MemoryManager::sendMsg().

6.322.4.14 setDataBuf()

```
void PrL1PrL2DramDirectoryMSI::ShmemMsg::setDataBuf (
              Byte * data_buf )  [inline]
```

Definition at line 85 of file shmem_msg.h.

References m_data_buf.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache().

6.322.4.15 setWhere()

```
void PrL1PrL2DramDirectoryMSI::ShmemMsg::setWhere (
              HitWhere::where_t where )  [inline]
```

Definition at line 86 of file shmem_msg.h.

References m_where.

Referenced by ParametricDramDirectoryMSI::MemoryManager::sendMsg().

6.322.5 Member Data Documentation

6.322.5.1 m_address

```
IntPtr PrL1PrL2DramDirectoryMSI::ShmemMsg::m_address  [private]
```

Definition at line 49 of file shmem_msg.h.

Referenced by getAddress().

6.322.5.2 m_data_buf

```
Byte* PrL1PrL2DramDirectoryMSI::ShmemMsg::m_data_buf  [private]
```

Definition at line 50 of file shmem_msg.h.

Referenced by getDataBuf(), makeMsgBuf(), and setDataBuf().

6.322.5.3 m_data_length

```
UInt32 PrL1PrL2DramDirectoryMSI::ShmemMsg::m_data_length  [private]
```

Definition at line 51 of file shmem_msg.h.

Referenced by getDataLength(), getModeledLength(), getMsgLen(), and makeMsgBuf().

6.322.5.4 m_msg_type

```
msg_t PrL1PrL2DramDirectoryMSI::ShmemMsg::m_msg_type  [private]
```

Definition at line 44 of file shmem_msg.h.

Referenced by getModeledLength(), and getMsgType().

6.322.5.5 m_perf

```
ShmemPerf* PrL1PrL2DramDirectoryMSI::ShmemMsg::m_perf  [private]
```

Definition at line 52 of file shmem_msg.h.

Referenced by getPerf().

6.322.5.6 m_receiver_mem_component

```
MemComponent::component_t PrL1PrL2DramDirectoryMSI::ShmemMsg::m_receiver_mem_component  [private]
```

Definition at line 46 of file shmem_msg.h.

Referenced by getReceiverMemComponent().

6.322.5.7 m_requester

```
core_id_t PrL1PrL2DramDirectoryMSI::ShmemMsg::m_requester  [private]
```

Definition at line 47 of file shmem_msg.h.

Referenced by getRequester().

6.322.5.8 m_sender_mem_component

MemComponent::component_t PrL1PrL2DramDirectoryMSI::ShmemMsg::m_sender_mem_component [private]

Definition at line 45 of file shmem_msg.h.

Referenced by getSenderMemComponent().

6.322.5.9 m_where

HitWhere::where_t PrL1PrL2DramDirectoryMSI::ShmemMsg::m_where [private]

Definition at line 48 of file shmem_msg.h.

Referenced by getWhere(), and setWhere().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **shmem_msg.h**
- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **shmem_msg.cc**

6.323 ShmemPerf Class Reference

`#include <shmem_perf.h>`

Public Types

- enum **shmem_times_type_t** {
  NOC_BASE, **NOC_QUEUE**, **TD_ACCESS**, **INV_IMBALANCE**,
  REMOTE_CACHE_INV, **REMOTE_CACHE_FWD**, **REMOTE_CACHE_WB**, **PENDING_HIT**,
  NUCA_TAGS, **NUCA_QUEUE**, **NUCA_BUS**, **NUCA_DATA**,
  DRAM_CACHE, **DRAM_CACHE_TAGS**, **DRAM_CACHE_QUEUE**, **DRAM_CACHE_BUS**,
  DRAM_CACHE_DATA, **DRAM**, **DRAM_QUEUE**, **DRAM_BUS**,
  DRAM_DEVICE, **UNKNOWN**, **NUM_SHMEM_TIMES** }

Public Member Functions

- **ShmemPerf** ()
- void **disable** ()
- void **reset** (**SubsecondTime** time, **core_id_t** core_id)
- void **updateTime** (**SubsecondTime** time, **shmem_times_type_t** reason= **UNKNOWN**)
- void **updatePacket** (**NetPacket** &packet)
- void **add** (**ShmemPerf** *perf)
- **core_id_t** **getCore** () const
- **SubsecondTime** **getInitialTime** () const
- **SubsecondTime** & **getComponent** (**shmem_times_type_t** reason)

Private Attributes

- **core_id_t m_core_id**
- **SubsecondTime m_time_begin**
- **SubsecondTime m_time_last**
- std::vector< **SubsecondTime** > **m_times**

6.323.1 Detailed Description

Definition at line 10 of file shmem_perf.h.

6.323.2 Member Enumeration Documentation

6.323.2.1 shmem_times_type_t

enum **ShmemPerf::shmem_times_type_t**

Enumerator

NOC_BASE	
NOC_QUEUE	
TD_ACCESS	
INV_IMBALANCE	
REMOTE_CACHE_INV	
REMOTE_CACHE_FWD	
REMOTE_CACHE_WB	
PENDING_HIT	
NUCA_TAGS	
NUCA_QUEUE	
NUCA_BUS	
NUCA_DATA	
DRAM_CACHE	
DRAM_CACHE_TAGS	
DRAM_CACHE_QUEUE	
DRAM_CACHE_BUS	
DRAM_CACHE_DATA	
DRAM	
DRAM_QUEUE	
DRAM_BUS	
DRAM_DEVICE	
UNKNOWN	
NUM_SHMEM_TIMES	

Definition at line 13 of file shmem_perf.h.

6.323.3 Constructor & Destructor Documentation

6.323.3.1 ShmemPerf()

```
ShmemPerf::ShmemPerf ( )
```

Definition at line 40 of file shmem_perf.cc.

6.323.4 Member Function Documentation

6.323.4.1 add()

```
void ShmemPerf::add (
            ShmemPerf * perf )
```

Definition at line 85 of file shmem_perf.cc.

References getComponent(), m_times, and NUM_SHMEM_TIMES.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::updateUncoreStatistics().

6.323.4.2 disable()

```
void ShmemPerf::disable ( )
```

Definition at line 48 of file shmem_perf.cc.

References m_time_last, and SubsecondTime::MaxTime().

Referenced by ParametricDramDirectoryMSI::CacheCntlr::updateUncoreStatistics().

6.323.4.3 getComponent()

```
SubsecondTime& ShmemPerf::getComponent (
            shmem_times_type_t reason )  [inline]
```

Definition at line 48 of file shmem_perf.h.

References m_times.

Referenced by add(), and ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr().

6.323.4.4 getCore()

```
core_id_t ShmemPerf::getCore ( ) const  [inline]
```

Definition at line 46 of file shmem_perf.h.

References m_core_id.

6.323.4.5 getInitialTime()

```
SubsecondTime ShmemPerf::getInitialTime ( ) const  [inline]
```

Definition at line 47 of file shmem_perf.h.

References m_time_begin.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::updateUncoreStatistics().

6.323.4.6 reset()

```
void ShmemPerf::reset (
            SubsecondTime time,
            core_id_t core_id )
```

Definition at line 54 of file shmem_perf.cc.

References m_core_id, m_time_begin, m_time_last, m_times, NUM_SHMEM_TIMES, and SubsecondTime::Zero().

Referenced by ParametricDramDirectoryMSI::CacheCntlr::initiateDirectoryAccess(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache(), and ParametricDramDirectoryMSI::CacheCntlr::updateUncoreStatistics().

6.323.4.7 updatePacket()

```
void ShmemPerf::updatePacket (
            NetPacket & packet )
```

Definition at line 75 of file shmem_perf.cc.

References m_time_last, m_times, NOC_BASE, NOC_QUEUE, NetPacket::queue_delay, NetPacket::time, and updateTime().

Referenced by ParametricDramDirectoryMSI::MemoryManager::handleMsgFromNetwork().

6.323.4.8 updateTime()

```
void ShmemPerf::updateTime (
            SubsecondTime time,
            shmem_times_type_t reason = UNKNOWN )
```

Definition at line 63 of file shmem_perf.cc.

References LOG_ASSERT_ERROR, m_time_last, m_times, and NUM_SHMEM_TIMES.

Referenced by NucaCache::accessDataArray(), DramCache::accessDataArray(), ParametricDramDirectoryMSI::MemoryManager::broadcastMsg(), DramCache::doAccess(), DramPerfModelConstant::getAccessLatency(), DramPerfModelNormal::getAccessLatency(), DramPerfModelReadWrite::getAccessLatency(), ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), DramCntlrInterface::handleMsgFromTagDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), ParametricDramDirectoryMSI::CacheCntlr::processFlushReqFromDramDirectory(), ParametricDramDirectoryMSI::CacheCntlr::processInvReqFromDramDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbRepFromL2Cache(), ParametricDramDirectoryMSI::CacheCntlr::processWbReqFromDramDirectory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache(), NucaCache::read(), ParametricDramDirectoryMSI::MemoryManager::sendMsg(), updatePacket(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::updateShmemPerf().

6.323.5 Member Data Documentation

6.323.5.1 m_core_id

```
core_id_t ShmemPerf::m_core_id [private]
```

Definition at line 51 of file shmem_perf.h.

Referenced by getCore(), and reset().

6.323.5.2 m_time_begin

```
SubsecondTime ShmemPerf::m_time_begin [private]
```

Definition at line 52 of file shmem_perf.h.

Referenced by getInitialTime(), and reset().

6.323.5.3 m_time_last

```
SubsecondTime ShmemPerf::m_time_last [private]
```

Definition at line 53 of file shmem_perf.h.

Referenced by disable(), reset(), updatePacket(), and updateTime().

6.323.5.4 m_times

`std::vector< SubsecondTime> ShmemPerf::m_times [private]`

Definition at line 54 of file shmem_perf.h.

Referenced by add(), getComponent(), reset(), updatePacket(), and updateTime().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **shmem_perf.h**
- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **shmem_perf.cc**

6.324 ShmemPerfModel Class Reference

`#include <shmem_perf_model.h>`

Public Types

- enum **Thread_t** { **_USER_THREAD** = 0, **_SIM_THREAD**, **NUM_CORE_THREADS** }

Public Member Functions

- **ShmemPerfModel** ()
- ~**ShmemPerfModel** ()
- void **setElapsedTime** (**Thread_t** thread_num, **SubsecondTime** time)
- **SubsecondTime** **getElapsedTime** (**Thread_t** thread_num)
- void **incrElapsedTime** (**SubsecondTime** time, **Thread_t** thread_num)
- void **updateElapsedTime** (**SubsecondTime** time, **Thread_t** thread_num)
- void **incrTotalMemoryAccessLatency** (**SubsecondTime** shmem_time)
- void **enable** ()
- void **disable** ()
- bool **isEnabled** ()

Private Attributes

- **SubsecondTime** **m_elapsed_time** [**NUM_CORE_THREADS**]
- bool **m_enabled**
- **RwLock** **m_shmem_perf_model_lock**
- **UInt64** **m_num_memory_accesses**
- **SubsecondTime** **m_total_memory_access_latency**

6.324.1 Detailed Description

Definition at line 10 of file shmem_perf_model.h.

6.324.2 Member Enumeration Documentation

6.324.2.1 Thread_t

`enum ShmemPerfModel::Thread_t`

Enumerator

_USER_THREAD	
_SIM_THREAD	
NUM_CORE_THREADS	

Definition at line 13 of file shmem_perf_model.h.

6.324.3 Constructor & Destructor Documentation

6.324.3.1 ShmemPerfModel()

```
ShmemPerfModel::ShmemPerfModel ( )
```

Definition at line 8 of file shmem_perf_model.cc.

References m_elapsed_time, NUM_CORE_THREADS, and SubsecondTime::Zero().

6.324.3.2 ~ShmemPerfModel()

```
ShmemPerfModel::~ShmemPerfModel ( )
```

Definition at line 17 of file shmem_perf_model.cc.

6.324.4 Member Function Documentation

6.324.4.1 disable()

```
void ShmemPerfModel::disable ( )  [inline]
```

Definition at line 41 of file shmem_perf_model.h.

References m_enabled.

Referenced by Core::disablePerformanceModels().

6.324.4.2 enable()

```
void ShmemPerfModel::enable ( )  [inline]
```

Definition at line 40 of file shmem_perf_model.h.

References m_enabled.

Referenced by Core::enablePerformanceModels().

6.324.4.3 getElapsedTime()

```
SubsecondTime ShmemPerfModel::getElapsedTime (
            Thread_t thread_num )
```

Definition at line 31 of file shmem_perf_model.cc.

References m_elapsed_time.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::accessDRAM(), ParametricDramDirectoryMSI::↩ MemoryManager::broadcastMsg(), MemoryManagerBase::coreInitiateMemoryAccessFast(), ParametricDram↩ DirectoryMSI::CacheCntlr::doPrefetch(), ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDram↩ Directory(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::handleMsgFromL2Cache(), DramCntlrInterface↩ ::handleMsgFromTagDirectory(), Core::initiateMemoryAccess(), ParametricDramDirectoryMSI::CacheCntlr↩ ::insertCacheBlock(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDirectoryEntryAllocationReq(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), ParametricDramDirectory↩ MSI::CacheCntlr::processMemOpFromCore(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReq↩ FromPrevCache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbRepFromL2Cache(), PrL1↩ PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache(), ParametricDramDirectoryMSI::↩ CacheCntlr::retrieveCacheBlock(), ParametricDramDirectoryMSI::MemoryManager::sendMsg(), ParametricDram↩ DirectoryMSI::CacheCntlr::updateCounters(), and ParametricDramDirectoryMSI::CacheCntlr::writeCacheBlock().

6.324.4.4 incrElapsedTime()

```
void ShmemPerfModel::incrElapsedTime (
            SubsecondTime time,
            Thread_t thread_num )
```

Definition at line 49 of file shmem_perf_model.cc.

References atomic_add_subsecondtime(), itostr(), LOG_ASSERT_ERROR, LOG_PRINT, and m_elapsed_time.

Referenced by MemoryManagerFast::coreInitiateMemoryAccess(), PrL1PrL2DramDirectoryMSI::DramDirectory↩ Cache::getDirectoryEntry(), DramCntlrInterface::handleMsgFromTagDirectory(), ParametricDramDirectoryMS↩ I::MemoryManager::incrElapsedTime(), ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore(), PrL1PrL2DramDirectoryMSI::DramDirectoryCache::replaceDirectoryEntry(), and PrL1PrL2DramDirectoryMSI::↩ DramDirectoryCntlr::retrieveDataAndSendToL2Cache().

6.324.4.5 incrTotalMemoryAccessLatency()

```
void ShmemPerfModel::incrTotalMemoryAccessLatency (
              SubsecondTime shmem_time )
```

Definition at line 66 of file shmem_perf_model.cc.

References atomic_add_subsecondtime(), m_enabled, m_num_memory_accesses, and m_total_memory_↩ access_latency.

Referenced by Core::initiateMemoryAccess().

6.324.4.6 isEnabled()

```
bool ShmemPerfModel::isEnabled ( )  [inline]
```

Definition at line 42 of file shmem_perf_model.h.

References m_enabled.

6.324.4.7 setElapsedTime()

```
void ShmemPerfModel::setElapsedTime (
              Thread_t thread_num,
              SubsecondTime time )
```

Definition at line 21 of file shmem_perf_model.cc.

References itostr(), LOG_PRINT, m_elapsed_time, and NUM_CORE_THREADS.

Referenced by MemoryManagerBase::coreInitiateMemoryAccessFast(), ParametricDramDirectoryMSI::Cache↩ Cntlr::doPrefetch(), ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), Parametric↩ DramDirectoryMSI::MemoryManager::handleMsgFromNetwork(), Core::initiateMemoryAccess(), and PrL1PrL2↩ DramDirectoryMSI::DramDirectoryCntlr::processNextReqFromL2Cache().

6.324.4.8 updateElapsedTime()

```
void ShmemPerfModel::updateElapsedTime (
              SubsecondTime time,
              Thread_t thread_num )
```

Definition at line 39 of file shmem_perf_model.cc.

References itostr(), LOG_PRINT, and m_elapsed_time.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), PrL1PrL2↩ DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::Dram↩ DirectoryCntlr::processWbRepFromL2Cache(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::process↩ WbReqFromL2Cache().

6.324.5 Member Data Documentation

6.324.5.1 m_elapsed_time

```
SubsecondTime ShmemPerfModel::m_elapsed_time[ NUM_CORE_THREADS]  [private]
```

Definition at line 21 of file shmem_perf_model.h.

Referenced by getElapsedTime(), incrElapsedTime(), setElapsedTime(), ShmemPerfModel(), and update↩ ElapsedTime().

6.324.5.2 m_enabled

```
bool ShmemPerfModel::m_enabled  [private]
```

Definition at line 22 of file shmem_perf_model.h.

Referenced by disable(), enable(), incrTotalMemoryAccessLatency(), and isEnabled().

6.324.5.3 m_num_memory_accesses

```
UInt64 ShmemPerfModel::m_num_memory_accesses  [private]
```

Definition at line 25 of file shmem_perf_model.h.

Referenced by incrTotalMemoryAccessLatency().

6.324.5.4 m_shmem_perf_model_lock

```
RwLock ShmemPerfModel::m_shmem_perf_model_lock  [private]
```

Definition at line 23 of file shmem_perf_model.h.

6.324.5.5 m_total_memory_access_latency

SubsecondTime ShmemPerfModel::m_total_memory_access_latency [private]

Definition at line 26 of file shmem_perf_model.h.

Referenced by incrTotalMemoryAccessLatency().

The documentation for this class was generated from the following files:

- common/performance_model/ **shmem_perf_model.h**
- common/performance_model/ **shmem_perf_model.cc**

6.325 PrL1PrL2DramDirectoryMSI::ShmemReq Class Reference

#include <shmem_req.h>

Public Member Functions

- **ShmemReq** (**ShmemMsg** *shmem_msg, **SubsecondTime** time)
- ~**ShmemReq** ()
- **ShmemMsg** * **getShmemMsg** () const
- **SubsecondTime** **getTime** () const
- bool **getWaitForData** () const
- **core_id_t** **getForwardingFrom** () const
- bool **isForwarding** () const
- void **setTime** (**SubsecondTime** time)
- void **updateTime** (**SubsecondTime** time)
- void **setWaitForData** (bool wait)
- void **setForwardingFrom** (**core_id_t** core_id)

Private Attributes

- **ShmemMsg** * **m_shmem_msg**
- **SubsecondTime** **m_time**
- bool **m_wait_for_data**
- **core_id_t** **m_forwarding_from**

6.325.1 Detailed Description

Definition at line 9 of file shmem_req.h.

6.325.2 Constructor & Destructor Documentation

6.325.2.1 ShmemReq()

```
PrL1PrL2DramDirectoryMSI::ShmemReq::ShmemReq (
              ShmemMsg * shmem_msg,
              SubsecondTime time )
```

Definition at line 6 of file shmem_req.cc.

References PrL1PrL2DramDirectoryMSI::ShmemMsg::getDataBuf(), LOG_ASSERT_ERROR, and m_shmem_↩ msg.

6.325.2.2 ∼ShmemReq()

```
PrL1PrL2DramDirectoryMSI::ShmemReq::∼ShmemReq ( )
```

Definition at line 17 of file shmem_req.cc.

References m_shmem_msg.

6.325.3 Member Function Documentation

6.325.3.1 getForwardingFrom()

```
core_id_t PrL1PrL2DramDirectoryMSI::ShmemReq::getForwardingFrom ( ) const  [inline]
```

Definition at line 24 of file shmem_req.h.

References m_forwarding_from.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache(), and PrL1PrL2↩ DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache().

6.325.3.2 getShmemMsg()

```
ShmemMsg* PrL1PrL2DramDirectoryMSI::ShmemReq::getShmemMsg ( ) const  [inline]
```

Definition at line 21 of file shmem_req.h.

References m_shmem_msg.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDirectoryEntryAllocationReq(), Pr↩ L1PrL2DramDirectoryMSI::DramDirectoryCntlr::processDRAMReply(), PrL1PrL2DramDirectoryMSI::Dram↩ DirectoryCntlr::processExReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlush↩ RepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache(), PrL1Pr↩ L2DramDirectoryMSI::DramDirectoryCntlr::processNextReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::Dram↩ DirectoryCntlr::processNullifyReq(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processShReqFromL2↩ Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processUpgradeReqFromL2Cache(), PrL1PrL2Dram↩ DirectoryMSI::DramDirectoryCntlr::processWbRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectory↩ Cntlr::processWbReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSend↩ ToL2Cache(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::updateShmemPerf().

6.325.3.3 getTime()

```
SubsecondTime PrL1PrL2DramDirectoryMSI::ShmemReq::getTime ( ) const  [inline]
```

Definition at line 22 of file shmem_req.h.

References m_time.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), PrL1PrL2↩DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::Dram↩DirectoryCntlr::processNextReqFromL2Cache(), PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWb↩RepFromL2Cache(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processWbReqFromL2Cache().

6.325.3.4 getWaitForData()

```
bool PrL1PrL2DramDirectoryMSI::ShmemReq::getWaitForData ( ) const  [inline]
```

Definition at line 23 of file shmem_req.h.

References m_wait_for_data.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache().

6.325.3.5 isForwarding()

```
bool PrL1PrL2DramDirectoryMSI::ShmemReq::isForwarding ( ) const  [inline]
```

Definition at line 25 of file shmem_req.h.

References INVALID_CORE_ID, and m_forwarding_from.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache().

6.325.3.6 setForwardingFrom()

```
void PrL1PrL2DramDirectoryMSI::ShmemReq::setForwardingFrom (
            core_id_t core_id )  [inline]
```

Definition at line 34 of file shmem_req.h.

References m_forwarding_from.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache().

6.325.3.7 setTime()

```
void PrL1PrL2DramDirectoryMSI::ShmemReq::setTime (
              SubsecondTime time )  [inline]
```

Definition at line 27 of file shmem_req.h.

References m_time.

6.325.3.8 setWaitForData()

```
void PrL1PrL2DramDirectoryMSI::ShmemReq::setWaitForData (
             bool wait )  [inline]
```

Definition at line 33 of file shmem_req.h.

References m_wait_for_data.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::retrieveDataAndSendToL2Cache().

6.325.3.9 updateTime()

```
void PrL1PrL2DramDirectoryMSI::ShmemReq::updateTime (
              SubsecondTime time )  [inline]
```

Definition at line 28 of file shmem_req.h.

References m_time.

Referenced by PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::processFlushRepFromL2Cache(), PrL1PrL2↩ DramDirectoryMSI::DramDirectoryCntlr::processInvRepFromL2Cache(), PrL1PrL2DramDirectoryMSI::Dram↩ DirectoryCntlr::processWbRepFromL2Cache(), and PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr::process↩ WbReqFromL2Cache().

6.325.4 Member Data Documentation

6.325.4.1 m_forwarding_from

```
core_id_t PrL1PrL2DramDirectoryMSI::ShmemReq::m_forwarding_from  [private]
```

Definition at line 15 of file shmem_req.h.

Referenced by getForwardingFrom(), isForwarding(), and setForwardingFrom().

6.325.4.2 m_shmem_msg

```
ShmemMsg* PrL1PrL2DramDirectoryMSI::ShmemReq::m_shmem_msg  [private]
```

Definition at line 12 of file shmem_req.h.

Referenced by getShmemMsg(), ShmemReq(), and ~ShmemReq().

6.325.4.3 m_time

```
SubsecondTime PrL1PrL2DramDirectoryMSI::ShmemReq::m_time  [private]
```

Definition at line 13 of file shmem_req.h.

Referenced by getTime(), setTime(), and updateTime().

6.325.4.4 m_wait_for_data

```
bool PrL1PrL2DramDirectoryMSI::ShmemReq::m_wait_for_data  [private]
```

Definition at line 14 of file shmem_req.h.

Referenced by getWaitForData(), and setWaitForData().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **shmem_req.h**
- common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/ **shmem_req.cc**

6.326 SimBarrier Class Reference

```
#include <sync_server.h>
```

Public Member Functions

- **SimBarrier** (**UInt32** count)
- ~**SimBarrier** ()
- **SubsecondTime wait** (**thread_id_t** thread_id, **SubsecondTime** time)

Private Types

- typedef std::queue< **thread_id_t** > **ThreadQueue**

Private Attributes

- **ThreadQueue m_waiting**
- **UInt32 m_count**

6.326.1 Detailed Description

Definition at line 69 of file sync_server.h.

6.326.2 Member Typedef Documentation

6.326.2.1 ThreadQueue

```
typedef std::queue< thread_id_t >  SimBarrier::ThreadQueue  [private]
```

Definition at line 78 of file sync_server.h.

6.326.3 Constructor & Destructor Documentation

6.326.3.1 SimBarrier()

```
SimBarrier::SimBarrier (
              UInt32 count )
```

Definition at line 144 of file sync_server.cc.

6.326.3.2 ~SimBarrier()

```
SimBarrier::~SimBarrier ( )
```

Definition at line 149 of file sync_server.cc.

References m_waiting.

6.326.4 Member Function Documentation

6.326.4.1 wait()

```
SubsecondTime SimBarrier::wait (
            thread_id_t thread_id,
            SubsecondTime time )
```

Definition at line 161 of file sync_server.cc.

References m_count, m_waiting, and ThreadManager::STALL_BARRIER.

Referenced by SyncServer::barrierWait().

6.326.5 Member Data Documentation

6.326.5.1 m_count

```
UInt32 SimBarrier::m_count  [private]
```

Definition at line 81 of file sync_server.h.

Referenced by wait().

6.326.5.2 m_waiting

```
ThreadQueue SimBarrier::m_waiting  [private]
```

Definition at line 79 of file sync_server.h.

Referenced by wait(), and ~SimBarrier().

The documentation for this class was generated from the following files:

- common/system/ **sync_server.h**
- common/system/ **sync_server.cc**

6.327 SimCond Class Reference

```
#include <sync_server.h>
```

Classes

- class **CondWaiter**

Public Member Functions

- **SimCond** ()
- ∼**SimCond** ()
- **SubsecondTime wait** (**thread_id_t** thread_id, **SubsecondTime** time, **SimMutex** ∗mux)
- **thread_id_t signal** (**thread_id_t** thread_id, **SubsecondTime** time)
- void **broadcast** (**thread_id_t** thread_id, **SubsecondTime** time)

Private Types

- typedef std::queue< **CondWaiter** > **ThreadQueue**

Private Attributes

- **ThreadQueue m_waiting**

6.327.1 Detailed Description

Definition at line 44 of file sync_server.h.

6.327.2 Member Typedef Documentation

6.327.2.1 ThreadQueue

```
typedef std::queue< CondWaiter > SimCond::ThreadQueue [private]
```

Definition at line 65 of file sync_server.h.

6.327.3 Constructor & Destructor Documentation

6.327.3.1 SimCond()

```
SimCond::SimCond ( )
```

Definition at line 85 of file sync_server.cc.

6.327.3.2 ~SimCond()

```
SimCond::~SimCond ( )
```

Definition at line 86 of file sync_server.cc.

References m_waiting.

6.327.4 Member Function Documentation

6.327.4.1 broadcast()

```
void SimCond::broadcast (
              thread_id_t thread_id,
              SubsecondTime time )
```

Definition at line 133 of file sync_server.cc.

References SimMutex::lock_async(), SimCond::CondWaiter::m_mutex, SimCond::CondWaiter::m_thread_id, and m_waiting.

Referenced by SyncServer::condBroadcast().

6.327.4.2 signal()

```
thread_id_t SimCond::signal (
              thread_id_t thread_id,
              SubsecondTime time )
```

Definition at line 109 of file sync_server.cc.

References INVALID_THREAD_ID, SimMutex::lock_async(), SimCond::CondWaiter::m_mutex, SimCond::CondWaiter::m_thread_id, and m_waiting.

Referenced by SyncServer::condSignal().

6.327.4.3 wait()

```
SubsecondTime SimCond::wait (
              thread_id_t thread_id,
              SubsecondTime time,
              SimMutex * mux )
```

Definition at line 100 of file sync_server.cc.

References m_waiting, ThreadManager::STALL_COND, and SimMutex::unlock().

Referenced by SyncServer::condWait().

6.327.5 Member Data Documentation

6.327.5.1 m_waiting

ThreadQueue SimCond::m_waiting [private]

Definition at line 66 of file sync_server.h.

Referenced by broadcast(), signal(), wait(), and ∼SimCond().

The documentation for this class was generated from the following files:

- common/system/ **sync_server.h**
- common/system/ **sync_server.cc**

6.328 SimFutex Class Reference

#include <syscall_server.h>

Classes

- struct **Waiter**

Public Types

- typedef std::list< **Waiter** > **ThreadQueue**

Public Member Functions

- **SimFutex** ()
- ∼**SimFutex** ()
- bool **enqueueWaiter** (**thread_id_t** thread_id, int mask, **SubsecondTime** time, **SubsecondTime** timeout↩ _time, **SubsecondTime** &time_end)
- **thread_id_t** **dequeueWaiter** (**thread_id_t** thread_by, int mask, **SubsecondTime** time)
- **thread_id_t** **requeueWaiter** (**SimFutex** ∗requeue_futex)
- void **wakeTimedOut** (**SubsecondTime** time)
- **SubsecondTime** **getNextTimeout** (**SubsecondTime** time)

Private Attributes

- **ThreadQueue** **m_waiting**

6.328.1 Detailed Description

Definition at line 19 of file syscall_server.h.

6.328.2 Member Typedef Documentation

6.328.2.1 ThreadQueue

```
typedef std::list< Waiter>  SimFutex::ThreadQueue
```

Definition at line 31 of file syscall_server.h.

6.328.3 Constructor & Destructor Documentation

6.328.3.1 SimFutex()

```
SimFutex::SimFutex ( )
```

Definition at line 363 of file syscall_server.cc.

6.328.3.2 ~SimFutex()

```
SimFutex::~SimFutex ( )
```

Definition at line 366 of file syscall_server.cc.

References m_waiting.

6.328.4 Member Function Documentation

6.328.4.1 dequeueWaiter()

```
thread_id_t SimFutex::dequeueWaiter (
             thread_id_t thread_by,
            int mask,
             SubsecondTime time )
```

Definition at line 389 of file syscall_server.cc.

References INVALID_THREAD_ID, and m_waiting.

Referenced by SyscallServer::futexCmpRequeue(), and SyscallServer::wakeFutexOne().

6.328.4.2 enqueueWaiter()

```
bool SimFutex::enqueueWaiter (
             thread_id_t thread_id,
            int mask,
             SubsecondTime time,
             SubsecondTime timeout_time,
             SubsecondTime & time_end )
```

Definition at line 382 of file syscall_server.cc.

References m_waiting, and ThreadManager::STALL_FUTEX.

Referenced by SyscallServer::futexWait().

6.328.4.3 getNextTimeout()

```
SubsecondTime SimFutex::getNextTimeout (
             SubsecondTime time )
```

Definition at line 441 of file syscall_server.cc.

References m_waiting, and SubsecondTime::MaxTime().

6.328.4.4 requeueWaiter()

```
thread_id_t SimFutex::requeueWaiter (
             SimFutex * requeue_futex )
```

Definition at line 410 of file syscall_server.cc.

References INVALID_THREAD_ID, m_waiting, and SimFutex::Waiter::thread_id.

Referenced by SyscallServer::futexCmpRequeue().

6.328.4.5 wakeTimedOut()

```
void SimFutex::wakeTimedOut (
              SubsecondTime time )
```

Definition at line 424 of file syscall_server.cc.

References INVALID_THREAD_ID, and m_waiting.

6.328.5 Member Data Documentation

6.328.5.1 m_waiting

```
ThreadQueue SimFutex::m_waiting [private]
```

Definition at line 34 of file syscall_server.h.

Referenced by dequeueWaiter(), enqueueWaiter(), getNextTimeout(), requeueWaiter(), wakeTimedOut(), and ∼↩ SimFutex().

The documentation for this class was generated from the following files:

- common/system/ **syscall_server.h**
- common/system/ **syscall_server.cc**

6.329 SimMutex Class Reference

```
#include <sync_server.h>
```

Public Member Functions

- **SimMutex** ()
- ∼**SimMutex** ()
- bool **isLocked** (**thread_id_t** thread_id)
- **SubsecondTime** **lock** (**thread_id_t** thread_id, **SubsecondTime** time)
- bool **lock_async** (**thread_id_t** thread_id, **thread_id_t** thread_by, **SubsecondTime** time)
- **thread_id_t** **unlock** (**thread_id_t** thread_id, **SubsecondTime** time)

Static Public Attributes

- static const **thread_id_t** **NO_OWNER** = UINT_MAX

Private Types

- typedef std::queue< **thread_id_t** > **ThreadQueue**

Private Attributes

- **ThreadQueue m_waiting**
- **thread_id_t m_owner**

6.329.1 Detailed Description

Definition at line 17 of file sync_server.h.

6.329.2 Member Typedef Documentation

6.329.2.1 ThreadQueue

```
typedef std::queue< thread_id_t >  SimMutex::ThreadQueue  [private]
```

Definition at line 38 of file sync_server.h.

6.329.3 Constructor & Destructor Documentation

6.329.3.1 SimMutex()

```
SimMutex::SimMutex ( )
```

Definition at line 10 of file sync_server.cc.

6.329.3.2 ~SimMutex()

```
SimMutex::~SimMutex ( )
```

Definition at line 14 of file sync_server.cc.

References m_waiting.

6.329.4 Member Function Documentation

6.329.4.1 isLocked()

```
bool SimMutex::isLocked (
            thread_id_t thread_id )
```

Definition at line 28 of file sync_server.cc.

References m_owner, and NO_OWNER.

Referenced by SyncServer::mutexLock().

6.329.4.2 lock()

```
SubsecondTime SimMutex::lock (
            thread_id_t thread_id,
            SubsecondTime time )
```

Definition at line 38 of file sync_server.cc.

References m_owner, m_waiting, NO_OWNER, and ThreadManager::STALL_MUTEX.

Referenced by SyncServer::mutexLock().

6.329.4.3 lock_async()

```
bool SimMutex::lock_async (
            thread_id_t thread_id,
            thread_id_t thread_by,
            SubsecondTime time )
```

Definition at line 52 of file sync_server.cc.

References m_owner, m_waiting, and NO_OWNER.

Referenced by SimCond::broadcast(), and SimCond::signal().

6.329.4.4 unlock()

```
thread_id_t SimMutex::unlock (
            thread_id_t thread_id,
            SubsecondTime time )
```

Definition at line 67 of file sync_server.cc.

References m_owner, m_waiting, and NO_OWNER.

Referenced by SyncServer::mutexUnlock(), and SimCond::wait().

6.329.5 Member Data Documentation

6.329.5.1 m_owner

```
thread_id_t SimMutex::m_owner  [private]
```

Definition at line 41 of file sync_server.h.

Referenced by isLocked(), lock(), lock_async(), and unlock().

6.329.5.2 m_waiting

```
ThreadQueue SimMutex::m_waiting  [private]
```

Definition at line 40 of file sync_server.h.

Referenced by lock(), lock_async(), unlock(), and ~SimMutex().

6.329.5.3 NO_OWNER

```
const  thread_id_t SimMutex::NO_OWNER = UINT_MAX  [static]
```

Definition at line 20 of file sync_server.h.

Referenced by isLocked(), lock(), lock_async(), SyncServer::mutexUnlock(), and unlock().

The documentation for this class was generated from the following files:

- common/system/ **sync_server.h**
- common/system/ **sync_server.cc**

6.330 SimpleBimodalTable Class Reference

```
#include <simple_bimodal_table.h>
```

Inheritance diagram for SimpleBimodalTable:

Public Member Functions

- **SimpleBimodalTable** (**UInt32** entries)
- bool **predict** (**IntPtr** ip, **IntPtr** target)
- void **update** (bool predicted, bool actual, **IntPtr** ip, **IntPtr** target)
- void **reset** ()

Private Member Functions

- template<typename Addr >
  Addr **ilog2** (Addr **n**)

Private Attributes

- **UInt32 m_num_entries**
- **IntPtr m_mask**
- std::vector< **SaturatingPredictor**< 2 > > **m_table**

Additional Inherited Members

6.330.1 Detailed Description

Definition at line 9 of file simple_bimodal_table.h.

6.330.2 Constructor & Destructor Documentation

6.330.2.1 SimpleBimodalTable()

```
SimpleBimodalTable::SimpleBimodalTable (
              UInt32 entries )  [inline]
```

Definition at line 14 of file simple_bimodal_table.h.

References ilog2(), m_mask, m_num_entries, and reset().

6.330.3 Member Function Documentation

6.330.3.1 ilog2()

```
template<typename Addr >
Addr SimpleBimodalTable::ilog2 (
            Addr n )  [inline], [private]
```

Definition at line 55 of file simple_bimodal_table.h.

References n.

Referenced by SimpleBimodalTable().

6.330.3.2 predict()

```
bool SimpleBimodalTable::predict (
              IntPtr ip,
              IntPtr target )  [inline], [virtual]
```

Implements **BranchPredictor** (p. 125).

Definition at line 26 of file simple_bimodal_table.h.

References m_mask, and m_table.

Referenced by PentiumMBranchPredictor::predict().

6.330.3.3 reset()

```
void SimpleBimodalTable::reset ( )  [inline]
```

Definition at line 45 of file simple_bimodal_table.h.

References m_num_entries, and m_table.

Referenced by SimpleBimodalTable().

6.330.3.4 update()

```
void SimpleBimodalTable::update (
            bool predicted,
            bool actual,
             IntPtr ip,
             IntPtr target )  [inline], [virtual]
```

Implements **BranchPredictor** (p. 126).

Definition at line 32 of file simple_bimodal_table.h.

References m_mask, and m_table.

Referenced by PentiumMBranchPredictor::update().

6.330.4 Member Data Documentation

6.330.4.1 m_mask

```
IntPtr SimpleBimodalTable::m_mask [private]
```

Definition at line 65 of file simple_bimodal_table.h.

Referenced by predict(), SimpleBimodalTable(), and update().

6.330.4.2 m_num_entries

```
UInt32 SimpleBimodalTable::m_num_entries [private]
```

Definition at line 64 of file simple_bimodal_table.h.

Referenced by reset(), and SimpleBimodalTable().

6.330.4.3 m_table

```
std::vector< SaturatingPredictor<2> > SimpleBimodalTable::m_table [private]
```

Definition at line 66 of file simple_bimodal_table.h.

Referenced by predict(), reset(), and update().

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **simple_bimodal_table.h**

6.331 SimplePrefetcher Class Reference

```
#include <simple_prefetcher.h>
```

Inheritance diagram for SimplePrefetcher:

Public Member Functions

- **SimplePrefetcher** (String configName, **core_id_t core_id**, **UInt32 shared_cores**)
- virtual std::vector< **IntPtr** > **getNextAddress** (**IntPtr** current_address, **core_id_t core_id**)

Private Attributes

- const **core_id_t core_id**
- const **UInt32 shared_cores**
- const **UInt32 n_flows**
- const bool **flows_per_core**
- const **UInt32 num_prefetches**
- const bool **stop_at_page**
- **UInt32 n_flow_next**
- std::vector< std::vector< **IntPtr** > > **m_prev_address**

Additional Inherited Members

6.331.1 Detailed Description

Definition at line 6 of file simple_prefetcher.h.

6.331.2 Constructor & Destructor Documentation

6.331.2.1 SimplePrefetcher()

```
SimplePrefetcher::SimplePrefetcher (
             String configName,
              core_id_t core_id,
              UInt32 shared_cores )
```

Definition at line 10 of file simple_prefetcher.cc.

References flows_per_core, m_prev_address, n_flows, and shared_cores.

6.331.3 Member Function Documentation

6.331.3.1 getNextAddress()

```
std::vector< IntPtr > SimplePrefetcher::getNextAddress (
            IntPtr current_address,
            core_id_t core_id ) [virtual]
```

Implements **Prefetcher** (p. 938).

Definition at line 25 of file simple_prefetcher.cc.

References core_id, flows_per_core, m_prev_address, n_flow_next, n_flows, num_prefetches, PAGE_MASK, P↩AGE_SIZE, and stop_at_page.

6.331.4 Member Data Documentation

6.331.4.1 core_id

```
const core_id_t SimplePrefetcher::core_id [private]
```

Definition at line 13 of file simple_prefetcher.h.

Referenced by getNextAddress().

6.331.4.2 flows_per_core

```
const bool SimplePrefetcher::flows_per_core [private]
```

Definition at line 16 of file simple_prefetcher.h.

Referenced by getNextAddress(), and SimplePrefetcher().

6.331.4.3 m_prev_address

```
std::vector<std::vector< IntPtr> > SimplePrefetcher::m_prev_address [private]
```

Definition at line 20 of file simple_prefetcher.h.

Referenced by getNextAddress(), and SimplePrefetcher().

6.331.4.4 n_flow_next

```
UInt32 SimplePrefetcher::n_flow_next  [private]
```

Definition at line 19 of file simple_prefetcher.h.

Referenced by getNextAddress().

6.331.4.5 n_flows

```
const  UInt32 SimplePrefetcher::n_flows  [private]
```

Definition at line 15 of file simple_prefetcher.h.

Referenced by getNextAddress(), and SimplePrefetcher().

6.331.4.6 num_prefetches

```
const  UInt32 SimplePrefetcher::num_prefetches  [private]
```

Definition at line 17 of file simple_prefetcher.h.

Referenced by getNextAddress().

6.331.4.7 shared_cores

```
const  UInt32 SimplePrefetcher::shared_cores  [private]
```

Definition at line 14 of file simple_prefetcher.h.

Referenced by SimplePrefetcher().

6.331.4.8 stop_at_page

```
const bool SimplePrefetcher::stop_at_page  [private]
```

Definition at line 18 of file simple_prefetcher.h.

Referenced by getNextAddress().

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **simple_prefetcher.h**
- common/core/memory_subsystem/parametric_dram_directory_msi/ **simple_prefetcher.cc**

6.332 SimThread Class Reference

`#include <sim_thread.h>`

Inheritance diagram for SimThread:

Public Member Functions

- **SimThread** ()
- ∼**SimThread** ()
- void **spawn** ()

Private Member Functions

- void **run** ()

Static Private Member Functions

- static void **terminateFunc** (void ∗vp, **NetPacket** pkt)

Private Attributes

- **_Thread** ∗ **m_thread**

Additional Inherited Members

6.332.1 Detailed Description

Definition at line 8 of file sim_thread.h.

6.332.2 Constructor & Destructor Documentation

6.332.2.1 SimThread()

`SimThread::SimThread ( )`

Definition at line 9 of file sim_thread.cc.

6.332.2.2 ~SimThread()

```
SimThread::~SimThread ( )
```

Definition at line 14 of file sim_thread.cc.

References m_thread.

6.332.3 Member Function Documentation

6.332.3.1 run()

```
void SimThread::run ( )  [private], [virtual]
```

Implements **Runnable** (p. 1106).

Definition at line 19 of file sim_thread.cc.

References itostr(), LOG_PRINT, Network::netPullFromTransport(), Network::registerCallback(), CoreManager::↩
SIM_THREAD, SIM_THREAD_TERMINATE_THREADS, and terminateFunc().

6.332.3.2 spawn()

```
void SimThread::spawn ( )
```

Definition at line 48 of file sim_thread.cc.

References _Thread::create(), m_thread, and _Thread::run().

Referenced by SimThreadManager::spawnSimThreads().

6.332.3.3 terminateFunc()

```
void SimThread::terminateFunc (
            void * vp,
             NetPacket pkt )  [static], [private]
```

Definition at line 54 of file sim_thread.cc.

Referenced by run().

6.332.4 Member Data Documentation

6.332.4.1 m_thread

```
_Thread* SimThread::m_thread  [private]
```

Definition at line 21 of file sim_thread.h.

Referenced by spawn(), and ~SimThread().

The documentation for this class was generated from the following files:

- common/system/ **sim_thread.h**
- common/system/ **sim_thread.cc**

6.333 SimThreadManager Class Reference

```
#include <sim_thread_manager.h>
```

Public Member Functions

- **SimThreadManager** ()
- ~**SimThreadManager** ()
- void **spawnSimThreads** ()
- void **quitSimThreads** ()
- void **simThreadStartCallback** ()
- void **simThreadExitCallback** ()

Private Attributes

- **SimThread** * **m_sim_threads**
- **CoreThread** * **m_core_threads**
- **Lock** **m_active_threads_lock**
- **UInt32** **m_active_threads**

6.333.1 Detailed Description

Definition at line 7 of file sim_thread_manager.h.

6.333.2 Constructor & Destructor Documentation

6.333.2.1 SimThreadManager()

```
SimThreadManager::SimThreadManager ( )
```

Definition at line 8 of file sim_thread_manager.cc.

6.333.2.2 ∼SimThreadManager()

```
SimThreadManager::∼SimThreadManager ( )
```

Definition at line 13 of file sim_thread_manager.cc.

References LOG_ASSERT_WARNING, and m_active_threads.

6.333.3 Member Function Documentation

6.333.3.1 quitSimThreads()

```
void SimThreadManager::quitSimThreads ( )
```

Definition at line 51 of file sim_thread_manager.cc.

References Transport::barrier(), NetPacket::bufferSize(), CORE_THREAD_TERMINATE_THREADS, Transport↩ ::getGlobalNode(), Transport::getSingleton(), Config::getSingleton(), Config::getTotalCores(), LOG_PRINT, m_↩ active_threads, m_core_threads, m_sim_threads, NetPacket::receiver, Transport::Node::send(), SIM_THREAD↩ _TERMINATE_THREADS, and SubsecondTime::Zero().

Referenced by Simulator::∼Simulator().

6.333.3.2 simThreadExitCallback()

```
void SimThreadManager::simThreadExitCallback ( )
```

Definition at line 97 of file sim_thread_manager.cc.

References TLock< T_LockCreator >::acquire(), m_active_threads, m_active_threads_lock, and TLock< T_↩ LockCreator >::release().

6.333.3.3 simThreadStartCallback()

```
void SimThreadManager::simThreadStartCallback ( )
```

Definition at line 90 of file sim_thread_manager.cc.

References TLock< T_LockCreator >::acquire(), m_active_threads, m_active_threads_lock, and TLock< T_↩ LockCreator >::release().

6.333.3.4 spawnSimThreads()

```
void SimThreadManager::spawnSimThreads ( )
```

Definition at line 19 of file sim_thread_manager.cc.

References __attribute__, Config::getSingleton(), Config::getTotalCores(), LOG_PRINT, m_active_threads, m_↩ core_threads, m_sim_threads, SimThread::spawn(), and CoreThread::spawn().

Referenced by Simulator::start().

6.333.4 Member Data Documentation

6.333.4.1 m_active_threads

```
UInt32 SimThreadManager::m_active_threads  [private]
```

Definition at line 24 of file sim_thread_manager.h.

Referenced by quitSimThreads(), simThreadExitCallback(), simThreadStartCallback(), spawnSimThreads(), and ∼SimThreadManager().

6.333.4.2 m_active_threads_lock

```
Lock SimThreadManager::m_active_threads_lock  [private]
```

Definition at line 23 of file sim_thread_manager.h.

Referenced by simThreadExitCallback(), and simThreadStartCallback().

6.333.4.3 m_core_threads

```
CoreThread* SimThreadManager::m_core_threads  [private]
```

Definition at line 21 of file sim_thread_manager.h.

Referenced by quitSimThreads(), and spawnSimThreads().

6.333.4.4 m_sim_threads

```
SimThread* SimThreadManager::m_sim_threads  [private]
```

Definition at line 20 of file sim_thread_manager.h.

Referenced by quitSimThreads(), and spawnSimThreads().

The documentation for this class was generated from the following files:

- common/system/ **sim_thread_manager.h**
- common/system/ **sim_thread_manager.cc**

6.334 Simulator Class Reference

```
#include <simulator.h>
```

Public Member Functions

- **Simulator** ()
- ~**Simulator** ()
- void **start** ()
- **SyscallServer** * **getSyscallServer** ()
- **SyncServer** * **getSyncServer** ()
- **MagicServer** * **getMagicServer** ()
- **ClockSkewMinimizationServer** * **getClockSkewMinimizationServer** ()
- **CoreManager** * **getCoreManager** ()
- **SimThreadManager** * **getSimThreadManager** ()
- **ThreadManager** * **getThreadManager** ()
- **ClockSkewMinimizationManager** * **getClockSkewMinimizationManager** ()
- **FastForwardPerformanceManager** * **getFastForwardPerformanceManager** ()
- **Config** * **getConfig** ()
- **config::Config** * **getCfg** ()
- void **hideCfg** ()
- **StatsManager** * **getStatsManager** ()
- **ThreadStatsManager** * **getThreadStatsManager** ()
- **DvfsManager** * **getDvfsManager** ()
- **HooksManager** * **getHooksManager** ()
- **SamplingManager** * **getSamplingManager** ()
- **FaultinjectionManager** * **getFaultinjectionManager** ()
- **TraceManager** * **getTraceManager** ()
- **TagsManager** * **getTagsManager** ()
- **RoutineTracer** * **getRoutineTracer** ()
- **MemoryTracker** * **getMemoryTracker** ()
- void **setMemoryTracker** (**MemoryTracker** *memory_tracker)
- bool **isRunning** ()
- void **setInstrumentationMode** (**InstMode::inst_mode_t** new_mode, bool update_barrier)
- **InstMode::inst_mode_t** **getInstrumentationMode** ()
- void **createDecoder** ()
- dl::Decoder * **getDecoder** ()

Static Public Member Functions

- static **Simulator** * **getSingleton** ()
- static void **setConfig** (**config::Config** * **cfg**, **Config::SimulationMode** mode)
- static void **allocate** ()
- static void **release** ()
- static void **enablePerformanceModels** ()
- static void **disablePerformanceModels** ()

Private Member Functions

- void **printInstModeSummary** ()

Private Attributes

- **Config m_config**
- **Log m_log**
- **TagsManager** * **m_tags_manager**
- **SyscallServer** * **m_syscall_server**
- **SyncServer** * **m_sync_server**
- **MagicServer** * **m_magic_server**
- **ClockSkewMinimizationServer** * **m_clock_skew_minimization_server**
- **StatsManager** * **m_stats_manager**
- **Transport** * **m_transport**
- **CoreManager** * **m_core_manager**
- **ThreadManager** * **m_thread_manager**
- **ThreadStatsManager** * **m_thread_stats_manager**
- **SimThreadManager** * **m_sim_thread_manager**
- **ClockSkewMinimizationManager** * **m_clock_skew_minimization_manager**
- **FastForwardPerformanceManager** * **m_fastforward_performance_manager**
- **TraceManager** * **m_trace_manager**
- **DvfsManager** * **m_dvfs_manager**
- **HooksManager** * **m_hooks_manager**
- **SamplingManager** * **m_sampling_manager**
- **FaultinjectionManager** * **m_faultinjection_manager**
- **RoutineTracer** * **m_rtn_tracer**
- **MemoryTracker** * **m_memory_tracker**
- bool **m_running**
- bool **m_inst_mode_output**
- dl::DecoderFactory * **m_factory**

Static Private Attributes

- static **Simulator** * **m_singleton**
- static **config::Config** * **m_config_file**
- static bool **m_config_file_allowed** = true
- static **Config::SimulationMode m_mode**
- static dl::Decoder * **m_decoder**

6.334.1 Detailed Description

Definition at line 34 of file simulator.h.

6.334.2 Constructor & Destructor Documentation

6.334.2.1 Simulator()

```
Simulator::Simulator ( )
```

Definition at line 106 of file simulator.cc.

Referenced by allocate().

6.334.2.2 ∼Simulator()

```
Simulator::∼Simulator ( )
```

Definition at line 204 of file simulator.cc.

References Transport::barrier(), HooksManager::callHooks(), HooksManager::fini(), getMagicServer(), HookType::HOOK_SIM_END, LOG_PRINT, m_clock_skew_minimization_manager, m_clock_skew_minimization_server, m_config_file_allowed, m_core_manager, m_dvfs_manager, m_faultinjection_manager, m_hooks_manager, m_magic_server, m_rtn_tracer, m_sampling_manager, m_sim_thread_manager, m_stats_manager, m_sync_server, m_syscall_server, m_tags_manager, m_thread_stats_manager, m_transport, SimThreadManager::quitSimThreads(), StatsManager::recordStats(), TotalTimer::reports(), and MagicServer::setPerformance().

6.334.3 Member Function Documentation

6.334.3.1 allocate()

```
void Simulator::allocate ( )  [static]
```

Definition at line 39 of file simulator.cc.

References m_singleton, and Simulator().

Referenced by main().

6.334.3.2 createDecoder()

```
void Simulator::createDecoder ( )
```

Definition at line 51 of file simulator.cc.

References LOG_PRINT_ERROR, m_decoder, and m_factory.

Referenced by start().

6.334.3.3 disablePerformanceModels()

```
void Simulator::disablePerformanceModels ( )  [static]
```

Definition at line 272 of file simulator.cc.

References InstMode::DETAILED, and InstMode::inst_mode_roi.

Referenced by MagicServer::disablePerformance().

6.334.3.4 enablePerformanceModels()

```
void Simulator::enablePerformanceModels ( )  [static]
```

Definition at line 264 of file simulator.cc.

References InstMode::DETAILED, getFastForwardPerformanceManager(), and InstMode::inst_mode_roi.

Referenced by MagicServer::enablePerformance().

6.334.3.5 getCfg()

```
config::Config* Simulator::getCfg ( )  [inline]
```

Definition at line 57 of file simulator.h.

References m_config_file.

Referenced by start().

6.334.3.6 getClockSkewMinimizationManager()

```
ClockSkewMinimizationManager* Simulator::getClockSkewMinimizationManager ( )  [inline]
```

Definition at line 54 of file simulator.h.

References m_clock_skew_minimization_manager.

6.334.3.7 getClockSkewMinimizationServer()

```
ClockSkewMinimizationServer* Simulator::getClockSkewMinimizationServer ( )  [inline]
```

Definition at line 50 of file simulator.h.

References m_clock_skew_minimization_server.

Referenced by MagicServer::disablePerformance(), and setInstrumentationMode().

6.334.3.8 getConfig()

```
Config* Simulator::getConfig ( )  [inline]
```

Definition at line 56 of file simulator.h.

References m_config.

Referenced by MagicServer::disablePerformance(), MagicServer::Magic_unlocked(), printInstModeSummary(), setInstrumentationMode(), and start().

6.334.3.9 getCoreManager()

```
CoreManager* Simulator::getCoreManager ( )  [inline]
```

Definition at line 51 of file simulator.h.

References m_core_manager.

Referenced by MagicServer::disablePerformance().

6.334.3.10 getDecoder()

```
dl::Decoder * Simulator::getDecoder ( )
```

Definition at line 93 of file simulator.cc.

References m_decoder.

6.334.3.11 getDvfsManager()

```
DvfsManager* Simulator::getDvfsManager ( )  [inline]
```

Definition at line 65 of file simulator.h.

References m_dvfs_manager.

Referenced by MagicServer::setFrequency().

6.334.3.12 getFastForwardPerformanceManager()

```
FastForwardPerformanceManager* Simulator::getFastForwardPerformanceManager ( )  [inline]
```

Definition at line 55 of file simulator.h.

References m_fastforward_performance_manager.

Referenced by enablePerformanceModels(), setInstrumentationMode(), and start().

6.334.3.13 getFaultinjectionManager()

```
FaultinjectionManager* Simulator::getFaultinjectionManager ( )  [inline]
```

Definition at line 68 of file simulator.h.

References m_faultinjection_manager.

6.334.3.14 getHooksManager()

```
HooksManager* Simulator::getHooksManager ( )  [inline]
```

Definition at line 66 of file simulator.h.

References m_hooks_manager.

6.334.3.15 getInstrumentationMode()

```
InstMode::inst_mode_t Simulator::getInstrumentationMode ( )  [inline]
```

Definition at line 80 of file simulator.h.

References InstMode::inst_mode.

6.334.3.16 getMagicServer()

```
MagicServer* Simulator::getMagicServer ( )  [inline]
```

Definition at line 49 of file simulator.h.

References m_magic_server.

Referenced by ~Simulator().

6.334.3.17 getMemoryTracker()

```
MemoryTracker* Simulator::getMemoryTracker ( )  [inline]
```

Definition at line 72 of file simulator.h.

References m_memory_tracker.

6.334.3.18 getRoutineTracer()

```
RoutineTracer* Simulator::getRoutineTracer ( )  [inline]
```

Definition at line 71 of file simulator.h.

References m_rtn_tracer.

6.334.3.19 getSamplingManager()

```
SamplingManager* Simulator::getSamplingManager ( )  [inline]
```

Definition at line 67 of file simulator.h.

References m_sampling_manager.

6.334.3.20 getSimThreadManager()

```
SimThreadManager* Simulator::getSimThreadManager ( )  [inline]
```

Definition at line 52 of file simulator.h.

References m_sim_thread_manager.

6.334.3.21 getSingleton()

```
static  Simulator* Simulator::getSingleton ( )  [inline], [static]
```

Definition at line 42 of file simulator.h.

References m_singleton.

Referenced by __attribute__().

6.334.3.22 getStatsManager()

```
StatsManager* Simulator::getStatsManager ( )  [inline]
```

Definition at line 63 of file simulator.h.

References m_stats_manager.

6.334.3.23 getSyncServer()

```
SyncServer* Simulator::getSyncServer ( )  [inline]
```

Definition at line 48 of file simulator.h.

References m_sync_server.

6.334.3.24 getSyscallServer()

```
SyscallServer* Simulator::getSyscallServer ( )  [inline]
```

Definition at line 47 of file simulator.h.

References m_syscall_server.

6.334.3.25 getTagsManager()

```
TagsManager* Simulator::getTagsManager ( )  [inline]
```

Definition at line 70 of file simulator.h.

References m_tags_manager.

6.334.3.26 getThreadManager()

```
ThreadManager* Simulator::getThreadManager ( )  [inline]
```

Definition at line 53 of file simulator.h.

References m_thread_manager.

Referenced by MagicServer::Magic().

6.334.3.27 getThreadStatsManager()

```
ThreadStatsManager* Simulator::getThreadStatsManager ( )  [inline]
```

Definition at line 64 of file simulator.h.

References m_thread_stats_manager.

6.334.3.28 getTraceManager()

```
TraceManager* Simulator::getTraceManager ( )  [inline]
```

Definition at line 69 of file simulator.h.

References m_trace_manager.

6.334.3.29 hideCfg()

```
void Simulator::hideCfg ( )  [inline]
```

Definition at line 62 of file simulator.h.

References m_config_file_allowed.

6.334.3.30 isRunning()

```
bool Simulator::isRunning ( )  [inline]
```

Definition at line 75 of file simulator.h.

References m_running.

6.334.3.31 printInstModeSummary()

```
void Simulator::printInstModeSummary ( )  [private]
```

Definition at line 303 of file simulator.cc.

References getConfig(), InstMode::inst_mode_end, InstMode::inst_mode_init, inst_mode_names, InstMode::inst↩
_mode_roi, LOG_PRINT_ERROR, Config::PINTOOL, Config::ROI_FULL, Config::ROI_MAGIC, Config::ROI_SC↩
RIPT, and Config::STANDALONE.

Referenced by start().

6.334.3.32 release()

```
void Simulator::release ( )  [static]
```

Definition at line 98 of file simulator.cc.

References m_running, and m_singleton.

Referenced by ApplicationExit(), and simulatorAbort().

6.334.3.33 setConfig()

```
void Simulator::setConfig (
            config::Config * cfg,
            Config::SimulationMode mode )  [static]
```

Definition at line 45 of file simulator.cc.

References cfg, m_config_file, and m_mode.

Referenced by main().

6.334.3.34 setInstrumentationMode()

```
void Simulator::setInstrumentationMode (
              InstMode::inst_mode_t new_mode,
             bool update_barrier )
```

Definition at line 280 of file simulator.cc.

References InstMode::DETAILED, getClockSkewMinimizationServer(), getConfig(), getFastForwardPerformanceManager(), HookType::HOOK_INSTRUMENT_MODE, InstMode::inst_mode, inst_mode_names, InstMode::INVALID, m_inst_mode_output, Config::PINTOOL, ClockSkewMinimizationServer::setDisable(), and InstMode::updateInstrumentationMode().

Referenced by start().

6.334.3.35 setMemoryTracker()

```
void Simulator::setMemoryTracker (
              MemoryTracker * memory_tracker )  [inline]
```

Definition at line 73 of file simulator.h.

References m_memory_tracker.

6.334.3.36 start()

```
void Simulator::start ( )
```

Definition at line 130 of file simulator.cc.

References HooksManager::callHooks(), FastForwardPerformanceManager::create(), FaultinjectionManager::create(), Transport::create(), ClockSkewMinimizationManager::create(), ClockSkewMinimizationServer::create(), RoutineTracer::create(), createDecoder(), CircularLog::enableCallbacks(), Config::formatOutputFileName(), InstMode::fromString(), config::Config::getBool(), getCfg(), getConfig(), getFastForwardPerformanceManager(), HookType::HOOK_SIM_START, Fxsupport::init(), InstructionTracer::init(), TraceManager::init(), PthreadEmu::init(), HooksManager::init(), Instruction::initializeStaticInstructionModel(), InstMode::inst_mode_end, InstMode::inst_mode_init, InstMode::inst_mode_roi, LOG_PRINT, m_clock_skew_minimization_manager, m_clock_skew_minimization_server, m_config, m_config_file, m_core_manager, m_dvfs_manager, m_fastforward_performance_manager, m_faultinjection_manager, m_hooks_manager, m_inst_mode_output, m_magic_server, m_rtn_tracer, m_running, m_sampling_manager, m_sim_thread_manager, m_stats_manager, m_sync_server, m_syscall_server, m_thread_manager, m_thread_stats_manager, m_trace_manager, m_transport, printInstModeSummary(), StatsManager::recordStats(), Config::ROI_FULL, config::Config::saveAs(), setInstrumentationMode(), and SimThreadManager::spawnSimThreads().

6.334.4 Member Data Documentation

6.334.4.1 m_clock_skew_minimization_manager

```
ClockSkewMinimizationManager* Simulator::m_clock_skew_minimization_manager  [private]
```

Definition at line 100 of file simulator.h.

Referenced by getClockSkewMinimizationManager(), start(), and ~Simulator().

6.334.4.2 m_clock_skew_minimization_server

```
ClockSkewMinimizationServer* Simulator::m_clock_skew_minimization_server  [private]
```

Definition at line 93 of file simulator.h.

Referenced by getClockSkewMinimizationServer(), start(), and ~Simulator().

6.334.4.3 m_config

```
Config Simulator::m_config  [private]
```

Definition at line 87 of file simulator.h.

Referenced by getConfig(), and start().

6.334.4.4 m_config_file

```
config::Config * Simulator::m_config_file  [static], [private]
```

Definition at line 115 of file simulator.h.

Referenced by getCfg(), setConfig(), and start().

6.334.4.5 m_config_file_allowed

```
bool Simulator::m_config_file_allowed = true  [static], [private]
```

Definition at line 116 of file simulator.h.

Referenced by hideCfg(), and ~Simulator().

6.334.4.6 m_core_manager

```
CoreManager* Simulator::m_core_manager  [private]
```

Definition at line 96 of file simulator.h.

Referenced by getCoreManager(), start(), and ~Simulator().

6.334.4.7 m_decoder

```
dl::Decoder * Simulator::m_decoder  [static], [private]
```

Definition at line 120 of file simulator.h.

Referenced by createDecoder(), and getDecoder().

6.334.4.8 m_dvfs_manager

```
DvfsManager* Simulator::m_dvfs_manager  [private]
```

Definition at line 103 of file simulator.h.

Referenced by getDvfsManager(), start(), and ~Simulator().

6.334.4.9 m_factory

```
dl::DecoderFactory* Simulator::m_factory  [private]
```

Definition at line 122 of file simulator.h.

Referenced by createDecoder().

6.334.4.10 m_fastforward_performance_manager

```
FastForwardPerformanceManager* Simulator::m_fastforward_performance_manager  [private]
```

Definition at line 101 of file simulator.h.

Referenced by getFastForwardPerformanceManager(), and start().

6.334.4.11 m_faultinjection_manager

```
FaultinjectionManager* Simulator::m_faultinjection_manager  [private]
```

Definition at line 106 of file simulator.h.

Referenced by getFaultinjectionManager(), start(), and ~Simulator().

6.334.4.12 m_hooks_manager

```
HooksManager* Simulator::m_hooks_manager  [private]
```

Definition at line 104 of file simulator.h.

Referenced by getHooksManager(), start(), and ~Simulator().

6.334.4.13 m_inst_mode_output

```
bool Simulator::m_inst_mode_output  [private]
```

Definition at line 111 of file simulator.h.

Referenced by setInstrumentationMode(), and start().

6.334.4.14 m_log

```
Log Simulator::m_log  [private]
```

Definition at line 88 of file simulator.h.

6.334.4.15 m_magic_server

```
MagicServer* Simulator::m_magic_server  [private]
```

Definition at line 92 of file simulator.h.

Referenced by getMagicServer(), start(), and ~Simulator().

6.334.4.16 m_memory_tracker

```
MemoryTracker* Simulator::m_memory_tracker  [private]
```

Definition at line 108 of file simulator.h.

Referenced by getMemoryTracker(), and setMemoryTracker().

6.334.4.17 m_mode

```
Config::SimulationMode Simulator::m_mode  [static], [private]
```

Definition at line 117 of file simulator.h.

Referenced by setConfig().

6.334.4.18 m_rtn_tracer

```
RoutineTracer* Simulator::m_rtn_tracer  [private]
```

Definition at line 107 of file simulator.h.

Referenced by getRoutineTracer(), start(), and ~Simulator().

6.334.4.19 m_running

```
bool Simulator::m_running  [private]
```

Definition at line 110 of file simulator.h.

Referenced by isRunning(), release(), and start().

6.334.4.20 m_sampling_manager

```
SamplingManager* Simulator::m_sampling_manager  [private]
```

Definition at line 105 of file simulator.h.

Referenced by getSamplingManager(), start(), and ~Simulator().

6.334.4.21 m_sim_thread_manager

```
SimThreadManager* Simulator::m_sim_thread_manager  [private]
```

Definition at line 99 of file simulator.h.

Referenced by getSimThreadManager(), start(), and ~Simulator().

6.334.4.22 m_singleton

```
Simulator * Simulator::m_singleton  [static], [private]
```

Definition at line 113 of file simulator.h.

Referenced by allocate(), getSingleton(), and release().

6.334.4.23 m_stats_manager

```
StatsManager* Simulator::m_stats_manager  [private]
```

Definition at line 94 of file simulator.h.

Referenced by getStatsManager(), start(), and ~Simulator().

6.334.4.24 m_sync_server

```
SyncServer* Simulator::m_sync_server  [private]
```

Definition at line 91 of file simulator.h.

Referenced by getSyncServer(), start(), and ~Simulator().

6.334.4.25 m_syscall_server

```
SyscallServer* Simulator::m_syscall_server  [private]
```

Definition at line 90 of file simulator.h.

Referenced by getSyscallServer(), start(), and ~Simulator().

6.334.4.26 m_tags_manager

```
TagsManager* Simulator::m_tags_manager  [private]
```

Definition at line 89 of file simulator.h.

Referenced by getTagsManager(), and ~Simulator().

6.334.4.27 m_thread_manager

```
ThreadManager* Simulator::m_thread_manager  [private]
```

Definition at line 97 of file simulator.h.

Referenced by getThreadManager(), and start().

6.334.4.28 m_thread_stats_manager

```
ThreadStatsManager* Simulator::m_thread_stats_manager  [private]
```

Definition at line 98 of file simulator.h.

Referenced by getThreadStatsManager(), start(), and ~Simulator().

6.334.4.29 m_trace_manager

```
TraceManager* Simulator::m_trace_manager  [private]
```

Definition at line 102 of file simulator.h.

Referenced by getTraceManager(), and start().

6.334.4.30 m_transport

```
Transport* Simulator::m_transport  [private]
```

Definition at line 95 of file simulator.h.

Referenced by start(), and ~Simulator().

The documentation for this class was generated from the following files:

- common/system/ **simulator.h**
- common/system/ **simulator.cc**

6.335 SmTransport::SmNode Class Reference

`#include <smtransport.h>`

Inheritance diagram for SmTransport::SmNode:

Public Member Functions

- **SmNode** (**core_id_t** core_id, **SmTransport** ∗smt)
- ∼**SmNode** ()
- void **globalSend** (**SInt32**, const void ∗, **UInt32**)
- void **send** (**core_id_t**, const void ∗, **UInt32**)
- **Byte** ∗ **recv** ()
- bool **query** ()

Private Member Functions

- void **send** (**SmNode** ∗dest, const void ∗buffer, **UInt32** length)

Private Attributes

- std::queue< **Byte** ∗ > **m_queue**
- **Lock m_lock**
- **ConditionVariable m_cond**
- **SmTransport** ∗ **m_smt**

Additional Inherited Members

6.335.1 Detailed Description

Definition at line 15 of file smtransport.h.

6.335.2 Constructor & Destructor Documentation

6.335.2.1 SmNode()

```
SmTransport::SmNode::SmNode (
              core_id_t core_id,
              SmTransport * smt )
```

Definition at line 69 of file smtransport.cc.

6.335.2.2 ~SmNode()

```
SmTransport::SmNode::~SmNode ( )
```

Definition at line 75 of file smtransport.cc.

References LOG_ASSERT_WARNING.

6.335.3 Member Function Documentation

6.335.3.1 globalSend()

```
void SmTransport::SmNode::globalSend (
              SInt32 dest_proc,
             const void * buffer,
              UInt32 length )  [virtual]
```

Implements **Transport::Node** (p. 872).

Definition at line 81 of file smtransport.cc.

References LOG_ASSERT_ERROR.

6.335.3.2 query()

```
bool SmTransport::SmNode::query ( )  [virtual]
```

Implements **Transport::Node** (p. 872).

Definition at line 132 of file smtransport.cc.

6.335.3.3 recv()

```
Byte * SmTransport::SmNode::recv ( )  [virtual]
```

Implements **Transport::Node** (p. 872).

Definition at line 107 of file smtransport.cc.

References LOG_PRINT.

6.335.3.4 send() [1/2]

```
void SmTransport::SmNode::send (
              core_id_t dest_id,
             const void * buffer,
              UInt32 length )  [virtual]
```

Implements **Transport::Node** (p. 873).

Definition at line 87 of file smtransport.cc.

References LOG_ASSERT_ERROR.

6.335.3.5 send() [2/2]

```
void SmTransport::SmNode::send (
              SmNode * dest,
             const void * buffer,
              UInt32 length )  [private]
```

Definition at line 94 of file smtransport.cc.

References TLock< T_LockCreator >::acquire(), ConditionVariable::broadcast(), LOG_PRINT, m_cond, m_lock, m_queue, and TLock< T_LockCreator >::release().

6.335.4 Member Data Documentation

6.335.4.1 m_cond

```
ConditionVariable SmTransport::SmNode::m_cond  [private]
```

Definition at line 31 of file smtransport.h.

Referenced by send().

6.335.4.2 m_lock

```
Lock SmTransport::SmNode::m_lock  [private]
```

Definition at line 30 of file smtransport.h.

Referenced by send().

6.335.4.3 m_queue

```
std::queue< Byte*> SmTransport::SmNode::m_queue  [private]
```

Definition at line 29 of file smtransport.h.

Referenced by send().

6.335.4.4 m_smt

```
SmTransport* SmTransport::SmNode::m_smt  [private]
```

Definition at line 32 of file smtransport.h.

The documentation for this class was generated from the following files:

- common/transport/ **smtransport.h**
- common/transport/ **smtransport.cc**

6.336 SmTransport Class Reference

```
#include <smtransport.h>
```

Inheritance diagram for SmTransport:

Classes

- class **SmNode**

Public Member Functions

- **SmTransport** ()
- ~**SmTransport** ()
- **Node** * **createNode** (**core_id_t** core_id)
- void **barrier** ()
- **Node** * **getGlobalNode** ()

Private Member Functions

- **SmNode** * **getNodeFromId** (**core_id_t** core_id)
- void **clearNodeForId** (**core_id_t** core_id)

Private Attributes

- **Node** * **m_global_node**
- **SmNode** ** **m_core_nodes**

Additional Inherited Members

6.336.1 Detailed Description

Definition at line 9 of file smtransport.h.

6.336.2 Constructor & Destructor Documentation

6.336.2.1 SmTransport()

```
SmTransport::SmTransport ( )
```

Definition at line 9 of file smtransport.cc.

References Config::getSingleton(), Config::getTotalCores(), m_core_nodes, and m_global_node.

6.336.2.2 ~SmTransport()

```
SmTransport::~SmTransport ( )
```

Definition at line 17 of file smtransport.cc.

References m_core_nodes, and m_global_node.

6.336.3 Member Function Documentation

6.336.3.1 barrier()

```
void SmTransport::barrier ( )  [virtual]
```

Implements **Transport** (p. 1464).

Definition at line 42 of file smtransport.cc.

6.336.3.2 clearNodeForId()

```
void SmTransport::clearNodeForId (
            core_id_t core_id )  [private]
```

Definition at line 59 of file smtransport.cc.

References Config::getSingleton(), and m_core_nodes.

6.336.3.3 createNode()

```
Transport::Node * SmTransport::createNode (
            core_id_t core_id )  [virtual]
```

Implements **Transport** (p. 1465).

Definition at line 28 of file smtransport.cc.

References Config::getSingleton(), LOG_ASSERT_ERROR, LOG_PRINT, and m_core_nodes.

6.336.3.4 getGlobalNode()

```
Transport::Node * SmTransport::getGlobalNode ( )  [virtual]
```

Implements **Transport** (p. 1465).

Definition at line 47 of file smtransport.cc.

References m_global_node.

6.336.3.5 getNodeFromId()

```
SmTransport::SmNode * SmTransport::getNodeFromId (
            core_id_t core_id )  [private]
```

Definition at line 52 of file smtransport.cc.

References Config::getSingleton(), LOG_ASSERT_ERROR, and m_core_nodes.

6.336.4 Member Data Documentation

6.336.4.1 m_core_nodes

```
SmNode** SmTransport::m_core_nodes  [private]
```

Definition at line 42 of file smtransport.h.

Referenced by clearNodeForId(), createNode(), getNodeFromId(), SmTransport(), and ~SmTransport().

6.336.4.2 m_global_node

```
Node* SmTransport::m_global_node  [private]
```

Definition at line 41 of file smtransport.h.

Referenced by getGlobalNode(), SmTransport(), and ~SmTransport().

The documentation for this class was generated from the following files:

- common/transport/ **smtransport.h**
- common/transport/ **smtransport.cc**

6.337 SmtTimer::SmtThread Class Reference

```
#include <smt_timer.h>
```

Public Member Functions

- **SmtThread** (**Core** * **core**, **PerformanceModel** * **perf**)
- ~**SmtThread** ()

Public Attributes

- **Core** ∗const **core**
- const **PerformanceModel** ∗const **perf**
- const **Thread** ∗ **thread**
- bool **in_wakeup**
- bool **running**
- bool **in_barrier**
- **ConditionVariable** **cond**

6.337.1 Detailed Description

Definition at line 19 of file smt_timer.h.

6.337.2 Constructor & Destructor Documentation

6.337.2.1 SmtThread()

```
SmtTimer::SmtThread::SmtThread (
            Core * core,
            PerformanceModel * perf )
```

Definition at line 10 of file smt_timer.cc.

6.337.2.2 ∼SmtThread()

```
SmtTimer::SmtThread::~SmtThread ( )
```

Definition at line 20 of file smt_timer.cc.

6.337.3 Member Data Documentation

6.337.3.1 cond

```
ConditionVariable SmtTimer::SmtThread::cond
```

Definition at line 28 of file smt_timer.h.

Referenced by SmtTimer::barrier(), SmtTimer::barrierRelease(), SmtTimer::disable(), and SmtTimer::threadMigrate().

6.337.3.2 core

```
Core* const SmtTimer::SmtThread::core
```

Definition at line 21 of file smt_timer.h.

Referenced by SmtTimer::barrier(), RobSmtTimer::issueInstruction(), and SmtTimer::simulate().

6.337.3.3 in_barrier

```
bool SmtTimer::SmtThread::in_barrier
```

Definition at line 27 of file smt_timer.h.

Referenced by SmtTimer::barrier(), SmtTimer::barrierRelease(), SmtTimer::disable(), SmtTimer::getStateStr(), SmtTimer::isBarrierReached(), RobSmtTimer::printRob(), and SmtTimer::threadMigrate().

6.337.3.4 in_wakeup

```
bool SmtTimer::SmtThread::in_wakeup
```

Definition at line 25 of file smt_timer.h.

Referenced by RobSmtTimer::printRob().

6.337.3.5 perf

```
const PerformanceModel* const SmtTimer::SmtThread::perf
```

Definition at line 22 of file smt_timer.h.

6.337.3.6 running

```
bool SmtTimer::SmtThread::running
```

Definition at line 26 of file smt_timer.h.

Referenced by SmtTimer::barrierRelease(), RobSmtTimer::canExecute(), RobSmtTimer::doDispatch(), SmtTimer↩
::getStateStr(), SmtTimer::isBarrierReached(), RobSmtTimer::printRob(), and SmtTimer::threadMigrate().

6.337.3.7 thread

```
const Thread* SmtTimer::SmtThread::thread
```

Definition at line 23 of file smt_timer.h.

Referenced by SmtTimer::barrier(), and SmtTimer::threadMigrate().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/rob_performance_model/ **smt_timer.h**
- common/performance_model/performance_models/rob_performance_model/ **smt_timer.cc**

6.338 SmtTimer Class Reference

```
#include <smt_timer.h>
```

Inheritance diagram for SmtTimer:

Classes

- class **SmtThread**

Public Member Functions

- **SmtTimer** (uint64_t num_threads)
- virtual ~**SmtTimer** ()
- **UInt8 registerThread** (**Core** *core, **PerformanceModel** *perf)
- void **simulate** (**smtthread_id_t** thread_id)
- virtual void **synchronize** (**smtthread_id_t** thread_id, **SubsecondTime** time)=0
- void **enable** ()
- void **disable** ()

Public Attributes

- **Lock m_lock**

Protected Types

- typedef uint8_t **smtthread_id_t**

Protected Member Functions

- virtual void **initializeThread** (**smtthread_id_t** thread_num)=0
- virtual uint64_t **threadNumSurplusInstructions** (**smtthread_id_t** thread_num)=0
- virtual bool **threadHasEnoughInstructions** (**smtthread_id_t** thread_num)=0
- virtual void **notifyNumActiveThreadsChange** ()
- virtual void **pushInstructions** (**smtthread_id_t** thread_id, const std::vector< **DynamicMicroOp** * > &insts)=0
- virtual boost::tuple< uint64_t, **SubsecondTime** > **returnLatency** (**smtthread_id_t** thread_id)=0
- virtual void **execute** ()=0
- char **getStateStr** (**smtthread_id_t** thread_num)

Protected Attributes

- const uint64_t **m_num_threads**
- std::vector< **SmtThread** * > **m_threads**
- bool **in_sync**
- bool **enabled**
- **smtthread_id_t** **execute_thread**

Private Member Functions

- void **roiBegin** ()
- void **threadStart** (**HooksManager::ThreadTime** *argument)
- void **threadExit** (**HooksManager::ThreadTime** *argument)
- void **threadStall** (**HooksManager::ThreadStall** *argument)
- void **threadResume** (**HooksManager::ThreadResume** *argument)
- void **threadMigrate** (**HooksManager::ThreadMigrate** *argument)
- **smtthread_id_t** **findSmtThreadFromThread** (**thread_id_t** thread_id)
- **smtthread_id_t** **findSmtThreadFromCore** (**core_id_t** core_id)
- bool **barrier** (**smtthread_id_t** thread_id)
- bool **barrierRelease** (bool release_all, **smtthread_id_t** thread_id)
- bool **isBarrierReached** ()
- void **signalBarrier** ()

Static Private Member Functions

- static **SInt64** **hookRoiBegin** (**UInt64** object, **UInt64** argument)
- static **SInt64** **hookThreadStart** (**UInt64** object, **UInt64** argument)
- static **SInt64** **hookThreadExit** (**UInt64** object, **UInt64** argument)
- static **SInt64** **hookThreadStall** (**UInt64** object, **UInt64** argument)
- static **SInt64** **hookThreadResume** (**UInt64** object, **UInt64** argument)
- static **SInt64** **hookThreadMigrate** (**UInt64** object, **UInt64** argument)

6.338.1 Detailed Description

Definition at line 16 of file smt_timer.h.

6.338.2 Member Typedef Documentation

6.338.2.1 smtthread_id_t

```
typedef uint8_t  SmtTimer::smtthread_id_t  [protected]
```

Definition at line 34 of file smt_timer.h.

6.338.3 Constructor & Destructor Documentation

6.338.3.1 SmtTimer()

```
SmtTimer::SmtTimer (
            uint64_t num_threads )
```

Definition at line 24 of file smt_timer.cc.

References HookType::HOOK_ROI_BEGIN, HookType::HOOK_THREAD_EXIT, HookType::HOOK_THREAD_M↩IGRATE, HookType::HOOK_THREAD_RESUME, HookType::HOOK_THREAD_STALL, HookType::HOOK_THR↩EAD_START, hookRoiBegin(), hookThreadExit(), hookThreadMigrate(), hookThreadResume(), hookThreadStall(), and hookThreadStart().

6.338.3.2 ∼SmtTimer()

```
SmtTimer::∼SmtTimer ( )  [virtual]
```

Definition at line 38 of file smt_timer.cc.

References m_threads.

6.338.4 Member Function Documentation

6.338.4.1 barrier()

```
bool SmtTimer::barrier (
             smtthread_id_t thread_id )  [private]
```

Definition at line 190 of file smt_timer.cc.

References barrierRelease(), CLOG, SmtTimer::SmtThread::cond, SmtTimer::SmtThread::core, enabled, execute_thread, Thread::getId(), Core::getId(), SmtTimer::SmtThread::in_barrier, isBarrierReached(), LOG_A↩SSERT_ERROR, m_lock, m_threads, SmtTimer::SmtThread::thread, and ConditionVariable::wait().

Referenced by simulate().

6.338.4.2 barrierRelease()

```
bool SmtTimer::barrierRelease (
            bool release_all,
             smtthread_id_t thread_id )  [private]
```

Definition at line 101 of file smt_timer.cc.

References SmtTimer::SmtThread::cond, execute_thread, SmtTimer::SmtThread::in_barrier, m_threads, SmtTimer::SmtThread::running, ConditionVariable::signal(), threadHasEnoughInstructions(), and threadNumSurplusInstructions().

Referenced by barrier(), signalBarrier(), and simulate().

6.338.4.3 disable()

```
void SmtTimer::disable ( )
```

Definition at line 379 of file smt_timer.cc.

References SmtTimer::SmtThread::cond, enabled, SmtTimer::SmtThread::in_barrier, m_threads, and ConditionVariable::signal().

Referenced by RobSmtPerformanceModel::disableDetailedModel().

6.338.4.4 enable()

```
void SmtTimer::enable ( )
```

Definition at line 374 of file smt_timer.cc.

References enabled.

Referenced by RobSmtPerformanceModel::enableDetailedModel().

6.338.4.5 execute()

```
virtual void SmtTimer::execute ( )  [protected], [pure virtual]
```

Implemented in **RobSmtTimer** (p. 1021).

Referenced by simulate().

6.338.4.6 findSmtThreadFromCore()

```
SmtTimer::smtthread_id_t SmtTimer::findSmtThreadFromCore (
            core_id_t core_id )  [private]
```

Definition at line 66 of file smt_timer.cc.

References INVALID_CORE_ID, INVALID_SMTTHREAD_ID, and m_threads.

Referenced by threadMigrate().

6.338.4.7 findSmtThreadFromThread()

```
SmtTimer::smtthread_id_t SmtTimer::findSmtThreadFromThread (
            thread_id_t thread_id )  [private]
```

Definition at line 56 of file smt_timer.cc.

References INVALID_SMTTHREAD_ID, INVALID_THREAD_ID, and m_threads.

Referenced by threadExit(), threadMigrate(), threadResume(), threadStall(), and threadStart().

6.338.4.8 getStateStr()

```
char SmtTimer::getStateStr (
            smtthread_id_t thread_num )  [protected]
```

Definition at line 395 of file smt_timer.cc.

References SmtTimer::SmtThread::in_barrier, m_threads, and SmtTimer::SmtThread::running.

6.338.4.9 hookRoiBegin()

```
static  SInt64 SmtTimer::hookRoiBegin (
            UInt64 object,
            UInt64 argument )  [inline], [static], [private]
```

Definition at line 56 of file smt_timer.h.

Referenced by SmtTimer().

6.338.4.10 hookThreadExit()

```
static  SInt64 SmtTimer::hookThreadExit (
              UInt64 object,
              UInt64 argument )  [inline], [static], [private]
```

Definition at line 62 of file smt_timer.h.

Referenced by SmtTimer().

6.338.4.11 hookThreadMigrate()

```
static  SInt64 SmtTimer::hookThreadMigrate (
              UInt64 object,
              UInt64 argument )  [inline], [static], [private]
```

Definition at line 71 of file smt_timer.h.

Referenced by SmtTimer().

6.338.4.12 hookThreadResume()

```
static  SInt64 SmtTimer::hookThreadResume (
              UInt64 object,
              UInt64 argument )  [inline], [static], [private]
```

Definition at line 68 of file smt_timer.h.

Referenced by SmtTimer().

6.338.4.13 hookThreadStall()

```
static  SInt64 SmtTimer::hookThreadStall (
              UInt64 object,
              UInt64 argument )  [inline], [static], [private]
```

Definition at line 65 of file smt_timer.h.

Referenced by SmtTimer().

6.338.4.14 hookThreadStart()

```
static SInt64 SmtTimer::hookThreadStart (
              UInt64 object,
              UInt64 argument )  [inline], [static], [private]
```

Definition at line 59 of file smt_timer.h.

Referenced by SmtTimer().

6.338.4.15 initializeThread()

```
virtual void SmtTimer::initializeThread (
              smtthread_id_t thread_num )  [protected], [pure virtual]
```

Implemented in **RobSmtTimer** (p. 1022).

Referenced by registerThread().

6.338.4.16 isBarrierReached()

```
bool SmtTimer::isBarrierReached ( )  [private]
```

Definition at line 76 of file smt_timer.cc.

References SmtTimer::SmtThread::in_barrier, in_sync, m_threads, and SmtTimer::SmtThread::running.

Referenced by barrier(), and signalBarrier().

6.338.4.17 notifyNumActiveThreadsChange()

```
virtual void SmtTimer::notifyNumActiveThreadsChange ( )  [inline], [protected], [virtual]
```

Reimplemented in **RobSmtTimer** (p. 1023).

Definition at line 47 of file smt_timer.h.

Referenced by threadExit(), threadMigrate(), threadResume(), threadStall(), and threadStart().

6.338.4.18 pushInstructions()

```
virtual void SmtTimer::pushInstructions (
              smtthread_id_t thread_id,
             const std::vector<  DynamicMicroOp * > & insts )  [protected], [pure virtual]
```

Implemented in **RobSmtTimer** (p. 1024).

6.338.4.19 registerThread()

```
UInt8 SmtTimer::registerThread (
              Core * core,
              PerformanceModel * perf )
```

Definition at line 44 of file smt_timer.cc.

References initializeThread(), and m_threads.

Referenced by RobSmtPerformanceModel::RobSmtPerformanceModel().

6.338.4.20 returnLatency()

```
virtual boost::tuple<uint64_t, SubsecondTime> SmtTimer::returnLatency (
              smtthread_id_t thread_id )  [protected], [pure virtual]
```

Implemented in **RobSmtTimer** (p. 1024).

6.338.4.21 roiBegin()

```
void SmtTimer::roiBegin ( )  [private]
```

Definition at line 230 of file smt_timer.cc.

References m_threads, and synchronize().

6.338.4.22 signalBarrier()

```
void SmtTimer::signalBarrier ( )  [private]
```

Definition at line 224 of file smt_timer.cc.

References barrierRelease(), INVALID_SMTTHREAD_ID, and isBarrierReached().

Referenced by threadExit(), and threadStall().

6.338.4.23 simulate()

```
void SmtTimer::simulate (
              smtthread_id_t thread_id )
```

Definition at line 345 of file smt_timer.cc.

References TLock< T_LockCreator >::acquire(), barrier(), barrierRelease(), SmtTimer::SmtThread::core, execute(), execute_thread, Core::getClockSkewMinimizationClient(), in_sync, INVALID_THREAD_ID, m_lock, m_threads, TLock< T_LockCreator >::release(), ClockSkewMinimizationClient::synchronize(), and threadNumSurplusInstructions().

Referenced by RobSmtPerformanceModel::synchronize().

6.338.4.24 synchronize()

```
virtual void SmtTimer::synchronize (
              smtthread_id_t thread_id,
              SubsecondTime time )  [pure virtual]
```

Implemented in **RobSmtTimer** (p. 1025).

Referenced by roiBegin().

6.338.4.25 threadExit()

```
void SmtTimer::threadExit (
              HooksManager::ThreadTime * argument )  [private]
```

Definition at line 254 of file smt_timer.cc.

References findSmtThreadFromThread(), INVALID_SMTTHREAD_ID, m_lock, m_threads, notifyNumActiveThreadsChange(), signalBarrier(), and HooksManager::ThreadTime::thread_id.

6.338.4.26 threadHasEnoughInstructions()

```
virtual bool SmtTimer::threadHasEnoughInstructions (
              smtthread_id_t thread_num )  [protected], [pure virtual]
```

Implemented in **RobSmtTimer** (p. 1026).

Referenced by barrierRelease().

6.338.4.27 threadMigrate()

```
void SmtTimer::threadMigrate (
            HooksManager::ThreadMigrate * argument )  [private]
```

Definition at line 309 of file smt_timer.cc.

References SmtTimer::SmtThread::cond, HooksManager::ThreadMigrate::core_id, findSmtThreadFromCore(), findSmtThreadFromThread(), SmtTimer::SmtThread::in_barrier, INVALID_SMTTHREAD_ID, m_lock, m_threads, notifyNumActiveThreadsChange(), SmtTimer::SmtThread::running, ConditionVariable::signal(), SmtTimer::SmtThread::thread, and HooksManager::ThreadMigrate::thread_id.

6.338.4.28 threadNumSurplusInstructions()

```
virtual uint64_t SmtTimer::threadNumSurplusInstructions (
            smtthread_id_t thread_num )  [protected], [pure virtual]
```

Implemented in **RobSmtTimer** (p. 1026).

Referenced by barrierRelease(), and simulate().

6.338.4.29 threadResume()

```
void SmtTimer::threadResume (
            HooksManager::ThreadResume * argument )  [private]
```

Definition at line 286 of file smt_timer.cc.

References findSmtThreadFromThread(), INVALID_SMTTHREAD_ID, m_lock, m_threads, notifyNumActiveThreadsChange(), and HooksManager::ThreadResume::thread_id.

6.338.4.30 threadStall()

```
void SmtTimer::threadStall (
            HooksManager::ThreadStall * argument )  [private]
```

Definition at line 270 of file smt_timer.cc.

References findSmtThreadFromThread(), INVALID_SMTTHREAD_ID, m_lock, m_threads, notifyNumActiveThreadsChange(), signalBarrier(), and HooksManager::ThreadStall::thread_id.

6.338.4.31 threadStart()

```
void SmtTimer::threadStart (
            HooksManager::ThreadTime * argument )  [private]
```

Definition at line 242 of file smt_timer.cc.

References findSmtThreadFromThread(), INVALID_SMTTHREAD_ID, m_lock, m_threads, notifyNumActive↩ ThreadsChange(), and HooksManager::ThreadTime::thread_id.

6.338.5 Member Data Documentation

6.338.5.1 enabled

```
bool SmtTimer::enabled  [protected]
```

Definition at line 41 of file smt_timer.h.

Referenced by barrier(), disable(), and enable().

6.338.5.2 execute_thread

```
smtthread_id_t SmtTimer::execute_thread  [protected]
```

Definition at line 42 of file smt_timer.h.

Referenced by barrier(), barrierRelease(), and simulate().

6.338.5.3 in_sync

```
bool SmtTimer::in_sync  [protected]
```

Definition at line 40 of file smt_timer.h.

Referenced by isBarrierReached(), and simulate().

6.338.5.4 m_lock

```
Lock SmtTimer::m_lock
```

Definition at line 90 of file smt_timer.h.

Referenced by barrier(), RobSmtPerformanceModel::disableDetailedModel(), RobSmtPerformanceModel::enable↩DetailedModel(), RobSmtPerformanceModel::notifyElapsedTimeUpdate(), RobSmtPerformanceModel::simulate(), simulate(), RobSmtPerformanceModel::synchronize(), threadExit(), threadMigrate(), threadResume(), threadStall(), and threadStart().

6.338.5.5 m_num_threads

```
const uint64_t SmtTimer::m_num_threads [protected]
```

Definition at line 37 of file smt_timer.h.

Referenced by RobSmtTimer::computeCurrentWindowsize().

6.338.5.6 m_threads

```
std::vector< SmtThread *> SmtTimer::m_threads [protected]
```

Definition at line 38 of file smt_timer.h.

Referenced by barrier(), barrierRelease(), RobSmtTimer::canExecute(), RobSmtTimer::computeCurrent↩Windowsize(), disable(), RobSmtTimer::doCommit(), RobSmtTimer::doDispatch(), RobSmtTimer::doIssue(), RobSmtTimer::executeCycle(), findSmtThreadFromCore(), findSmtThreadFromThread(), getStateStr(), Rob↩SmtTimer::initializeThread(), isBarrierReached(), RobSmtTimer::issueInstruction(), RobSmtTimer::printRob(), registerThread(), roiBegin(), simulate(), RobSmtTimer::synchronize(), threadExit(), threadMigrate(), thread↩Resume(), threadStall(), threadStart(), and ∼SmtTimer().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/rob_performance_model/ **smt_timer.h**
- common/performance_model/performance_models/rob_performance_model/ **smt_timer.cc**

6.339 SpawnInstruction Class Reference

```
#include <instruction.h>
```

Inheritance diagram for SpawnInstruction:

Public Member Functions

- **SpawnInstruction** (**SubsecondTime** time)
- **SubsecondTime getCost** (**Core** ∗core) const
- **SubsecondTime getTime** () const

Private Attributes

- **SubsecondTime m_time**

Additional Inherited Members

6.339.1 Detailed Description

Definition at line 172 of file instruction.h.

6.339.2 Constructor & Destructor Documentation

6.339.2.1 SpawnInstruction()

```
SpawnInstruction::SpawnInstruction (
              SubsecondTime time )
```

Definition at line 94 of file instruction.cc.

6.339.3 Member Function Documentation

6.339.3.1 getCost()

```
SubsecondTime SpawnInstruction::getCost (
              Core * core ) const  [virtual]
```

Reimplemented from **PseudoInstruction** (p. 948).

Definition at line 99 of file instruction.cc.

References LOG_ASSERT_ERROR, and SubsecondTime::Zero().

6.339.3.2 getTime()

```
SubsecondTime SpawnInstruction::getTime ( ) const
```

Definition at line 105 of file instruction.cc.

References m_time.

Referenced by PerformanceModel::queuePseudoInstruction().

6.339.4 Member Data Documentation

6.339.4.1 m_time

```
SubsecondTime SpawnInstruction::m_time  [private]
```

Definition at line 180 of file instruction.h.

Referenced by getTime().

The documentation for this class was generated from the following files:

- common/performance_model/ **instruction.h**
- common/performance_model/ **instruction.cc**

6.340 SpinLoopDetectionState Struct Reference

`#include <spin_loop_detection.h>`

Public Attributes

- intptr_t **write_addr**
- uint64_t **write_value**
- ADDRINT **reg_value** [REG_MACHINE_LAST]
- **SpinLoopDetector** * **sld**

6.340.1 Detailed Description

Definition at line 18 of file spin_loop_detection.h.

6.340.2 Member Data Documentation

6.340.2.1 reg_value

```
ADDRINT SpinLoopDetectionState::reg_value[REG_MACHINE_LAST]
```

Definition at line 22 of file spin_loop_detection.h.

6.340.2.2 sld

```
SpinLoopDetector* SpinLoopDetectionState::sld
```

Definition at line 23 of file spin_loop_detection.h.

6.340.2.3 write_addr

```
intptr_t SpinLoopDetectionState::write_addr
```

Definition at line 20 of file spin_loop_detection.h.

6.340.2.4 write_value

```
uint64_t SpinLoopDetectionState::write_value
```

Definition at line 21 of file spin_loop_detection.h.

The documentation for this struct was generated from the following file:

- pin/ **spin_loop_detection.h**

6.341 SpinLoopDetector Class Reference

```
#include <spin_loop_detector.h>
```

Classes

- struct **SdtEntry**

Public Member Functions

- **SpinLoopDetector** (**Thread** *thread)
- void **commitBCT** (uint64_t eip)
- void **commitNonSilentStore** ()
- void **commitRegisterWrite** (**reg_t** reg, uint64_t old_value, uint64_t value)
- bool **inCandidateSpin** ()

Private Types

- typedef uint16_t **reg_t**
- typedef std::pair< uint64_t, uint16_t > **RubEntry**
- typedef std::unordered_map< **reg_t**, **RubEntry** > **Rub**
- typedef std::deque< **SdtEntry** > **Sdt**

Private Attributes

- const **Thread** * **m_thread**
- **Rub** **m_rub**
- **Sdt** **m_sdt**
- uint16_t **m_sdt_bitmask**
- uint8_t **m_sdt_nextid**

Static Private Attributes

- static const size_t **SDT_MAX_SIZE** = 16

6.341.1 Detailed Description

Definition at line 10 of file spin_loop_detector.h.

6.341.2 Member Typedef Documentation

6.341.2.1 reg_t

```
typedef uint16_t SpinLoopDetector::reg_t [private]
```

Definition at line 14 of file spin_loop_detector.h.

6.341.2.2 Rub

```
typedef std::unordered_map< reg_t, RubEntry> SpinLoopDetector::Rub [private]
```

Definition at line 16 of file spin_loop_detector.h.

6.341.2.3 RubEntry

```
typedef std::pair<uint64_t, uint16_t>  SpinLoopDetector::RubEntry  [private]
```

Definition at line 15 of file spin_loop_detector.h.

6.341.2.4 Sdt

```
typedef std::deque< SdtEntry>  SpinLoopDetector::Sdt  [private]
```

Definition at line 26 of file spin_loop_detector.h.

6.341.3 Constructor & Destructor Documentation

6.341.3.1 SpinLoopDetector()

```
SpinLoopDetector::SpinLoopDetector (
            Thread * thread )  [inline]
```

Definition at line 35 of file spin_loop_detector.h.

6.341.4 Member Function Documentation

6.341.4.1 commitBCT()

```
void SpinLoopDetector::commitBCT (
            uint64_t eip )
```

Definition at line 10 of file spin_loop_detector.cc.

References Thread::getCore(), PerformanceModel::getElapsedTime(), Core::getInstructionCount(), Core::get↩PerformanceModel(), m_rub, m_sdt, m_sdt_bitmask, m_sdt_nextid, m_thread, SDT_MAX_SIZE, and Core↩::updateSpinCount().

6.341.4.2 commitNonSilentStore()

```
void SpinLoopDetector::commitNonSilentStore ( )
```

Definition at line 77 of file spin_loop_detector.cc.

References m_rub, and m_sdt.

6.341.4.3 commitRegisterWrite()

```
void SpinLoopDetector::commitRegisterWrite (
              reg_t reg,
             uint64_t old_value,
             uint64_t value )
```

Definition at line 84 of file spin_loop_detector.cc.

References m_rub, and m_sdt_bitmask.

6.341.4.4 inCandidateSpin()

```
bool SpinLoopDetector::inCandidateSpin ( )  [inline]
```

Definition at line 41 of file spin_loop_detector.h.

References m_sdt.

6.341.5 Member Data Documentation

6.341.5.1 m_rub

```
Rub SpinLoopDetector::m_rub  [private]
```

Definition at line 29 of file spin_loop_detector.h.

Referenced by commitBCT(), commitNonSilentStore(), and commitRegisterWrite().

6.341.5.2 m_sdt

```
Sdt SpinLoopDetector::m_sdt  [private]
```

Definition at line 30 of file spin_loop_detector.h.

Referenced by commitBCT(), commitNonSilentStore(), and inCandidateSpin().

6.341.5.3 m_sdt_bitmask

```
uint16_t SpinLoopDetector::m_sdt_bitmask  [private]
```

Definition at line 31 of file spin_loop_detector.h.

Referenced by commitBCT(), and commitRegisterWrite().

6.341.5.4 m_sdt_nextid

```
uint8_t SpinLoopDetector::m_sdt_nextid  [private]
```

Definition at line 32 of file spin_loop_detector.h.

Referenced by commitBCT().

6.341.5.5 m_thread

```
const  Thread* SpinLoopDetector::m_thread  [private]
```

Definition at line 28 of file spin_loop_detector.h.

Referenced by commitBCT().

6.341.5.6 SDT_MAX_SIZE

```
const size_t SpinLoopDetector::SDT_MAX_SIZE = 16  [static], [private]
```

Definition at line 13 of file spin_loop_detector.h.

Referenced by commitBCT().

The documentation for this class was generated from the following files:

- common/misc/ **spin_loop_detector.h**
- common/misc/ **spin_loop_detector.cc**

6.342 StableIterator< T > Class Template Reference

```
#include <stable_iterator.h>
```

Public Member Functions

- **StableIterator** (const **StableIterator**< T > &s)
- **StableIterator** (std::vector< T > &vec, unsigned int offset)
- T * **getPtr** ()
- T * **operator->** ()
- T & **operator*** ()
- **StableIterator**< T > **operator=** (const **StableIterator**< T > & **src**)

Private Attributes

- std::vector< T > & **_vec**
- unsigned int **_offset**

6.342.1 Detailed Description

template<**class T**>
class StableIterator< **T** >

Definition at line 7 of file stable_iterator.h.

6.342.2 Constructor & Destructor Documentation

6.342.2.1 StableIterator() [1/2]

```
template<class T >
StableIterator< T >::  StableIterator (
            const  StableIterator< T > & s )  [inline]
```

Definition at line 10 of file stable_iterator.h.

6.342.2.2 StableIterator() [2/2]

```
template<class T >
StableIterator< T >::  StableIterator (
            std::vector< T > & vec,
            unsigned int offset )  [inline]
```

Definition at line 13 of file stable_iterator.h.

6.342.3 Member Function Documentation

6.342.3.1 getPtr()

```
template<class T >
T*  StableIterator< T >::getPtr ( )  [inline]
```

Definition at line 15 of file stable_iterator.h.

References StableIterator< T >::_offset, and StableIterator< T >::_vec.

Referenced by StableIterator< T >::operator*(), and StableIterator< T >::operator->().

6.342.3.2 operator*()

```
template<class T >
T&  StableIterator< T >::operator* ( )  [inline]
```

Definition at line 17 of file stable_iterator.h.

References StableIterator< T >::getPtr().

6.342.3.3 operator->()

```
template<class T >
T*  StableIterator< T >::operator-> ( )  [inline]
```

Definition at line 16 of file stable_iterator.h.

References StableIterator< T >::getPtr().

6.342.3.4 operator=()

```
template<class T >
StableIterator<T>  StableIterator< T >::operator= (
             const  StableIterator< T > & src )  [inline]
```

Definition at line 19 of file stable_iterator.h.

References src.

6.342.4 Member Data Documentation

6.342.4.1 _offset

```
template<class T >
unsigned int  StableIterator< T >::_offset  [private]
```

Definition at line 25 of file stable_iterator.h.

Referenced by StableIterator< T >::getPtr().

6.342.4.2 _vec

```
template<class T >
std::vector<T>&  StableIterator< T >::_vec  [private]
```

Definition at line 24 of file stable_iterator.h.

Referenced by StableIterator< T >::getPtr().

The documentation for this class was generated from the following file:

- common/misc/ **stable_iterator.h**

6.343 stack_frame Struct Reference

Public Attributes

- struct **stack_frame** * **next**
- void * **ret**

6.343.1 Detailed Description

Definition at line 3 of file callstack.cc.

6.343.2 Member Data Documentation

6.343.2.1 next

```
struct  stack_frame* stack_frame::next
```

Definition at line 4 of file callstack.cc.

Referenced by get_call_stack_from().

6.343.2.2 ret

```
void* stack_frame::ret
```

Definition at line 5 of file callstack.cc.

Referenced by get_call_stack_from().

The documentation for this struct was generated from the following file:

- common/misc/ **callstack.cc**

6.344 ThreadStatsManager::StatCallback Struct Reference

Public Member Functions

- **StatCallback** ()
- **StatCallback** (const char ∗name, **ThreadStatCallback** func, **UInt64** user)
- **UInt64 call** (**ThreadStatType** type, **thread_id_t** thread_id, **Core** ∗core)

Public Attributes

- const char ∗ **m_name**
- **ThreadStatCallback m_func**
- **UInt64 m_user**

6.344.1 Detailed Description

Definition at line 62 of file thread_stats_manager.h.

6.344.2 Constructor & Destructor Documentation

6.344.2.1 StatCallback() [1/2]

```
ThreadStatsManager::StatCallback::StatCallback ( )  [inline]
```

Definition at line 67 of file thread_stats_manager.h.

6.344.2.2 StatCallback() [2/2]

```
ThreadStatsManager::StatCallback::StatCallback (
            const char * name,
             ThreadStatCallback func,
             UInt64 user )  [inline]
```

Definition at line 68 of file thread_stats_manager.h.

6.344.3 Member Function Documentation

6.344.3.1 call()

```
UInt64 ThreadStatsManager::StatCallback::call (
             ThreadStatType type,
             thread_id_t thread_id,
             Core * core )  [inline]
```

Definition at line 69 of file thread_stats_manager.h.

References m_func, and m_user.

6.344.4 Member Data Documentation

6.344.4.1 m_func

```
ThreadStatCallback ThreadStatsManager::StatCallback::m_func
```

Definition at line 64 of file thread_stats_manager.h.

Referenced by call().

6.344.4.2 m_name

```
const char* ThreadStatsManager::StatCallback::m_name
```

Definition at line 63 of file thread_stats_manager.h.

6.344.4.3 m_user

```
UInt64 ThreadStatsManager::StatCallback::m_user
```

Definition at line 65 of file thread_stats_manager.h.

Referenced by call().

The documentation for this struct was generated from the following file:

- common/system/ **thread_stats_manager.h**

6.345 StatHist Class Reference

```
#include <stats.h>
```

Public Member Functions

- **StatHist** ()
- **StatHist** & **operator+=** (**StatHist** &stat)
- void **update** (unsigned long v)
- void **print** ()

Private Attributes

- unsigned long **n**
- unsigned long **s**
- unsigned long **s2**
- unsigned long **min**
- unsigned long **max**
- unsigned long **hist** [**HIST_MAX**]
- char **dummy** [64]

Static Private Attributes

- static const int **HIST_MAX** = 20

6.345.1 Detailed Description

Definition at line 110 of file stats.h.

6.345.2 Constructor & Destructor Documentation

6.345.2.1 StatHist()

```
StatHist::StatHist ( )  [inline]
```

Definition at line 117 of file stats.h.

References hist.

6.345.3 Member Function Documentation

6.345.3.1 operator+=()

```
StatHist & StatHist::operator+= (
            StatHist & stat )
```

Definition at line 238 of file stats.cc.

References hist, HIST_MAX, max, min, n, s, and s2.

6.345.3.2 print()

```
void StatHist::print ( )
```

Definition at line 269 of file stats.cc.

References hist, HIST_MAX, max, min, n, s, and s2.

6.345.3.3 update()

```
void StatHist::update (
            unsigned long v )
```

Definition at line 252 of file stats.cc.

References floorLog2(), hist, HIST_MAX, max, min, n, s, and s2.

6.345.4 Member Data Documentation

6.345.4.1 dummy

```
char StatHist::dummy[64]  [private]
```

Definition at line 115 of file stats.h.

6.345.4.2 hist

```
unsigned long StatHist::hist[ HIST_MAX]  [private]
```

Definition at line 114 of file stats.h.

Referenced by operator+=(), print(), StatHist(), and update().

6.345.4.3 HIST_MAX

```
const int StatHist::HIST_MAX = 20  [static], [private]
```

Definition at line 112 of file stats.h.

Referenced by operator+=(), print(), and update().

6.345.4.4 max

```
unsigned long StatHist::max  [private]
```

Definition at line 113 of file stats.h.

Referenced by operator+=(), print(), and update().

6.345.4.5 min

```
unsigned long StatHist::min  [private]
```

Definition at line 113 of file stats.h.

Referenced by operator+=(), print(), and update().

6.345.4.6 n

```
unsigned long StatHist::n  [private]
```

Definition at line 113 of file stats.h.

Referenced by operator+=(), print(), and update().

6.345.4.7 s

```
unsigned long StatHist::s  [private]
```

Definition at line 113 of file stats.h.

Referenced by operator+=(), print(), and update().

6.345.4.8 s2

```
unsigned long StatHist::s2  [private]
```

Definition at line 113 of file stats.h.

Referenced by operator+=(), print(), and update().

The documentation for this class was generated from the following files:

- common/misc/ **stats.h**
- common/misc/ **stats.cc**

6.346 statsGetterObject Struct Reference

Public Attributes

- PyObject_HEAD **StatsMetricBase** * **metric**

6.346.1 Detailed Description

Definition at line 37 of file py_stats.cc.

6.346.2 Member Data Documentation

6.346.2.1 metric

```
PyObject_HEAD  StatsMetricBase* statsGetterObject::metric
```

Definition at line 39 of file py_stats.cc.

Referenced by getStatsGetter(), and statsGetterGet().

The documentation for this struct was generated from the following file:

- common/scripting/ **py_stats.cc**

6.347 StatsManager Class Reference

```
#include <stats.h>
```

Public Types

- enum **event_type_t** {
  EVENT_MARKER = 1, **EVENT_THREAD_NAME**, **EVENT_APP_START**, **EVENT_APP_EXIT**,
  EVENT_THREAD_CREATE, **EVENT_THREAD_EXIT** }

Public Member Functions

- **StatsManager** ()
- ~**StatsManager** ()
- void **init** ()
- void **recordStats** (String prefix)
- void **registerMetric** (**StatsMetricBase** *metric)
- **StatsMetricBase** * **getMetricObject** (String objectName, **UInt32** index, String metricName)
- void **logTopology** (String component, **core_id_t** core_id, **core_id_t** master_id)
- void **logMarker** (**SubsecondTime** time, **core_id_t** core_id, **thread_id_t** thread_id, **UInt64** value0, **UInt64** value1, const char *description)
- void **logEvent** (**event_type_t** event, **SubsecondTime** time, **core_id_t** core_id, **thread_id_t** thread_id, **UInt64** value0, **UInt64** value1, const char *description)

Private Types

- typedef std::unordered_map< **UInt64**, **StatsMetricBase** * > **StatsIndexList**
- typedef std::pair< **UInt64**, **StatsIndexList** > **StatsMetricWithKey**
- typedef std::unordered_map< std::string, **StatsMetricWithKey** > **StatsMetricList**
- typedef std::unordered_map< std::string, **StatsMetricList** > **StatsObjectList**

Private Member Functions

- int **busy_handler** (int count)
- void **recordMetricName** (**UInt64** keyId, std::string objectName, std::string metricName)

Static Private Member Functions

- static int **__busy_handler** (void *self, int count)

Private Attributes

- **UInt64 m_keyid**
- **UInt64 m_prefixnum**
- sqlite3 * **m_db**
- sqlite3_stmt * **m_stmt_insert_name**
- sqlite3_stmt * **m_stmt_insert_prefix**
- sqlite3_stmt * **m_stmt_insert_value**
- **StatsObjectList m_objects**

6.347.1 Detailed Description

Definition at line 58 of file stats.h.

6.347.2 Member Typedef Documentation

6.347.2.1 StatsIndexList

typedef std::unordered_map< **UInt64**, **StatsMetricBase** *> **StatsManager::StatsIndexList** [private]

Definition at line 92 of file stats.h.

6.347.2.2 StatsMetricList

typedef std::unordered_map<std::string, **StatsMetricWithKey**> **StatsManager::StatsMetricList** [private]

Definition at line 94 of file stats.h.

6.347.2.3 StatsMetricWithKey

typedef std::pair< **UInt64**, **StatsIndexList**> **StatsManager::StatsMetricWithKey** [private]

Definition at line 93 of file stats.h.

6.347.2.4 StatsObjectList

typedef std::unordered_map<std::string, **StatsMetricList**> **StatsManager::StatsObjectList**
[private]

Definition at line 95 of file stats.h.

6.347.3 Member Enumeration Documentation

6.347.3.1 event_type_t

enum **StatsManager::event_type_t**

Enumerator

EVENT_MARKER	
EVENT_THREAD_NAME	
EVENT_APP_START	
EVENT_APP_EXIT	
EVENT_THREAD_CREATE	
EVENT_THREAD_EXIT	

Definition at line 62 of file stats.h.

6.347.4 Constructor & Destructor Documentation

6.347.4.1 StatsManager()

StatsManager::StatsManager ( )

Definition at line 43 of file stats.cc.

References getWallclockTimeCallback(), init(), and registerMetric().

6.347.4.2 ~StatsManager()

StatsManager::~StatsManager ( )

Definition at line 53 of file stats.cc.

References m_db, m_objects, m_stmt_insert_name, m_stmt_insert_prefix, and m_stmt_insert_value.

6.347.5 Member Function Documentation

6.347.5.1 __busy_handler()

```
static int StatsManager::__busy_handler (
            void * self,
            int count )  [inline], [static], [private]
```

Definition at line 98 of file stats.h.

Referenced by init().

6.347.5.2 busy_handler()

```
int StatsManager::busy_handler (
            int count )  [private]
```

Definition at line 105 of file stats.cc.

References LOG_PRINT_WARNING.

6.347.5.3 getMetricObject()

```
StatsMetricBase * StatsManager::getMetricObject (
            String objectName,
             UInt32 index,
            String metricName )
```

Definition at line 192 of file stats.cc.

References m_objects.

6.347.5.4 init()

```
void StatsManager::init ( )
```

Definition at line 70 of file stats.cc.

References __busy_handler(), db_create_stmts, db_insert_stmt_name, db_insert_stmt_prefix, db_insert_stmt_value, LOG_ASSERT_ERROR, m_db, m_objects, m_stmt_insert_name, m_stmt_insert_prefix, m_stmt_insert_value, and recordMetricName().

Referenced by StatsManager().

6.347.5.5 logEvent()

```
void StatsManager::logEvent (
              event_type_t event,
              SubsecondTime time,
              core_id_t core_id,
              thread_id_t thread_id,
              UInt64 value0,
              UInt64 value1,
             const char * description )
```

Definition at line 218 of file stats.cc.

References SubsecondTime::getFS(), LOG_ASSERT_ERROR, m_db, and SubsecondTime::MaxTime().

Referenced by logMarker().

6.347.5.6 logMarker()

```
void StatsManager::logMarker (
              SubsecondTime time,
              core_id_t core_id,
              thread_id_t thread_id,
              UInt64 value0,
              UInt64 value1,
             const char * description )  [inline]
```

Definition at line 78 of file stats.h.

References EVENT_MARKER, and logEvent().

6.347.5.7 logTopology()

```
void StatsManager::logTopology (
             String component,
              core_id_t core_id,
              core_id_t master_id )
```

Definition at line 205 of file stats.cc.

References LOG_ASSERT_ERROR, and m_db.

6.347.5.8 recordMetricName()

```
void StatsManager::recordMetricName (
             UInt64 keyId,
            std::string objectName,
            std::string metricName )  [private]
```

Definition at line 117 of file stats.cc.

References LOG_ASSERT_ERROR, and m_stmt_insert_name.

Referenced by init(), and registerMetric().

6.347.5.9 recordStats()

```
void StatsManager::recordStats (
            String prefix )
```

Definition at line 129 of file stats.cc.

References HookType::HOOK_PRE_STAT_WRITE, LOG_ASSERT_ERROR, m_db, m_objects, m_prefixnum, m_stmt_insert_prefix, and m_stmt_insert_value.

Referenced by Simulator::start(), and Simulator::∼Simulator().

6.347.5.10 registerMetric()

```
void StatsManager::registerMetric (
             StatsMetricBase * metric )
```

Definition at line 172 of file stats.cc.

References StatsMetricBase::index, LOG_ASSERT_ERROR, m_db, m_keyid, m_objects, StatsMetricBase↩ ::metricName, StatsMetricBase::objectName, and recordMetricName().

Referenced by StatsManager().

6.347.6 Member Data Documentation

6.347.6.1 m_db

```
sqlite3* StatsManager::m_db  [private]
```

Definition at line 86 of file stats.h.

Referenced by init(), logEvent(), logTopology(), recordStats(), registerMetric(), and ∼StatsManager().

6.347.6.2 m_keyid

```
UInt64 StatsManager::m_keyid  [private]
```

Definition at line 83 of file stats.h.

Referenced by registerMetric().

6.347.6.3 m_objects

```
StatsObjectList StatsManager::m_objects  [private]
```

Definition at line 96 of file stats.h.

Referenced by getMetricObject(), init(), recordStats(), registerMetric(), and ~StatsManager().

6.347.6.4 m_prefixnum

```
UInt64 StatsManager::m_prefixnum  [private]
```

Definition at line 84 of file stats.h.

Referenced by recordStats().

6.347.6.5 m_stmt_insert_name

```
sqlite3_stmt* StatsManager::m_stmt_insert_name  [private]
```

Definition at line 87 of file stats.h.

Referenced by init(), recordMetricName(), and ~StatsManager().

6.347.6.6 m_stmt_insert_prefix

```
sqlite3_stmt* StatsManager::m_stmt_insert_prefix  [private]
```

Definition at line 88 of file stats.h.

Referenced by init(), recordStats(), and ~StatsManager().

6.347.6.7 m_stmt_insert_value

```
sqlite3_stmt* StatsManager::m_stmt_insert_value  [private]
```

Definition at line 89 of file stats.h.

Referenced by init(), recordStats(), and ~StatsManager().

The documentation for this class was generated from the following files:

- common/misc/ **stats.h**
- common/misc/ **stats.cc**

6.348 StatsMetric< T > Class Template Reference

```
#include <stats.h>
```

Inheritance diagram for StatsMetric< T >:

Public Member Functions

- **StatsMetric** (String _objectName, **UInt32** _index, String _metricName, T *_metric)
- virtual **UInt64** **recordMetric** ()
- virtual bool **isDefault** ()

Public Attributes

- T * **metric**

6.348.1 Detailed Description

template<class T>
class StatsMetric< T >

Definition at line 25 of file stats.h.

6.348.2 Constructor & Destructor Documentation

6.348.2.1 StatsMetric()

```
template<class T >
StatsMetric< T >::  StatsMetric (
             String _objectName,
              UInt32 _index,
             String _metricName,
             T * _metric )  [inline]
```

Definition at line 29 of file stats.h.

6.348.3 Member Function Documentation

6.348.3.1 isDefault()

```
template<class T >
virtual bool  StatsMetric< T >::isDefault ( )  [inline], [virtual]
```

Reimplemented from **StatsMetricBase** (p. 1291).

Definition at line 36 of file stats.h.

References StatsMetric< T >::recordMetric().

6.348.3.2 recordMetric()

```
template<class T >
virtual  UInt64  StatsMetric< T >::recordMetric ( )  [inline], [virtual]
```

Implements **StatsMetricBase** (p. 1291).

Definition at line 32 of file stats.h.

References StatsMetric< T >::metric.

Referenced by StatsMetric< T >::isDefault().

6.348.4 Member Data Documentation

6.348.4.1 metric

```
template<class T >
T*  StatsMetric< T >::metric
```

Definition at line 28 of file stats.h.

Referenced by StatsMetric< T >::recordMetric().

The documentation for this class was generated from the following file:

- common/misc/ **stats.h**

6.349 StatsMetricBase Class Reference

```
#include <stats.h>
```

Inheritance diagram for StatsMetricBase:

Public Member Functions

- **StatsMetricBase** (String _objectName, **UInt32** _index, String _metricName)
- virtual ~**StatsMetricBase** ()
- virtual **UInt64** **recordMetric** ()=0
- virtual bool **isDefault** ()

Public Attributes

- String **objectName**
- **UInt32** **index**
- String **metricName**

6.349.1 Detailed Description

Definition at line 9 of file stats.h.

6.349.2 Constructor & Destructor Documentation

6.349.2.1 StatsMetricBase()

```
StatsMetricBase::StatsMetricBase (
            String _objectName,
             UInt32 _index,
            String _metricName )  [inline]
```

Definition at line 15 of file stats.h.

6.349.2.2 ~StatsMetricBase()

```
virtual StatsMetricBase::~StatsMetricBase ( )  [inline], [virtual]
```

Definition at line 18 of file stats.h.

6.349.3 Member Function Documentation

6.349.3.1 isDefault()

```
virtual bool StatsMetricBase::isDefault ( )  [inline], [virtual]
```

Reimplemented in **StatsMetric**< **T** > (p. 1289).

Definition at line 20 of file stats.h.

6.349.3.2 recordMetric()

```
virtual  UInt64 StatsMetricBase::recordMetric ( )  [pure virtual]
```

Implemented in **StatsMetricCallback** (p. 1293), and **StatsMetric**< **T** > (p. 1289).

Referenced by ThreadStatNamedStat::callback(), getStatsValue(), hook_print_core0_ipc(), and statsGetterGet().

6.349.4 Member Data Documentation

6.349.4.1 index

```
UInt32 StatsMetricBase::index
```

Definition at line 13 of file stats.h.

Referenced by StatsMetricCallback::recordMetric(), and StatsManager::registerMetric().

6.349.4.2 metricName

```
String StatsMetricBase::metricName
```

Definition at line 14 of file stats.h.

Referenced by StatsMetricCallback::recordMetric(), and StatsManager::registerMetric().

6.349.4.3 objectName

```
String StatsMetricBase::objectName
```

Definition at line 12 of file stats.h.

Referenced by StatsMetricCallback::recordMetric(), and StatsManager::registerMetric().

The documentation for this class was generated from the following file:

- common/misc/ **stats.h**

6.350 StatsMetricCallback Class Reference

```
#include <stats.h>
```

Inheritance diagram for StatsMetricCallback:

Public Member Functions

- **StatsMetricCallback** (String _objectName, **UInt32** _index, String _metricName, **StatsCallback** _func, **UInt64** _arg)
- virtual **UInt64** **recordMetric** ()

Public Attributes

- **StatsCallback func**
- **UInt64 arg**

6.350.1 Detailed Description

Definition at line 43 of file stats.h.

6.350.2 Constructor & Destructor Documentation

6.350.2.1 StatsMetricCallback()

```
StatsMetricCallback::StatsMetricCallback (
             String _objectName,
              UInt32 _index,
             String _metricName,
              StatsCallback _func,
              UInt64 _arg )  [inline]
```

Definition at line 48 of file stats.h.

6.350.3 Member Function Documentation

6.350.3.1 recordMetric()

```
virtual  UInt64 StatsMetricCallback::recordMetric ( )  [inline], [virtual]
```

Implements **StatsMetricBase** (p. 1291).

Definition at line 51 of file stats.h.

References arg, func, StatsMetricBase::index, StatsMetricBase::metricName, and StatsMetricBase::objectName.

6.350.4 Member Data Documentation

6.350.4.1 arg

```
UInt64 StatsMetricCallback::arg
```

Definition at line 47 of file stats.h.

Referenced by recordMetric().

6.350.4.2 func

```
StatsCallback StatsMetricCallback::func
```

Definition at line 46 of file stats.h.

Referenced by recordMetric().

The documentation for this class was generated from the following file:

- common/misc/ **stats.h**

6.351 subsecond_time_s Struct Reference

```
#include <subsecond_time_c.h>
```

Public Attributes

- uint64_t **m_time**

6.351.1 Detailed Description

Definition at line 17 of file subsecond_time_c.h.

6.351.2 Member Data Documentation

6.351.2.1 m_time

```
uint64_t subsecond_time_s::m_time
```

Definition at line 28 of file subsecond_time_c.h.

Referenced by SubsecondTime::operator subsecond_time_t().

The documentation for this struct was generated from the following file:

- common/misc/ **subsecond_time_c.h**

6.352 SubsecondTime Class Reference

`#include <subsecond_time.h>`

Public Member Functions

- **SubsecondTime** ()
- **SubsecondTime** (const **SubsecondTime** &_time)
- **SubsecondTime** (uint64_t multiplier, const **SubsecondTime** &_time)
- **SubsecondTime** (const **subsecond_time_t** &sstime)
- **UInt64 getFS** () const
- **UInt64 getPS** () const
- **UInt64 getNS** () const
- **UInt64 getUS** () const
- **UInt64 getMS** () const
- **UInt64 getSEC** () const
- **SubsecondTime** & **operator+=** (const **SubsecondTime** &rhs)
- **SubsecondTime** & **operator-=** (const **SubsecondTime** &rhs)
- **SubsecondTime** & **operator<<=** (uint64_t rhs)
- template<class T >
  SubsecondTime & **operator*=** (T rhs)
- **SubsecondTime operator/** (const uint64_t &divisor) const
- **operator subsecond_time_t** () const
- uint64_t **getInternalDataForced** (void) const
- void **setInternalDataForced** (uint64_t new_time)
- **SubsecondTime** & **operator*=** (const **SubsecondTime** &rhs)
- **SubsecondTime** & **operator/=** (const **SubsecondTime** &rhs)
- **SubsecondTime operator*** (const **SubsecondTime** &rhs)
- **SubsecondTime operator*** (float rhs)
- **SubsecondTime operator/** (const **SubsecondTime** &rhs)
- **SubsecondTime operator%** (const **SubsecondTime** &rhs)

Static Public Member Functions

- static const **SubsecondTime FS** (uint64_t fs=1)
- static const **SubsecondTime PS** (uint64_t ps=1)
- static const **SubsecondTime NS** (uint64_t ns=1)
- static const **SubsecondTime US** (uint64_t us=1)
- static const **SubsecondTime MS** (uint64_t ms=1)
- static const **SubsecondTime SEC** (uint64_t sec=1)
- static const **SubsecondTime Zero** ()
- static const **SubsecondTime MaxTime** ()
- static const **SubsecondTime FSfromFloat** (float fs)
- static const **SubsecondTime PSfromFloat** (float ps)
- static const **SubsecondTime NSfromFloat** (float ns)
- static const **SubsecondTime USfromFloat** (float us)
- static const **SubsecondTime MSfromFloat** (float ms)
- static const **SubsecondTime SECfromFloat** (float sec)
- static uint64_t **divideRounded** (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)

Private Member Functions

- **SubsecondTime** (uint64_t _time)

Private Attributes

- uint64_t **m_time**

Static Private Attributes

- static const uint64_t **FS_1** = 1
- static const uint64_t **PS_1** = **FS_1** ∗ 1000
- static const uint64_t **NS_1** = **PS_1** ∗ 1000
- static const uint64_t **US_1** = **NS_1** ∗ 1000
- static const uint64_t **MS_1** = **US_1** ∗ 1000
- static const uint64_t **SEC_1** = **MS_1** ∗ 1000

Friends

- class **ComponentPeriod**
- bool **operator==** (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)
- bool **operator!=** (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)
- bool **operator**< (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)
- bool **operator<=** (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)
- bool **operator>=** (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)
- bool **operator**> (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)
- void **atomic_add_subsecondtime** (**SubsecondTime** &src_dest, const **SubsecondTime** & **src**)
- std::ostream & **operator**<< (std::ostream &os, const **SubsecondTime** &time)
- std::ostream & **operator**<< (std::ostream &os, const **ComponentTime** &time)

6.352.1 Detailed Description

Definition at line 46 of file subsecond_time.h.

6.352.2 Constructor & Destructor Documentation

6.352.2.1 SubsecondTime() [1/5]

```
SubsecondTime::SubsecondTime ( )  [inline]
```

Definition at line 66 of file subsecond_time.h.

Referenced by FS(), FSfromFloat(), MaxTime(), MS(), MSfromFloat(), NS(), NSfromFloat(), operator/(), PS(), P↩ SfromFloat(), SEC(), SECfromFloat(), US(), USfromFloat(), and Zero().

6.352.2.2 SubsecondTime() [2/5]

```
SubsecondTime::SubsecondTime (
            const  SubsecondTime & _time )  [inline]
```

Definition at line 69 of file subsecond_time.h.

6.352.2.3 SubsecondTime() [3/5]

```
SubsecondTime::SubsecondTime (
            uint64_t multiplier,
            const  SubsecondTime & _time )  [inline]
```

Definition at line 72 of file subsecond_time.h.

6.352.2.4 SubsecondTime() [4/5]

```
SubsecondTime::SubsecondTime (
            const  subsecond_time_t & sstime )  [inline]
```

Definition at line 75 of file subsecond_time.h.

6.352.2.5 SubsecondTime() [5/5]

```
SubsecondTime::SubsecondTime (
            uint64_t _time )  [inline], [explicit], [private]
```

Definition at line 204 of file subsecond_time.h.

6.352.3 Member Function Documentation

6.352.3.1 divideRounded()

```
static uint64_t SubsecondTime::divideRounded (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [inline], [static]
```

Definition at line 195 of file subsecond_time.h.

References m_time.

Referenced by MagicServer::disablePerformance(), ComponentTime::getCycleCount(), TraceThread::handle↩ EmuFunc(), MicroOpPerformanceModel::handleInstruction(), handleRdtsc(), hook_print_core0_ipc(), RobTimer↩ ::issueInstruction(), RobSmtTimer::issueInstruction(), IntervalTimer::issueMemOp(), FastForwardPerformance↩ Manager::recalibrateInstructionsCallback(), SamplingManager::recalibrateInstructionsCallback(), Scheduler↩ Sequential::results_on_file(), SchedulerSequential::results_on_screen(), RobPerformanceModel::simulate(), Rob↩ SmtPerformanceModel::simulate(), and LoopTracer::traceInstruction().

6.352.3.2 FS()

```
static const SubsecondTime SubsecondTime::FS (
            uint64_t fs = 1 ) [inline], [static]
```

Definition at line 49 of file subsecond_time.h.

References FS_1, and SubsecondTime().

Referenced by DramPerfModelConstant::DramPerfModelConstant(), DramPerfModelNormal::DramPerfModelNormal(), DramPerfModelReadWrite::DramPerfModelReadWrite(), hook_print_core0_ipc(), NormalTimeDistribution::next(), and QueueModelHistoryList::QueueModelHistoryList().

6.352.3.3 FSfromFloat()

```
static const SubsecondTime SubsecondTime::FSfromFloat (
            float fs ) [inline], [static]
```

Definition at line 58 of file subsecond_time.h.

References FS_1, and SubsecondTime().

6.352.3.4 getFS()

```
UInt64 SubsecondTime::getFS ( ) const [inline]
```

Definition at line 79 of file subsecond_time.h.

References FS_1, and m_time.

Referenced by ThreadStatsManager::calculateWaitingCosts(), getFrequency(), getTime(), hookCallbackSubsecondTime(), hookCallbackSyscallEnter(), hookCallbackSyscallExit(), hookCallbackThreadMigrateType(), hookCallbackThreadResumeType(), hookCallbackThreadStallType(), hookCallbackThreadTimeType(), StatsManager::logEvent(), makeStatsValue< ComponentTime >(), makeStatsValue< SubsecondTime >(), ThreadStatsManager::metricCallback(), and ThreadStatsManager::ThreadStats::update().

6.352.3.5 getInternalDataForced()

```
uint64_t SubsecondTime::getInternalDataForced (
            void ) const [inline]
```

Definition at line 132 of file subsecond_time.h.

References m_time.

Referenced by QueueModelHistoryList::getQueueUtilization().

6.352.3.6 getMS()

```
UInt64 SubsecondTime::getMS ( ) const  [inline]
```

Definition at line 83 of file subsecond_time.h.

References m_time, and MS_1.

6.352.3.7 getNS()

```
UInt64 SubsecondTime::getNS ( ) const  [inline]
```

Definition at line 81 of file subsecond_time.h.

References m_time, and NS_1.

Referenced by BarrierSyncServer::barrierRelease(), PeriodicSampling::callbackDetailed(), MagicServer::disable↩Performance(), DramPerfModel::dramAccessed(), lite::emuClockGettime(), lite::emuGettimeofday(), Trace↩Thread::handleEmuFunc(), ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), hook↩_print_core0_ipc(), printCodeCacheStats(), printCodeCacheTrace(), ParametricDramDirectoryMSI::CacheCntlr↩::processMemOpFromCore(), SchedulerSequential::results_on_file(), SchedulerSequential::results_on_screen(), and SyscallMdl::runEnter().

6.352.3.8 getPS()

```
UInt64 SubsecondTime::getPS ( ) const  [inline]
```

Definition at line 80 of file subsecond_time.h.

References m_time, and PS_1.

Referenced by QueueModelWindowedMG1::addItem(), QueueModelWindowedMG1::computeQueueDelay(), RobTimer::issueInstruction(), RobSmtTimer::issueInstruction(), and SchedulerPinnedBase::reschedule().

6.352.3.9 getSEC()

```
UInt64 SubsecondTime::getSEC ( ) const  [inline]
```

Definition at line 84 of file subsecond_time.h.

References m_time, and SEC_1.

6.352.3.10 getUS()

```
UInt64 SubsecondTime::getUS ( ) const  [inline]
```

Definition at line 82 of file subsecond_time.h.

References m_time, and US_1.

6.352.3.11 MaxTime()

```
static const  SubsecondTime SubsecondTime::MaxTime ( )  [inline], [static]
```

Definition at line 56 of file subsecond_time.h.

References SubsecondTime().

Referenced by BarrierSyncServer::barrierRelease(), PeriodicSampling::callbackDetailed(), CarbonMutexTrylock(), ParametricDramDirectoryMSI::CacheCntlr::cleanupMshr(), QueueModelHistoryList::computeUsingHistoryList(), RobTimer::countOutstandingMemop(), RobSmtTimer::countOutstandingMemop(), ThreadManager::create↩Thread_unlocked(), FastForwardPerformanceManager::disable(), ShmemPerf::disable(), SamplingManager↩::disableFastForward(), DramCache::doAccess(), RobTimer::doCommit(), RobSmtTimer::doCommit(), Rob↩Timer::doDispatch(), RobSmtTimer::doDispatch(), RobTimer::doIssue(), RobSmtTimer::doIssue(), RobTimer↩::execute(), RobSmtTimer::executeCycle(), SamplingManager::getCoreHistoricCPI(), SimFutex::getNextTimeout(), SyscallServer::getNextTimeout(), ContentionModel::getTagCompletionTime(), SyscallServer::handleFutexCall(), lite::handleMemoryReadFaultinjection(), lite::handleMemoryWriteFaultinjection(), RobTimer::RobEntry::init(), RobSmtTimer::RobEntry::init(), Core::initiateMemoryAccess(), RobTimer::issueInstruction(), RobSmtTimer↩::issueInstruction(), StatsManager::logEvent(), MagicServer::Magic_unlocked(), PthreadEmu::MutexTrylock(), TraceManager::newThread(), ThreadManager::onThreadExit(), RobTimer::printRob(), RobSmtTimer::print↩Rob(), ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore(), Core::readInstructionMemory(), RobSmtTimer::setDependencies(), BarrierSyncServer::setFastForward(), MagicServer::setFrequency(), Rob↩SmtTimer::setStoreAddressProducers(), TraceManager::signalDone(), RobTimer::simulate(), RobSmtTimer↩::synchronize(), RobSmtTimer::tryDispatch(), RobSmtTimer::tryIssue(), and ThreadStatsManager::update().

6.352.3.12 MS()

```
static const  SubsecondTime SubsecondTime::MS (
            uint64_t ms = 1 )  [inline], [static]
```

Definition at line 53 of file subsecond_time.h.

References MS_1, and SubsecondTime().

6.352.3.13 MSfromFloat()

```
static const  SubsecondTime SubsecondTime::MSfromFloat (
            float ms )  [inline], [static]
```

Definition at line 62 of file subsecond_time.h.

References MS_1, and SubsecondTime().

6.352.3.14 NS()

```
static const  SubsecondTime SubsecondTime::NS (
            uint64_t ns = 1 )  [inline], [static]
```

Definition at line 51 of file subsecond_time.h.

References NS_1, and SubsecondTime().

Referenced by BarrierSyncClient::BarrierSyncClient(), BarrierSyncServer::BarrierSyncServer(), codeCache↩ PeriodicCallback(), QueueModelBasic::computeQueueDelay(), Directory::Directory(), DvfsManager::Dvfs↩ Manager(), SyscallServer::handleFutexCall(), Network::netRecv(), SyscallMdl::runEnter(), SyncServer::Sync↩ Server(), and SyscallServer::SyscallServer().

6.352.3.15 NSfromFloat()

```
static const  SubsecondTime SubsecondTime::NSfromFloat (
            float ns )  [inline], [static]
```

Definition at line 60 of file subsecond_time.h.

References NS_1, and SubsecondTime().

6.352.3.16 operator subsecond_time_t()

```
SubsecondTime::operator  subsecond_time_t ( ) const  [inline]
```

Definition at line 124 of file subsecond_time.h.

References subsecond_time_s::m_time, and m_time.

6.352.3.17 operator%()

```
 SubsecondTime SubsecondTime::operator% (
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 188 of file subsecond_time.h.

References m_time.

6.352.3.18 operator∗() [1/2]

```
SubsecondTime SubsecondTime::operator* (
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 170 of file subsecond_time.h.

References m_time.

6.352.3.19 operator∗() [2/2]

```
SubsecondTime SubsecondTime::operator* (
            float rhs )  [inline]
```

Definition at line 176 of file subsecond_time.h.

References m_time.

6.352.3.20 operator∗=() [1/2]

```
SubsecondTime& SubsecondTime::operator*= (
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 157 of file subsecond_time.h.

References m_time.

6.352.3.21 operator∗=() [2/2]

```
template<class T >
SubsecondTime& SubsecondTime::operator*= (
            T rhs )  [inline]
```

Definition at line 110 of file subsecond_time.h.

References m_time.

6.352.3.22 operator+=()

```
SubsecondTime& SubsecondTime::operator+= (
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 89 of file subsecond_time.h.

References m_time.

6.352.3.23 operator-=()

```
SubsecondTime& SubsecondTime::operator-= (
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 95 of file subsecond_time.h.

References m_time.

6.352.3.24 operator/() [1/2]

```
SubsecondTime SubsecondTime::operator/ (
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 182 of file subsecond_time.h.

References m_time.

6.352.3.25 operator/() [2/2]

```
SubsecondTime SubsecondTime::operator/ (
            const uint64_t & divisor ) const  [inline]
```

Definition at line 116 of file subsecond_time.h.

References m_time, and SubsecondTime().

6.352.3.26 operator/=()

```
SubsecondTime& SubsecondTime::operator/= (
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 163 of file subsecond_time.h.

References m_time.

6.352.3.27 operator<<=()

```
SubsecondTime& SubsecondTime::operator<<= (
            uint64_t rhs )  [inline]
```

Definition at line 101 of file subsecond_time.h.

References m_time.

6.352.3.28 PS()

```
static const  SubsecondTime SubsecondTime::PS (
            uint64_t ps = 1 )  [inline], [static]
```

Definition at line 50 of file subsecond_time.h.

References PS_1, and SubsecondTime().

Referenced by QueueModelWindowedMG1::computeQueueDelay(), and SchedulerPinnedBase::reschedule().

6.352.3.29 PSfromFloat()

```
static const  SubsecondTime SubsecondTime::PSfromFloat (
            float ps )  [inline], [static]
```

Definition at line 59 of file subsecond_time.h.

References PS_1, and SubsecondTime().

6.352.3.30 SEC()

```
static const  SubsecondTime SubsecondTime::SEC (
            uint64_t sec = 1 )  [inline], [static]
```

Definition at line 54 of file subsecond_time.h.

References SEC_1, and SubsecondTime().

Referenced by ComponentPeriod::fromFreqHz(), SyscallServer::handleFutexCall(), SyscallMdl::runEnter(), and ComponentPeriod::setPeriodFromFreqHz().

6.352.3.31 SECfromFloat()

```
static const  SubsecondTime SubsecondTime::SECfromFloat (
            float sec )  [inline], [static]
```

Definition at line 63 of file subsecond_time.h.

References SEC_1, and SubsecondTime().

6.352.3.32 setInternalDataForced()

```
void SubsecondTime::setInternalDataForced (
            uint64_t new_time )  [inline]
```

Definition at line 138 of file subsecond_time.h.

References m_time.

Referenced by MovingGeometricMean< T >::compute(), SchedulerSequential::results_on_file(), and SchedulerSequential::results_on_screen().

6.352.3.33 US()

```
static const  SubsecondTime SubsecondTime::US (
            uint64_t us = 1 )  [inline], [static]
```

Definition at line 52 of file subsecond_time.h.

References SubsecondTime(), and US_1.

Referenced by ComponentBandwidth::getLatency(), and ComponentBandwidth::getRoundedLatency().

6.352.3.34 USfromFloat()

```
static const  SubsecondTime SubsecondTime::USfromFloat (
            float us )  [inline], [static]
```

Definition at line 61 of file subsecond_time.h.

References SubsecondTime(), and US_1.

6.352.3.35 Zero()

```
static const SubsecondTime SubsecondTime::Zero ( ) [inline], [static]
```

Definition at line 55 of file subsecond_time.h.

References SubsecondTime().

Referenced by SyncClient::__mutexLock(), NucaCache::accessDataArray(), DramCache::accessDataArray(), Core::accessMemory(), Core::accessMemoryFast(), ParametricDramDirectoryMSI::MemoryManager::accessTLB(), TraceThread::addDetailedMemoryInfo(), SyscallServer::applyRescheduleCost(), SyncClient::barrierWait(), ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), PeriodicSampling::callbackDetailed(), PeriodicSampling::callbackFastForward(), MovingGeometricMean< T >::compute(), NetworkModelEMeshHopByHop::computeEjectionPortQueueDelay(), NetworkModelEMeshHopByHop::computeInjectionPortQueueDelay(), NetworkModelEMeshHopByHop::computeLatency(), QueueModelBasic::computeQueueDelay(), QueueModelWindowedMG1::computeQueueDelay(), QueueModelHistoryList::computeUsingHistoryList(), SyncClient::condBroadcast(), SyncClient::condSignal(), SyncClient::condWait(), MemoryManagerFast::coreInitiateMemoryAccess(), FastForwardPerformanceManager::disable(), SamplingManager::disableFastForward(), MagicServer::disablePerformance(), FastNehalem::Dram::Dram(), DynamicMicroOp::DynamicMicroOp(), SamplingManager::enableFastForward(), RobTimer::execute(), RobSmtTimer::executeCycle(), FastforwardPerformanceModel::FastforwardPerformanceModel(), DramPerfModelConstant::getAccessLatency(), DramPerfModelNormal::getAccessLatency(), DramPerfModelReadWrite::getAccessLatency(), ContentionModel::getCompletionTime(), SamplingManager::getCoreHistoricCPI(), ParametricDramDirectoryMSI::MemoryManager::getCost(), SyncInstruction::getCost(), SpawnInstruction::getCost(), MemoryManagerFast::getL1HitLatency(), CachePerfModelParallel::getLatency(), CachePerfModelSequential::getLatency(), DirectoryEntryLimitedNoBroadcast< DirectorySharers >::getLatency(), DirectoryEntryLimitless< DirectorySharers >::getLatency(), DynamicMicroOp::getPeriod(), QueueModelHistoryList::getQueueUtilization(), handleCheckScheduled(), PerformanceModel::handleIdleInstruction(), MicroOpPerformanceModel::handleInstruction(), lite::handleMemoryReadDetailed(), lite::handleMemoryReadDetailedIssue(), lite::handleMemoryReadFaultinjection(), lite::handleMemoryWriteDetailed(), lite::handleMemoryWriteDetailedIssue(), lite::handleMemoryWriteFaultinjection(), RobTimer::RobEntry::init(), RobSmtTimer::RobEntry::init(), TraceManager::init(), initCodeCacheTracing(), RobSmtTimer::initializeThread(), Core::initiateMemoryAccess(), ParametricDramDirectoryMSI::CacheCntlr::insertCacheBlock(), FastForwardPerformanceManager::instr_count(), SamplingManager::instr_count(), IntervalTimer::IntervalTimer(), MicroOpPerformanceModel::MicroOpPerformanceModel(), PthreadEmu::MutexLock(), SyncClient::mutexTrylock(), PthreadEmu::MutexTrylock(), SyncClient::mutexUnlock(), SyncServer::mutexUnlock(), PthreadEmu::MutexUnlock(), Core::nativeMemOp(), OneIPCPerformanceModel::OneIPCPerformanceModel(), PeriodicSampling::PeriodicSampling(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache(), lite::pthreadAfter(), QueueModelHistoryList::QueueModelHistoryList(), SimThreadManager::quitSimThreads(), SamplingManager::recalibrateInstructionsCallback(), BarrierSyncServer::releaseThread(), ShmemPerf::reset(), RobSmtTimer::returnLatency(), RobTimer::RobTimer(), NetworkModelEMeshHopByHop::routePacket(), TraceThread::run(), PerformanceModel::setElapsedTime(), ShmemPerfModel::ShmemPerfModel(), RobTimer::simulate(), ThreadManager::spawnThread(), FastForwardPerformanceManager::step(), PeriodicSampling::stepFastForward(), BarrierSyncClient::synchronize(), PerformanceModel::synchronize(), SchedulerPinnedBase::threadSetAffinity(), BottleGraphManager::threadStart(), SchedulerPinnedBase::threadYield(), ThreadStatsManager::ThreadStats::update(), ParametricDramDirectoryMSI::CacheCntlr::updateCacheBlock(), ParametricDramDirectoryMSI::CacheCntlr::updateUncoreStatistics(), and NetworkModelBusGlobal::useBus().

6.352.4 Friends And Related Function Documentation

6.352.4.1 atomic_add_subsecondtime

```
void atomic_add_subsecondtime (
              SubsecondTime & src_dest,
             const SubsecondTime & src ) [friend]
```

Definition at line 282 of file subsecond_time.h.

6.352.4.2 ComponentPeriod

```
friend class  ComponentPeriod  [friend]
```

Definition at line 201 of file subsecond_time.h.

6.352.4.3 operator"!=

```
bool operator!= (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [friend]
```

Definition at line 261 of file subsecond_time.h.

6.352.4.4 operator<

```
bool operator< (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [friend]
```

Definition at line 265 of file subsecond_time.h.

6.352.4.5 operator<< [1/2]

```
std::ostream& operator<< (
            std::ostream & os,
            const  ComponentTime & time )  [friend]
```

Definition at line 631 of file subsecond_time.h.

6.352.4.6 operator<< [2/2]

```
std::ostream& operator<< (
            std::ostream & os,
            const  SubsecondTime & time )  [friend]
```

Definition at line 8 of file subsecond_time.cc.

6.352.4.7 operator<=

```
bool operator<= (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [friend]
```

Definition at line 269 of file subsecond_time.h.

6.352.4.8 operator==

```
bool operator== (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [friend]
```

Definition at line 257 of file subsecond_time.h.

6.352.4.9 operator>

```
bool operator> (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [friend]
```

Definition at line 277 of file subsecond_time.h.

6.352.4.10 operator>=

```
bool operator>= (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [friend]
```

Definition at line 273 of file subsecond_time.h.

6.352.5 Member Data Documentation

6.352.5.1 FS_1

```
const uint64_t SubsecondTime::FS_1 = 1  [static], [private]
```

Definition at line 210 of file subsecond_time.h.

Referenced by FS(), FSfromFloat(), and getFS().

6.352.5.2 m_time

```
uint64_t SubsecondTime::m_time  [private]
```

Definition at line 217 of file subsecond_time.h.

Referenced by atomic_add_subsecondtime(), divideRounded(), getFS(), getInternalDataForced(), getMS(), getNS(), ComponentPeriod::getPeriodInFreqMHz(), getPS(), getSEC(), getUS(), operator subsecond_time_t(), operator%(), operator*(), operator*=(), operator+=(), operator-=(), operator/(), operator/=(), operator<(), operator<<(), operator<<=(), operator==(), and setInternalDataForced().

6.352.5.3 MS_1

```
const uint64_t SubsecondTime::MS_1 =  US_1 * 1000  [static], [private]
```

Definition at line 214 of file subsecond_time.h.

Referenced by getMS(), MS(), and MSfromFloat().

6.352.5.4 NS_1

```
const uint64_t SubsecondTime::NS_1 =  PS_1 * 1000  [static], [private]
```

Definition at line 212 of file subsecond_time.h.

Referenced by getNS(), NS(), and NSfromFloat().

6.352.5.5 PS_1

```
const uint64_t SubsecondTime::PS_1 =  FS_1 * 1000  [static], [private]
```

Definition at line 211 of file subsecond_time.h.

Referenced by getPS(), PS(), and PSfromFloat().

6.352.5.6 SEC_1

```
const uint64_t SubsecondTime::SEC_1 =  MS_1 * 1000  [static], [private]
```

Definition at line 215 of file subsecond_time.h.

Referenced by getSEC(), SEC(), and SECfromFloat().

6.352.5.7 US_1

```
const uint64_t SubsecondTime::US_1 =  NS_1 * 1000  [static], [private]
```

Definition at line 213 of file subsecond_time.h.

Referenced by ComponentPeriod::getPeriodInFreqMHz(), getUS(), US(), and USfromFloat().

The documentation for this class was generated from the following file:

- common/misc/ **subsecond_time.h**

6.353 SubsecondTimeCycleConverter Class Reference

```
#include <subsecond_time.h>
```

Public Member Functions

- **SubsecondTimeCycleConverter** (const **ComponentPeriod** *period)
- **SubsecondTime cyclesToSubsecondTime** (**UInt64** cycles) const
- **UInt64 subsecondTimeToCycles** (**SubsecondTime** time) const

Private Member Functions

- **SubsecondTimeCycleConverter** ()

Private Attributes

- const **ComponentPeriod** * **m_period**

6.353.1 Detailed Description

Definition at line 362 of file subsecond_time.h.

6.353.2 Constructor & Destructor Documentation

6.353.2.1 SubsecondTimeCycleConverter() [1/2]

```
SubsecondTimeCycleConverter::SubsecondTimeCycleConverter (
            const  ComponentPeriod * period )  [inline]
```

Definition at line 364 of file subsecond_time.h.

6.353.2.2 SubsecondTimeCycleConverter() [2/2]

```
SubsecondTimeCycleConverter::SubsecondTimeCycleConverter ( )  [inline], [private]
```

Definition at line 377 of file subsecond_time.h.

6.353.3 Member Function Documentation

6.353.3.1 cyclesToSubsecondTime()

```
SubsecondTime SubsecondTimeCycleConverter::cyclesToSubsecondTime (
            UInt64 cycles ) const  [inline]
```

Definition at line 367 of file subsecond_time.h.

References m_period.

Referenced by ContentionModel::getBarrierCompletionTime(), ContentionModel::getCompletionTime(), ContentionModel::getNumUsed(), ContentionModel::getStartTime(), and ContentionModel::hasFreeSlot().

6.353.3.2 subsecondTimeToCycles()

```
UInt64 SubsecondTimeCycleConverter::subsecondTimeToCycles (
            SubsecondTime time ) const  [inline]
```

Definition at line 371 of file subsecond_time.h.

References m_period.

Referenced by ContentionModel::getBarrierCompletionTime(), ContentionModel::getCompletionTime(), and ContentionModel::getStartTime().

6.353.4 Member Data Documentation

6.353.4.1 m_period

```
const  ComponentPeriod* SubsecondTimeCycleConverter::m_period  [private]
```

Definition at line 380 of file subsecond_time.h.

Referenced by cyclesToSubsecondTime(), and subsecondTimeToCycles().

The documentation for this class was generated from the following file:

- common/misc/ **subsecond_time.h**

6.354 SyncClient Class Reference

`#include <sync_client.h>`

Public Member Functions

- **SyncClient** (**Thread** ∗)
- ∼**SyncClient** ()
- void **mutexInit** (**carbon_mutex_t** ∗mux)
- **SubsecondTime** **mutexLock** (**carbon_mutex_t** ∗mux, **SubsecondTime** delay= **SubsecondTime::Zero**())
- std::pair< **SubsecondTime**, bool > **mutexTrylock** (**carbon_mutex_t** ∗mux)
- **SubsecondTime** **mutexUnlock** (**carbon_mutex_t** ∗mux, **SubsecondTime** delay= **SubsecondTime::Zero**())
- void **condInit** (**carbon_cond_t** ∗cond)
- **SubsecondTime** **condWait** (**carbon_cond_t** ∗cond, **carbon_mutex_t** ∗mux)
- **SubsecondTime** **condSignal** (**carbon_cond_t** ∗cond)
- **SubsecondTime** **condBroadcast** (**carbon_cond_t** ∗cond)
- void **barrierInit** (**carbon_barrier_t** ∗barrier, **UInt32** count)
- **SubsecondTime** **barrierWait** (**carbon_barrier_t** ∗barrier)

Private Member Functions

- std::pair< **SubsecondTime**, bool > **__mutexLock** (**carbon_mutex_t** ∗mux, bool tryLock, **SubsecondTime** delay= **SubsecondTime::Zero**())

Private Attributes

- **Thread** ∗ **m_thread**
- **SyncServer** ∗ **m_server**

6.354.1 Detailed Description

Definition at line 11 of file sync_client.h.

6.354.2 Constructor & Destructor Documentation

6.354.2.1 SyncClient()

```
SyncClient::SyncClient (
            Thread * thread )
```

Definition at line 12 of file sync_client.cc.

6.354.2.2 ~SyncClient()

```
SyncClient::~SyncClient ( )
```

Definition at line 18 of file sync_client.cc.

6.354.3 Member Function Documentation

6.354.3.1 __mutexLock()

```
std::pair< SubsecondTime, bool > SyncClient::__mutexLock (
            carbon_mutex_t * mux,
            bool tryLock,
            SubsecondTime delay = SubsecondTime::Zero() ) [private]
```

Definition at line 39 of file sync_client.cc.

References Thread::getCore(), PerformanceModel::getElapsedTime(), Thread::getId(), Core::getPerformanceModel(), m_server, SyncServer::mutexLock(), SyncInstruction::PTHREAD_MUTEX, PerformanceModel::queuePseudoInstruction(), Thread::reschedule(), and SubsecondTime::Zero().

Referenced by mutexLock(), and mutexTrylock().

6.354.3.2 barrierInit()

```
void SyncClient::barrierInit (
            carbon_barrier_t * barrier,
            UInt32 count )
```

Definition at line 111 of file sync_client.cc.

References SyncServer::barrierInit(), Thread::getId(), and m_server.

Referenced by CarbonBarrierInit().

6.354.3.3 barrierWait()

```
SubsecondTime SyncClient::barrierWait (
            carbon_barrier_t * barrier )
```

Definition at line 118 of file sync_client.cc.

References SyncServer::barrierWait(), Thread::getCore(), PerformanceModel::getElapsedTime(), Thread::getId(), Core::getPerformanceModel(), m_server, SyncInstruction::PTHREAD_BARRIER, PerformanceModel::queuePseudoInstruction(), Thread::reschedule(), and SubsecondTime::Zero().

Referenced by CarbonBarrierWait().

6.354.3.4 condBroadcast()

```
SubsecondTime SyncClient::condBroadcast (
            carbon_cond_t * cond )
```

Definition at line 101 of file sync_client.cc.

References SyncServer::condBroadcast(), Thread::getCore(), PerformanceModel::getElapsedTime(), Thread::getId(), Core::getPerformanceModel(), m_server, and SubsecondTime::Zero().

Referenced by CarbonCondBroadcast().

6.354.3.5 condInit()

```
void SyncClient::condInit (
            carbon_cond_t * cond )
```

Definition at line 68 of file sync_client.cc.

References SyncServer::condInit(), Thread::getId(), and m_server.

Referenced by CarbonCondInit().

6.354.3.6 condSignal()

```
SubsecondTime SyncClient::condSignal (
            carbon_cond_t * cond )
```

Definition at line 91 of file sync_client.cc.

References SyncServer::condSignal(), Thread::getCore(), PerformanceModel::getElapsedTime(), Thread::getId(), Core::getPerformanceModel(), m_server, and SubsecondTime::Zero().

Referenced by CarbonCondSignal().

6.354.3.7 condWait()

```
SubsecondTime SyncClient::condWait (
            carbon_cond_t * cond,
            carbon_mutex_t * mux )
```

Definition at line 75 of file sync_client.cc.

References SyncServer::condWait(), Thread::getCore(), PerformanceModel::getElapsedTime(), Thread::getId(), Core::getPerformanceModel(), m_server, SyncInstruction::PTHREAD_COND, PerformanceModel::queuePseudoInstruction(), Thread::reschedule(), and SubsecondTime::Zero().

Referenced by CarbonCondWait().

6.354.3.8 mutexInit()

```
void SyncClient::mutexInit (
              carbon_mutex_t * mux )
```

Definition at line 22 of file sync_client.cc.

References Thread::getId(), m_server, and SyncServer::mutexInit().

Referenced by CarbonMutexInit().

6.354.3.9 mutexLock()

```
SubsecondTime SyncClient::mutexLock (
              carbon_mutex_t * mux,
              SubsecondTime delay = SubsecondTime::Zero() )
```

Definition at line 29 of file sync_client.cc.

References __mutexLock().

Referenced by CarbonMutexLock().

6.354.3.10 mutexTrylock()

```
std::pair< SubsecondTime, bool > SyncClient::mutexTrylock (
              carbon_mutex_t * mux )
```

Definition at line 34 of file sync_client.cc.

References __mutexLock(), and SubsecondTime::Zero().

Referenced by CarbonMutexTrylock().

6.354.3.11 mutexUnlock()

```
SubsecondTime SyncClient::mutexUnlock (
              carbon_mutex_t * mux,
              SubsecondTime delay = SubsecondTime::Zero() )
```

Definition at line 55 of file sync_client.cc.

References Thread::getCore(), PerformanceModel::getElapsedTime(), Thread::getId(), Core::getPerformance↩Model(), m_server, SyncServer::mutexUnlock(), SyncInstruction::PTHREAD_MUTEX, PerformanceModel↩::queuePseudoInstruction(), and SubsecondTime::Zero().

Referenced by CarbonMutexUnlock().

6.354.4 Member Data Documentation

6.354.4.1 m_server

```
SyncServer* SyncClient::m_server  [private]
```

Definition at line 32 of file sync_client.h.

Referenced by __mutexLock(), barrierInit(), barrierWait(), condBroadcast(), condInit(), condSignal(), condWait(), mutexInit(), and mutexUnlock().

6.354.4.2 m_thread

```
Thread* SyncClient::m_thread  [private]
```

Definition at line 31 of file sync_client.h.

The documentation for this class was generated from the following files:

- common/system/ **sync_client.h**
- common/system/ **sync_client.cc**

6.355 SyncInstruction Class Reference

```
#include <instruction.h>
```

Inheritance diagram for SyncInstruction:

Public Types

- enum **sync_type_t** {
  FUTEX, **PTHREAD_MUTEX**, **PTHREAD_COND**, **PTHREAD_BARRIER**,
  JOIN, **PAUSE**, **SLEEP**, **SYSCALL**,
  UNSCHEDULED, **NUM_TYPES** }

Public Member Functions

- **SyncInstruction** (**SubsecondTime** time, **sync_type_t** sync_type)
- **SubsecondTime getCost** (**Core** ∗core) const
- **SubsecondTime getTime** () const
- **sync_type_t getSyncType** () const

Private Attributes

- **SubsecondTime m_time**
- **sync_type_t m_sync_type**

Additional Inherited Members

6.355.1 Detailed Description

Definition at line 145 of file instruction.h.

6.355.2 Member Enumeration Documentation

6.355.2.1 sync_type_t

enum **SyncInstruction::sync_type_t**

Enumerator

FUTEX	
PTHREAD_MUTEX	
PTHREAD_COND	
PTHREAD_BARRIER	
JOIN	
PAUSE	
SLEEP	
SYSCALL	
UNSCHEDULED	
NUM_TYPES	

Definition at line 148 of file instruction.h.

6.355.3 Constructor & Destructor Documentation

6.355.3.1 SyncInstruction()

```
SyncInstruction::SyncInstruction (
              SubsecondTime time,
              sync_type_t sync_type )
```

Definition at line 79 of file instruction.cc.

6.355.4 Member Function Documentation

6.355.4.1 getCost()

```
SubsecondTime SyncInstruction::getCost (
              Core * core ) const  [virtual]
```

Reimplemented from **PseudoInstruction** (p. 948).

Definition at line 85 of file instruction.cc.

References LOG_ASSERT_ERROR, and SubsecondTime::Zero().

6.355.4.2 getSyncType()

```
sync_type_t SyncInstruction::getSyncType ( ) const  [inline]
```

Definition at line 164 of file instruction.h.

References m_sync_type.

Referenced by PerformanceModel::handleIdleInstruction().

6.355.4.3 getTime()

```
SubsecondTime SyncInstruction::getTime ( ) const  [inline]
```

Definition at line 163 of file instruction.h.

References m_time.

Referenced by PerformanceModel::handleIdleInstruction().

6.355.5 Member Data Documentation

6.355.5.1 m_sync_type

```
sync_type_t SyncInstruction::m_sync_type [private]
```

Definition at line 168 of file instruction.h.

Referenced by getSyncType().

6.355.5.2 m_time

```
SubsecondTime SyncInstruction::m_time [private]
```

Definition at line 167 of file instruction.h.

Referenced by getTime().

The documentation for this class was generated from the following files:

- common/performance_model/ **instruction.h**
- common/performance_model/ **instruction.cc**

6.356 SyncServer Class Reference

```
#include <sync_server.h>
```

Public Member Functions

- **SyncServer** ()
- ∼**SyncServer** ()
- void **mutexInit** (**thread_id_t** thread_id, **carbon_mutex_t** ∗mux)
- std::pair< **SubsecondTime**, bool > **mutexLock** (**thread_id_t** thread_id, **carbon_mutex_t** ∗mux, bool tryLock, **SubsecondTime** time)
- **SubsecondTime** **mutexUnlock** (**thread_id_t** thread_id, **carbon_mutex_t** ∗mux, **SubsecondTime** time)
- void **condInit** (**thread_id_t** thread_id, **carbon_cond_t** ∗cond)
- **SubsecondTime** **condWait** (**thread_id_t** thread_id, **carbon_cond_t** ∗cond, **carbon_mutex_t** ∗mux, **SubsecondTime** time)
- **SubsecondTime** **condSignal** (**thread_id_t** thread_id, **carbon_cond_t** ∗cond, **SubsecondTime** time)
- **SubsecondTime** **condBroadcast** (**thread_id_t** thread_id, **carbon_cond_t** ∗cond, **SubsecondTime** time)
- void **barrierInit** (**thread_id_t** thread_id, **carbon_barrier_t** ∗barrier, **UInt32** count)
- **SubsecondTime** **barrierWait** (**thread_id_t** thread_id, **carbon_barrier_t** ∗barrier, **SubsecondTime** time)

Private Types

- typedef std::unordered_map< **carbon_mutex_t** ∗, **SimMutex** > **MutexVector**
- typedef std::unordered_map< **carbon_cond_t** ∗, **SimCond** > **CondVector**
- typedef std::vector< **SimBarrier** > **BarrierVector**

Private Member Functions

- **SimMutex** * **getMutex** (**carbon_mutex_t** *mux, bool canCreate=true)
- **SimCond** * **getCond** (**carbon_cond_t** *cond, bool canCreate=true)

Private Attributes

- **MutexVector m_mutexes**
- **CondVector m_conds**
- **BarrierVector m_barriers**
- **SubsecondTime m_reschedule_cost**

6.356.1 Detailed Description

Definition at line 84 of file sync_server.h.

6.356.2 Member Typedef Documentation

6.356.2.1 BarrierVector

```
typedef std::vector< SimBarrier>  SyncServer::BarrierVector  [private]
```

Definition at line 88 of file sync_server.h.

6.356.2.2 CondVector

```
typedef std::unordered_map< carbon_cond_t *,  SimCond>  SyncServer::CondVector  [private]
```

Definition at line 87 of file sync_server.h.

6.356.2.3 MutexVector

```
typedef std::unordered_map< carbon_mutex_t *,  SimMutex>  SyncServer::MutexVector  [private]
```

Definition at line 86 of file sync_server.h.

6.356.3 Constructor & Destructor Documentation

6.356.3.1 SyncServer()

```
SyncServer::SyncServer ( )
```

Definition at line 185 of file sync_server.cc.

References m_reschedule_cost, and SubsecondTime::NS().

6.356.3.2 ~SyncServer()

```
SyncServer::~SyncServer ( )
```

Definition at line 190 of file sync_server.cc.

6.356.4 Member Function Documentation

6.356.4.1 barrierInit()

```
void SyncServer::barrierInit (
              thread_id_t thread_id,
              carbon_barrier_t * barrier,
              UInt32 count )
```

Definition at line 299 of file sync_server.cc.

References m_barriers.

Referenced by SyncClient::barrierInit().

6.356.4.2 barrierWait()

```
 SubsecondTime SyncServer::barrierWait (
              thread_id_t thread_id,
              carbon_barrier_t * barrier,
              SubsecondTime time )
```

Definition at line 307 of file sync_server.cc.

References m_barriers, and SimBarrier::wait().

Referenced by SyncClient::barrierWait().

6.356.4.3 condBroadcast()

```
SubsecondTime SyncServer::condBroadcast (
            thread_id_t thread_id,
            carbon_cond_t * cond,
            SubsecondTime time )
```

Definition at line 289 of file sync_server.cc.

References SimCond::broadcast(), and getCond().

Referenced by SyncClient::condBroadcast().

6.356.4.4 condInit()

```
void SyncServer::condInit (
            thread_id_t thread_id,
            carbon_cond_t * cond )
```

Definition at line 264 of file sync_server.cc.

References getCond().

Referenced by SyncClient::condInit().

6.356.4.5 condSignal()

```
SubsecondTime SyncServer::condSignal (
            thread_id_t thread_id,
            carbon_cond_t * cond,
            SubsecondTime time )
```

Definition at line 279 of file sync_server.cc.

References getCond(), and SimCond::signal().

Referenced by SyncClient::condSignal().

6.356.4.6 condWait()

```
SubsecondTime SyncServer::condWait (
            thread_id_t thread_id,
            carbon_cond_t * cond,
            carbon_mutex_t * mux,
            SubsecondTime time )
```

Definition at line 270 of file sync_server.cc.

References getCond(), getMutex(), and SimCond::wait().

Referenced by SyncClient::condWait().

6.356.4.7 getCond()

```
SimCond * SyncServer::getCond (
              carbon_cond_t * cond,
             bool canCreate = true )  [private]
```

Definition at line 246 of file sync_server.cc.

References LOG_ASSERT_ERROR, and m_conds.

Referenced by condBroadcast(), condInit(), condSignal(), and condWait().

6.356.4.8 getMutex()

```
SimMutex * SyncServer::getMutex (
              carbon_mutex_t * mux,
             bool canCreate = true )  [private]
```

Definition at line 193 of file sync_server.cc.

References LOG_ASSERT_ERROR, and m_mutexes.

Referenced by condWait(), mutexInit(), mutexLock(), and mutexUnlock().

6.356.4.9 mutexInit()

```
void SyncServer::mutexInit (
              thread_id_t thread_id,
              carbon_mutex_t * mux )
```

Definition at line 211 of file sync_server.cc.

References getMutex().

Referenced by SyncClient::mutexInit().

6.356.4.10 mutexLock()

```
std::pair<  SubsecondTime, bool > SyncServer::mutexLock (
              thread_id_t thread_id,
              carbon_mutex_t * mux,
             bool tryLock,
              SubsecondTime time )
```

Definition at line 217 of file sync_server.cc.

References getMutex(), SimMutex::isLocked(), SimMutex::lock(), and m_reschedule_cost.

Referenced by SyncClient::__mutexLock().

6.356.4.11 mutexUnlock()

```
SubsecondTime SyncServer::mutexUnlock (
            thread_id_t thread_id,
            carbon_mutex_t * mux,
            SubsecondTime time )
```

Definition at line 234 of file sync_server.cc.

References getMutex(), m_reschedule_cost, SimMutex::NO_OWNER, SimMutex::unlock(), and SubsecondTime↩ ::Zero().

Referenced by SyncClient::mutexUnlock().

6.356.5 Member Data Documentation

6.356.5.1 m_barriers

```
BarrierVector SyncServer::m_barriers [private]
```

Definition at line 92 of file sync_server.h.

Referenced by barrierInit(), and barrierWait().

6.356.5.2 m_conds

```
CondVector SyncServer::m_conds [private]
```

Definition at line 91 of file sync_server.h.

Referenced by getCond().

6.356.5.3 m_mutexes

```
MutexVector SyncServer::m_mutexes [private]
```

Definition at line 90 of file sync_server.h.

Referenced by getMutex().

6.356.5.4 m_reschedule_cost

```
SubsecondTime SyncServer::m_reschedule_cost  [private]
```

Definition at line 111 of file sync_server.h.

Referenced by mutexLock(), mutexUnlock(), and SyncServer().

The documentation for this class was generated from the following files:

- common/system/ **sync_server.h**
- common/system/ **sync_server.cc**

6.357 SyscallMdl::syscall_args_t Struct Reference

```
#include <syscall_model.h>
```

Public Attributes

- **IntPtr arg0**
- **IntPtr arg1**
- **IntPtr arg2**
- **IntPtr arg3**
- **IntPtr arg4**
- **IntPtr arg5**

6.357.1 Detailed Description

Definition at line 14 of file syscall_model.h.

6.357.2 Member Data Documentation

6.357.2.1 arg0

```
IntPtr SyscallMdl::syscall_args_t::arg0
```

Definition at line 16 of file syscall_model.h.

Referenced by SyscallMdl::formatSyscall(), SyscallMdl::handleFutexCall(), lite::handleSyscall(), hookCallback↩SyscallEnter(), and SyscallMdl::runEnter().

6.357.2.2 arg1

IntPtr SyscallMdl::syscall_args_t::arg1

Definition at line 17 of file syscall_model.h.

Referenced by SyscallMdl::formatSyscall(), SyscallMdl::handleFutexCall(), lite::handleSyscall(), hookCallback↩SyscallEnter(), and SyscallMdl::runEnter().

6.357.2.3 arg2

IntPtr SyscallMdl::syscall_args_t::arg2

Definition at line 18 of file syscall_model.h.

Referenced by SyscallMdl::formatSyscall(), SyscallMdl::handleFutexCall(), lite::handleSyscall(), hookCallback↩SyscallEnter(), and SyscallMdl::runEnter().

6.357.2.4 arg3

IntPtr SyscallMdl::syscall_args_t::arg3

Definition at line 19 of file syscall_model.h.

Referenced by SyscallMdl::formatSyscall(), SyscallMdl::handleFutexCall(), lite::handleSyscall(), and hook↩CallbackSyscallEnter().

6.357.2.5 arg4

IntPtr SyscallMdl::syscall_args_t::arg4

Definition at line 20 of file syscall_model.h.

Referenced by SyscallMdl::formatSyscall(), SyscallMdl::handleFutexCall(), lite::handleSyscall(), and hook↩CallbackSyscallEnter().

6.357.2.6 arg5

IntPtr SyscallMdl::syscall_args_t::arg5

Definition at line 21 of file syscall_model.h.

Referenced by SyscallMdl::formatSyscall(), SyscallMdl::handleFutexCall(), lite::handleSyscall(), and hook↩CallbackSyscallEnter().

The documentation for this struct was generated from the following file:

- common/core/ **syscall_model.h**

6.358 SyscallMdl Class Reference

`#include <syscall_model.h>`

Classes

- struct **futex_counters_t**
- struct **HookSyscallEnter**
- struct **HookSyscallExit**
- struct **syscall_args_t**

Public Member Functions

- **SyscallMdl** (**Thread** *thread)
- ~**SyscallMdl** ()
- bool **runEnter** (**IntPtr** syscall_number, **syscall_args_t** &args)
- **IntPtr** **runExit** (**IntPtr** old_return)
- bool **isEmulated** () const
- bool **inSyscall** () const
- **IntPtr** **getCurrentSyscallNumber** () const
- const **syscall_args_t** **getCurrentSyscallArguments** () const
- String **formatSyscall** () const

Private Member Functions

- **IntPtr** **handleFutexCall** (**syscall_args_t** &args)
- void **futexCount** (uint32_t function, **SubsecondTime** delay)

Private Attributes

- struct **SyscallMdl::futex_counters_t** * **futex_counters**
- **Thread** * **m_thread**
- **IntPtr** **m_syscall_number**
- bool **m_emulated**
- bool **m_stalled**
- **IntPtr** **m_ret_val**
- bool **m_in_syscall**
- **syscall_args_t** **m_syscall_args**
- **UInt64** **m_stdout_bytes**
- **UInt64** **m_stderr_bytes**

Static Private Attributes

- static const char * **futex_names** [ ]

6.358.1 Detailed Description

Definition at line 11 of file syscall_model.h.

6.358.2 Constructor & Destructor Documentation

6.358.2.1 SyscallMdl()

```
SyscallMdl::SyscallMdl (
            Thread * thread )
```

Definition at line 34 of file syscall_model.cc.

References __attribute__, SyscallMdl::futex_counters_t::count, SyscallMdl::futex_counters_t::delay, futex_↩ counters, futex_names, Thread::getId(), LOG_ASSERT_ERROR, m_stderr_bytes, m_stdout_bytes, and register↩ StatsMetric().

6.358.2.2 ~SyscallMdl()

```
SyscallMdl::~SyscallMdl ( )
```

Definition at line 58 of file syscall_model.cc.

References futex_counters.

6.358.3 Member Function Documentation

6.358.3.1 formatSyscall()

```
String SyscallMdl::formatSyscall ( ) const
```

Definition at line 396 of file syscall_model.cc.

References SyscallMdl::syscall_args_t::arg0, SyscallMdl::syscall_args_t::arg1, SyscallMdl::syscall_args_t::arg2, SyscallMdl::syscall_args_t::arg3, SyscallMdl::syscall_args_t::arg4, SyscallMdl::syscall_args_t::arg5, itostr(), m_↩ syscall_args, m_syscall_number, and syscall_string().

Referenced by RoutineTracerOndemand::RtnThread::printStack().

6.358.3.2 futexCount()

```
void SyscallMdl::futexCount (
            uint32_t function,
             SubsecondTime delay )  [private]
```

Definition at line 351 of file syscall_model.cc.

References SyscallMdl::futex_counters_t::count, SyscallMdl::futex_counters_t::delay, and futex_counters.

Referenced by handleFutexCall().

6.358.3.3 getCurrentSyscallArguments()

```
const  syscall_args_t SyscallMdl::getCurrentSyscallArguments ( ) const  [inline]
```

Definition at line 50 of file syscall_model.h.

References m_syscall_args.

6.358.3.4 getCurrentSyscallNumber()

```
IntPtr SyscallMdl::getCurrentSyscallNumber ( ) const  [inline]
```

Definition at line 49 of file syscall_model.h.

References m_syscall_number.

Referenced by lite::syscallExitRunModel().

6.358.3.5 handleFutexCall()

```
IntPtr SyscallMdl::handleFutexCall (
             syscall_args_t & args )  [private]
```

Definition at line 357 of file syscall_model.cc.

References SyscallMdl::syscall_args_t::arg0, SyscallMdl::syscall_args_t::arg1, SyscallMdl::syscall_args_t::arg2, SyscallMdl::syscall_args_t::arg3, SyscallMdl::syscall_args_t::arg4, SyscallMdl::syscall_args_t::arg5, SyncInstruction::FUTEX, FUTEX_CMD_MASK, FUTEX_PRIVATE_FLAG, futexCount(), Thread::getCore(), PerformanceModel::getElapsedTime(), Thread::getId(), Core::getPerformanceModel(), LOG_ASSERT_ERROR, m_thread, SyscallServer::futex_args_t::op, PerformanceModel::queuePseudoInstruction(), Thread::reschedule(), PthreadEmu::STATE_RUNNING, PthreadEmu::STATE_WAITING, SyscallServer::futex_args_t::timeout, SyscallServer::futex_args_t::uaddr, SyscallServer::futex_args_t::uaddr2, PthreadEmu::updateState(), SyscallServer::futex_args_t::val, SyscallServer::futex_args_t::val2, and SyscallServer::futex_args_t::val3.

Referenced by runEnter().

6.358.3.6 inSyscall()

```
bool SyscallMdl::inSyscall ( ) const  [inline]
```

Definition at line 48 of file syscall_model.h.

References m_in_syscall.

Referenced by RoutineTracerOndemand::RtnThread::printStack().

6.358.3.7 isEmulated()

```
bool SyscallMdl::isEmulated ( ) const  [inline]
```

Definition at line 47 of file syscall_model.h.

References m_emulated.

Referenced by lite::syscallEnterRunModel(), and lite::syscallExitRunModel().

6.358.3.8 runEnter()

```
bool SyscallMdl::runEnter (
              IntPtr syscall_number,
              syscall_args_t & args )
```

Definition at line 63 of file syscall_model.cc.

References Core::accessMemory(), SyscallMdl::syscall_args_t::arg0, SyscallMdl::syscall_args_t::arg1, SyscallMdl::syscall_args_t::arg2, SyscallMdl::HookSyscallEnter::args, CLOCK_MONOTONIC_COARSE, CLOCK_MONOTONIC_RAW, CLOG, SyscallMdl::HookSyscallEnter::core_id, Thread::getCore(), PerformanceModel::getElapsedTime(), Thread::getId(), Core::getId(), SubsecondTime::getNS(), Core::getPerformanceModel(), handleFutexCall(), HookType::HOOK_SYSCALL_ENTER, LOG_ASSERT_ERROR, LOG_PRINT, m_emulated, m_in_syscall, m_ret_val, m_stalled, m_stderr_bytes, m_stdout_bytes, m_syscall_args, m_syscall_number, m_thread, Core::NONE, SubsecondTime::NS(), PerformanceModel::queuePseudoInstruction(), Core::READ, Thread::reschedule(), SubsecondTime::SEC(), SyncInstruction::SLEEP, ThreadManager::STALL_PAUSE, ThreadManager::STALL_SYSCALL, SyscallMdl::HookSyscallEnter::syscall_number, SyscallMdl::HookSyscallEnter::thread_id, SyscallMdl::HookSyscallEnter::time, SyncInstruction::UNSCHEDULED, and Core::WRITE.

Referenced by lite::handleSyscall(), and TraceThread::handleSyscallFunc().

6.358.3.9 runExit()

```
IntPtr SyscallMdl::runExit (
            IntPtr old_return )
```

Definition at line 304 of file syscall_model.cc.

References CLOG, SyscallMdl::HookSyscallExit::core_id, SyscallMdl::HookSyscallExit::emulated, Thread::getCore(), PerformanceModel::getElapsedTime(), Thread::getId(), Core::getId(), Core::getPerformanceModel(), HookType::HOOK_SYSCALL_EXIT, INVALID_THREAD_ID, m_emulated, m_in_syscall, m_ret_val, m_stalled, m_syscall_number, m_thread, SyncInstruction::PAUSE, PerformanceModel::queuePseudoInstruction(), Thread::reschedule(), SyscallMdl::HookSyscallExit::ret_val, SyncInstruction::SYSCALL, SyscallMdl::HookSyscallExit::thread_id, and SyscallMdl::HookSyscallExit::time.

Referenced by TraceThread::handleSyscallFunc(), lite::syscallExitRunModel(), and TraceThread::unblock().

6.358.4 Member Data Documentation

6.358.4.1 futex_counters

```
struct SyscallMdl::futex_counters_t * SyscallMdl::futex_counters [private]
```

Referenced by futexCount(), SyscallMdl(), and ~SyscallMdl().

6.358.4.2 futex_names

```
const char * SyscallMdl::futex_names [static], [private]
```

Initial value:

```
=
{
   "FUTEX_WAIT", "FUTEX_WAKE", "FUTEX_FD", "FUTEX_REQUEUE",
   "FUTEX_CMP_REQUEUE", "FUTEX_WAKE_OP", "FUTEX_LOCK_PI", "FUTEX_UNLOCK_PI",
   "FUTEX_TRYLOCK_PI", "FUTEX_WAIT_BITSET", "FUTEX_WAKE_BITSET", "FUTEX_WAIT_REQUEUE_PI",
   "FUTEX_CMP_REQUEUE_PI"
}
```

Definition at line 54 of file syscall_model.h.

Referenced by SyscallMdl().

6.358.4.3 m_emulated

```
bool SyscallMdl::m_emulated [private]
```

Definition at line 64 of file syscall_model.h.

Referenced by isEmulated(), runEnter(), and runExit().

6.358.4.4 m_in_syscall

```
bool SyscallMdl::m_in_syscall  [private]
```

Definition at line 67 of file syscall_model.h.

Referenced by inSyscall(), runEnter(), and runExit().

6.358.4.5 m_ret_val

```
IntPtr SyscallMdl::m_ret_val  [private]
```

Definition at line 66 of file syscall_model.h.

Referenced by runEnter(), and runExit().

6.358.4.6 m_stalled

```
bool SyscallMdl::m_stalled  [private]
```

Definition at line 65 of file syscall_model.h.

Referenced by runEnter(), and runExit().

6.358.4.7 m_stderr_bytes

```
UInt64 SyscallMdl::m_stderr_bytes  [private]
```

Definition at line 71 of file syscall_model.h.

Referenced by runEnter(), and SyscallMdl().

6.358.4.8 m_stdout_bytes

```
UInt64 SyscallMdl::m_stdout_bytes  [private]
```

Definition at line 70 of file syscall_model.h.

Referenced by runEnter(), and SyscallMdl().

6.358.4.9 m_syscall_args

```
syscall_args_t SyscallMdl::m_syscall_args [private]
```

Definition at line 68 of file syscall_model.h.

Referenced by formatSyscall(), getCurrentSyscallArguments(), and runEnter().

6.358.4.10 m_syscall_number

```
IntPtr SyscallMdl::m_syscall_number [private]
```

Definition at line 63 of file syscall_model.h.

Referenced by formatSyscall(), getCurrentSyscallNumber(), runEnter(), and runExit().

6.358.4.11 m_thread

```
Thread* SyscallMdl::m_thread [private]
```

Definition at line 62 of file syscall_model.h.

Referenced by handleFutexCall(), runEnter(), and runExit().

The documentation for this class was generated from the following files:

- common/core/ **syscall_model.h**
- common/core/ **syscall_model.cc**

6.359 SyscallServer Class Reference

`#include <syscall_server.h>`

Classes

- struct **futex_args_t**

Public Member Functions

- **SyscallServer** ()
- ~**SyscallServer** ()
- void **handleSleepCall** (**thread_id_t** thread_id, **SubsecondTime** wake_time, **SubsecondTime** curr_time, **SubsecondTime** &end_time)
- **IntPtr handleFutexCall** (**thread_id_t** thread_id, **futex_args_t** &args, **SubsecondTime** curr_time, **SubsecondTime** &end_time)
- **SubsecondTime getNextTimeout** (**SubsecondTime** time)

Private Types

- typedef std::unordered_map< **IntPtr**, **SimFutex** > **FutexMap**

Private Member Functions

- **IntPtr** **futexWait** (**thread_id_t** thread_id, int *uaddr, int val, int act_val, int val3, **SubsecondTime** curr_time, **SubsecondTime** timeout_time, **SubsecondTime** &end_time)
- **IntPtr** **futexWake** (**thread_id_t** thread_id, int *uaddr, int nr_wake, int val3, **SubsecondTime** curr_time, **SubsecondTime** &end_time)
- **IntPtr** **futexWakeOp** (**thread_id_t** thread_id, int *uaddr, int *uaddr2, int nr_wake, int nr_wake2, int op, **SubsecondTime** curr_time, **SubsecondTime** &end_time)
- **IntPtr** **futexCmpRequeue** (**thread_id_t** thread_id, int *uaddr, int val, int *uaddr2, int val3, int act_val, **SubsecondTime** curr_time, **SubsecondTime** &end_time)
- **SimFutex** * **findFutexByUaddr** (int *uaddr, **thread_id_t** thread_id)
- **thread_id_t** **wakeFutexOne** (**SimFutex** *sim_futex, **thread_id_t** thread_by, int mask, **SubsecondTime** curr_time)
- int **futexDoOp** (**Core** *core, int op, int *uaddr)
- void **futexPeriodic** (**SubsecondTime** time)
- **SubsecondTime** **applyRescheduleCost** (**thread_id_t** thread_id, bool conditional=true)

Static Private Member Functions

- static **SInt64** **hook_periodic** (**UInt64** ptr, **UInt64** time)

Private Attributes

- **SubsecondTime** **m_reschedule_cost**
- **SimFutex::ThreadQueue** **m_sleeping**
- **FutexMap** **m_futexes**

Friends

- class **ThreadManager**

6.359.1 Detailed Description

Definition at line 46 of file syscall_server.h.

6.359.2 Member Typedef Documentation

6.359.2.1 FutexMap

typedef std::unordered_map< **IntPtr**, **SimFutex**> **SyscallServer::FutexMap** [private]

Definition at line 93 of file syscall_server.h.

6.359.3 Constructor & Destructor Documentation

6.359.3.1 SyscallServer()

```
SyscallServer::SyscallServer ( )
```

Definition at line 16 of file syscall_server.cc.

References HookType::HOOK_PERIODIC, hook_periodic(), m_reschedule_cost, and SubsecondTime::NS().

6.359.3.2 ~SyscallServer()

```
SyscallServer::~SyscallServer ( )
```

Definition at line 23 of file syscall_server.cc.

6.359.4 Member Function Documentation

6.359.4.1 applyRescheduleCost()

```
SubsecondTime SyscallServer::applyRescheduleCost (
            thread_id_t thread_id,
            bool conditional = true )  [private]
```

Definition at line 109 of file syscall_server.cc.

References m_reschedule_cost, and SubsecondTime::Zero().

Referenced by futexWake(), futexWakeOp(), and wakeFutexOne().

6.359.4.2 findFutexByUaddr()

```
SimFutex * SyscallServer::findFutexByUaddr (
            int * uaddr,
            thread_id_t thread_id )  [private]
```

Definition at line 123 of file syscall_server.cc.

References m_futexes.

Referenced by futexCmpRequeue(), futexWait(), futexWake(), and futexWakeOp().

6.359.4.3 futexCmpRequeue()

```
IntPtr SyscallServer::futexCmpRequeue (
            thread_id_t thread_id,
            int * uaddr,
            int val,
            int * uaddr2,
            int val3,
            int act_val,
            SubsecondTime curr_time,
            SubsecondTime & end_time )  [private]
```

Definition at line 280 of file syscall_server.cc.

References SimFutex::dequeueWaiter(), findFutexByUaddr(), FUTEX_BITSET_MATCH_ANY, INVALID_THREAD_ID, LOG_PRINT, and SimFutex::requeueWaiter().

Referenced by handleFutexCall().

6.359.4.4 futexDoOp()

```
int SyscallServer::futexDoOp (
            Core * core,
            int op,
            int * uaddr )  [private]
```

Definition at line 176 of file syscall_server.cc.

References Core::accessMemory(), Core::LOCK, LOG_ASSERT_ERROR, LOG_PRINT_WARNING_ONCE, Core::READ_EX, Core::UNLOCK, and Core::WRITE.

Referenced by futexWakeOp().

6.359.4.5 futexPeriodic()

```
void SyscallServer::futexPeriodic (
            SubsecondTime time )  [private]
```

Definition at line 320 of file syscall_server.cc.

References m_futexes, and m_sleeping.

6.359.4.6 futexWait()

```
IntPtr SyscallServer::futexWait (
             thread_id_t thread_id,
            int * uaddr,
            int val,
            int act_val,
            int val3,
             SubsecondTime curr_time,
             SubsecondTime timeout_time,
             SubsecondTime & end_time )  [private]
```

Definition at line 131 of file syscall_server.cc.

References SimFutex::enqueueWaiter(), findFutexByUaddr(), and LOG_PRINT.

Referenced by handleFutexCall().

6.359.4.7 futexWake()

```
IntPtr SyscallServer::futexWake (
             thread_id_t thread_id,
            int * uaddr,
            int nr_wake,
            int val3,
             SubsecondTime curr_time,
             SubsecondTime & end_time )  [private]
```

Definition at line 157 of file syscall_server.cc.

References applyRescheduleCost(), findFutexByUaddr(), INVALID_THREAD_ID, LOG_PRINT, and wakeFutex↩
One().

Referenced by handleFutexCall().

6.359.4.8 futexWakeOp()

```
IntPtr SyscallServer::futexWakeOp (
             thread_id_t thread_id,
            int * uaddr,
            int * uaddr2,
            int nr_wake,
            int nr_wake2,
            int op,
             SubsecondTime curr_time,
             SubsecondTime & end_time )  [private]
```

Definition at line 244 of file syscall_server.cc.

References applyRescheduleCost(), findFutexByUaddr(), FUTEX_BITSET_MATCH_ANY, futexDoOp(), Thread↩
::getCore(), INVALID_THREAD_ID, LOG_ASSERT_ERROR, LOG_PRINT, and wakeFutexOne().

Referenced by handleFutexCall().

6.359.4.9 getNextTimeout()

```
SubsecondTime SyscallServer::getNextTimeout (
              SubsecondTime time )
```

Definition at line 343 of file syscall_server.cc.

References m_futexes, m_sleeping, and SubsecondTime::MaxTime().

6.359.4.10 handleFutexCall()

```
IntPtr SyscallServer::handleFutexCall (
              thread_id_t thread_id,
              futex_args_t & args,
              SubsecondTime curr_time,
              SubsecondTime & end_time )
```

Definition at line 35 of file syscall_server.cc.

References Core::accessMemory(), CLOG, FUTEX_BITSET_MATCH_ANY, FUTEX_CMD_MASK, FUTEX_PRIVATE_FLAG, FUTEX_WAIT_BITSET, FUTEX_WAKE_BITSET, futexCmpRequeue(), futexWait(), futexWake(), futexWakeOp(), LOG_ASSERT_ERROR, LOG_ASSERT_WARNING_ONCE, LOG_PRINT, LOG_PRINT_ERROR, SubsecondTime::MaxTime(), Core::NONE, SubsecondTime::NS(), SyscallServer::futex_args_t::op, Core::READ, SubsecondTime::SEC(), SyscallServer::futex_args_t::timeout, SyscallServer::futex_args_t::uaddr, SyscallServer::futex_args_t::uaddr2, SyscallServer::futex_args_t::val, SyscallServer::futex_args_t::val2, and SyscallServer::futex_args_t::val3.

6.359.4.11 handleSleepCall()

```
void SyscallServer::handleSleepCall (
              thread_id_t thread_id,
              SubsecondTime wake_time,
              SubsecondTime curr_time,
              SubsecondTime & end_time )
```

Definition at line 27 of file syscall_server.cc.

References m_sleeping, and ThreadManager::STALL_SLEEP.

6.359.4.12 hook_periodic()

```
static  SInt64 SyscallServer::hook_periodic (
              UInt64 ptr,
              UInt64 time )  [inline], [static], [private]
```

Definition at line 81 of file syscall_server.h.

Referenced by SyscallServer().

6.359.4.13 wakeFutexOne()

```
thread_id_t SyscallServer::wakeFutexOne (
             SimFutex * sim_futex,
             thread_id_t thread_by,
            int mask,
             SubsecondTime curr_time )  [private]
```

Definition at line 151 of file syscall_server.cc.

References applyRescheduleCost(), and SimFutex::dequeueWaiter().

Referenced by futexWake(), and futexWakeOp().

6.359.5 Friends And Related Function Documentation

6.359.5.1 ThreadManager

```
friend class  ThreadManager  [friend]
```

Definition at line 96 of file syscall_server.h.

6.359.6 Member Data Documentation

6.359.6.1 m_futexes

```
FutexMap SyscallServer::m_futexes  [private]
```

Definition at line 94 of file syscall_server.h.

Referenced by findFutexByUaddr(), futexPeriodic(), and getNextTimeout().

6.359.6.2 m_reschedule_cost

```
SubsecondTime SyscallServer::m_reschedule_cost  [private]
```

Definition at line 87 of file syscall_server.h.

Referenced by applyRescheduleCost(), and SyscallServer().

6.359.6.3 m_sleeping

```
SimFutex::ThreadQueue SyscallServer::m_sleeping  [private]
```

Definition at line 90 of file syscall_server.h.

Referenced by futexPeriodic(), getNextTimeout(), and handleSleepCall().

The documentation for this class was generated from the following files:

- common/system/ **syscall_server.h**
- common/system/ **syscall_server.cc**

6.360 GhbPrefetcher::TableEntry Struct Reference

Public Member Functions

- **TableEntry** ()

Public Attributes

- **UInt32 ghbIndex**
- **SInt64 delta**
- **UInt32 generation**

6.360.1 Detailed Description

Definition at line 26 of file ghb_prefetcher.h.

6.360.2 Constructor & Destructor Documentation

6.360.2.1 TableEntry()

```
GhbPrefetcher::TableEntry::TableEntry ( )  [inline]
```

Definition at line 31 of file ghb_prefetcher.h.

6.360.3 Member Data Documentation

6.360.3.1 delta

SInt64 GhbPrefetcher::TableEntry::delta

Definition at line 29 of file ghb_prefetcher.h.

6.360.3.2 generation

UInt32 GhbPrefetcher::TableEntry::generation

Definition at line 30 of file ghb_prefetcher.h.

6.360.3.3 ghbIndex

UInt32 GhbPrefetcher::TableEntry::ghbIndex

Definition at line 28 of file ghb_prefetcher.h.

The documentation for this struct was generated from the following file:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **ghb_prefetcher.h**

6.361 Tag Struct Reference

#include <tags.h>

Public Member Functions

- **Tag** (String **tag**, **UInt64 value**)
- bool **operator**< (const **Tag** &rhs) const
- bool **operator==** (const **Tag** &rhs) const

Public Attributes

- String **tag**
- **UInt64 value**

6.361.1 Detailed Description

Definition at line 23 of file tags.h.

6.361.2 Constructor & Destructor Documentation

6.361.2.1 Tag()

```
Tag::Tag (
            String tag,
             UInt64 value )  [inline]
```

Definition at line 27 of file tags.h.

6.361.3 Member Function Documentation

6.361.3.1 operator<()

```
bool Tag::operator< (
            const  Tag & rhs ) const  [inline]
```

Definition at line 29 of file tags.h.

References tag, and value.

6.361.3.2 operator==()

```
bool Tag::operator== (
            const  Tag & rhs ) const  [inline]
```

Definition at line 41 of file tags.h.

References tag, and value.

6.361.4 Member Data Documentation

6.361.4.1 tag

```
String Tag::tag
```

Definition at line 24 of file tags.h.

Referenced by operator<(), and operator==().

6.361.4.2 value

```
UInt64 Tag::value
```

Definition at line 25 of file tags.h.

Referenced by operator<(), and operator==().

The documentation for this struct was generated from the following file:

- common/misc/ **tags.h**

6.362 TagsManager Class Reference

```
#include <tags.h>
```

Public Member Functions

- **TagsManager** (**config::Config** *config)
- void **addTag** (String objname, **UInt64** id, String tag, **UInt64** value=1)
- void **updateTag** (String objname, **UInt64** id, String tag, **UInt64** value=1)
- bool **hasTag** (String objname, **UInt64** id, String tag)
- **UInt64** **getTag** (String objname, **UInt64** id, String tag)
- void **deleteTag** (String objname, **UInt64** id, String tag)

Private Attributes

- std::map< String, std::map< **Tag**, **UInt64** > > **m_tags**

6.362.1 Detailed Description

Definition at line 47 of file tags.h.

6.362.2 Constructor & Destructor Documentation

6.362.2.1 TagsManager()

```
TagsManager::TagsManager (
            config::Config * config )
```

Definition at line 18 of file tags.cc.

References addTag(), and config::Section::getSubsections().

6.362.3 Member Function Documentation

6.362.3.1 addTag()

```
void TagsManager::addTag (
             String objname,
              UInt64 id,
             String tag,
              UInt64 value = 1 )  [inline]
```

Definition at line 52 of file tags.h.

References m_tags.

Referenced by TagsManager(), and updateTag().

6.362.3.2 deleteTag()

```
void TagsManager::deleteTag (
             String objname,
              UInt64 id,
             String tag )  [inline]
```

Definition at line 72 of file tags.h.

References m_tags.

6.362.3.3 getTag()

```
 UInt64 TagsManager::getTag (
             String objname,
              UInt64 id,
             String tag )  [inline]
```

Definition at line 67 of file tags.h.

References m_tags.

6.362.3.4 hasTag()

```
bool TagsManager::hasTag (
             String objname,
              UInt64 id,
             String tag )  [inline]
```

Definition at line 62 of file tags.h.

References m_tags.

6.362.3.5 updateTag()

```
void TagsManager::updateTag (
             String objname,
              UInt64 id,
             String tag,
              UInt64 value = 1 )  [inline]
```

Definition at line 57 of file tags.h.

References addTag().

6.362.4 Member Data Documentation

6.362.4.1 m_tags

```
std::map<String, std::map< Tag, UInt64> > TagsManager::m_tags  [private]
```

Definition at line 79 of file tags.h.

Referenced by addTag(), deleteTag(), getTag(), and hasTag().

The documentation for this class was generated from the following files:

- common/misc/ **tags.h**
- common/misc/ **tags.cc**

6.363 TFixedPoint< one > Class Template Reference

```
#include <fixed_point.h>
```

Public Member Functions

- **TFixedPoint** ()
- **TFixedPoint** (**SInt64** i)
- void **set_raw** (**SInt64** value)
- void **set_int** (**SInt64** value)
- bool **operator==** (const **TFixedPoint** &fp) const
- bool **operator==** (**SInt64** i) const
- **TFixedPoint** **operator+** (const **TFixedPoint** &fp) const
- **TFixedPoint** **operator+** (**SInt64** i) const
- **TFixedPoint** **operator-** (const **TFixedPoint** &fp) const
- **TFixedPoint** **operator-** (**SInt64** i) const
- **TFixedPoint** **operator*** (const **TFixedPoint** &fp) const
- **TFixedPoint** **operator*** (**SInt64** i) const
- **TFixedPoint** **operator/** (const **TFixedPoint** &fp) const
- **TFixedPoint** **operator/** (**SInt64** i) const

Static Public Member Functions

- static **TFixedPoint** **from_raw** (**SInt64** value)
- static **SInt64** **floor** (const **TFixedPoint** fp)

Private Attributes

- **SInt64** **m_value**

Static Private Attributes

- static const **SInt64** **m_one** = one

6.363.1 Detailed Description

template<SInt64 one>
class TFixedPoint< one >

Definition at line 13 of file fixed_point.h.

6.363.2 Constructor & Destructor Documentation

6.363.2.1 TFixedPoint() [1/2]

```
template<SInt64 one>
TFixedPoint< one >:: TFixedPoint ( ) [inline]
```

Definition at line 20 of file fixed_point.h.

6.363.2.2 TFixedPoint() [2/2]

```
template<SInt64 one>
TFixedPoint< one >::  TFixedPoint (
              SInt64 i )  [inline]
```

Definition at line 21 of file fixed_point.h.

6.363.3 Member Function Documentation

6.363.3.1 floor()

```
template<SInt64 one>
static  SInt64  TFixedPoint< one >::floor (
             const  TFixedPoint< one > fp )  [inline], [static]
```

Definition at line 42 of file fixed_point.h.

Referenced by Timer::getTime().

6.363.3.2 from_raw()

```
template<SInt64 one>
static  TFixedPoint  TFixedPoint< one >::from_raw (
              SInt64 value )  [inline], [static]
```

Definition at line 25 of file fixed_point.h.

Referenced by TFixedPoint< __UINT64_C(0x4000)>::operator*(), TFixedPoint< __UINT64_C(0x4000)>↩ ::operator+(), TFixedPoint< __UINT64_C(0x4000)>::operator-(), and TFixedPoint< __UINT64_C(0x4000)>↩ ::operator/().

6.363.3.3 operator*() [1/2]

```
template<SInt64 one>
TFixedPoint  TFixedPoint< one >::operator* (
             const  TFixedPoint< one > & fp ) const  [inline]
```

Definition at line 36 of file fixed_point.h.

6.363.3.4 operator∗() [2/2]

```
template<SInt64 one>
TFixedPoint  TFixedPoint< one >::operator* (
             SInt64 i ) const  [inline]
```

Definition at line 37 of file fixed_point.h.

6.363.3.5 operator+() [1/2]

```
template<SInt64 one>
TFixedPoint  TFixedPoint< one >::operator+ (
            const  TFixedPoint< one > & fp ) const  [inline]
```

Definition at line 30 of file fixed_point.h.

6.363.3.6 operator+() [2/2]

```
template<SInt64 one>
TFixedPoint  TFixedPoint< one >::operator+ (
             SInt64 i ) const  [inline]
```

Definition at line 31 of file fixed_point.h.

6.363.3.7 operator-() [1/2]

```
template<SInt64 one>
TFixedPoint  TFixedPoint< one >::operator- (
            const  TFixedPoint< one > & fp ) const  [inline]
```

Definition at line 33 of file fixed_point.h.

6.363.3.8 operator-() [2/2]

```
template<SInt64 one>
TFixedPoint  TFixedPoint< one >::operator- (
             SInt64 i ) const  [inline]
```

Definition at line 34 of file fixed_point.h.

6.363.3.9 operator/() [1/2]

```
template<SInt64 one>
TFixedPoint  TFixedPoint< one >::operator/ (
             const  TFixedPoint< one > & fp ) const  [inline]
```

Definition at line 39 of file fixed_point.h.

6.363.3.10 operator/() [2/2]

```
template<SInt64 one>
TFixedPoint  TFixedPoint< one >::operator/ (
              SInt64 i ) const  [inline]
```

Definition at line 40 of file fixed_point.h.

6.363.3.11 operator==() [1/2]

```
template<SInt64 one>
bool  TFixedPoint< one >::operator== (
             const  TFixedPoint< one > & fp ) const  [inline]
```

Definition at line 27 of file fixed_point.h.

6.363.3.12 operator==() [2/2]

```
template<SInt64 one>
bool  TFixedPoint< one >::operator== (
              SInt64 i ) const  [inline]
```

Definition at line 28 of file fixed_point.h.

6.363.3.13 set_int()

```
template<SInt64 one>
void  TFixedPoint< one >::set_int (
              SInt64 value )  [inline]
```

Definition at line 24 of file fixed_point.h.

Referenced by TFixedPoint< __UINT64_C(0x4000)>::TFixedPoint().

6.363.3.14 set_raw()

```
template<SInt64 one>
void  TFixedPoint< one >::set_raw (
              SInt64 value )  [inline]
```

Definition at line 23 of file fixed_point.h.

Referenced by TFixedPoint< __UINT64_C(0x4000)>::from_raw().

6.363.4 Member Data Documentation

6.363.4.1 m_one

```
template<SInt64 one>
const  SInt64  TFixedPoint< one >::m_one = one  [static], [private]
```

Definition at line 17 of file fixed_point.h.

Referenced by TFixedPoint< __UINT64_C(0x4000)>::floor(), TFixedPoint< __UINT64_C(0x4000)>::operator+(), TFixedPoint< __UINT64_C(0x4000)>::operator-(), TFixedPoint< __UINT64_C(0x4000)>::operator/(), TFixed↩Point< __UINT64_C(0x4000)>::operator==(), and TFixedPoint< __UINT64_C(0x4000)>::set_int().

6.363.4.2 m_value

```
template<SInt64 one>
SInt64  TFixedPoint< one >::m_value  [private]
```

Definition at line 16 of file fixed_point.h.

Referenced by TFixedPoint< __UINT64_C(0x4000)>::floor(), TFixedPoint< __UINT64_C(0x4000)>::operator*(), TFixedPoint< __UINT64_C(0x4000)>::operator+(), TFixedPoint< __UINT64_C(0x4000)>::operator-(), TFixed↩Point< __UINT64_C(0x4000)>::operator/(), and TFixedPoint< __UINT64_C(0x4000)>::operator==().

The documentation for this class was generated from the following file:

- common/misc/ **fixed_point.h**

6.364 Thread Class Reference

```
#include <thread.h>
```

Public Types

- typedef **UInt64**(∗ **va2pa_func_t**) (**UInt64** arg, **UInt64** va)

Public Member Functions

- **Thread** (**thread_id_t** thread_id, **app_id_t** app_id)
- ∼**Thread** ()
- **thread_id_t getId** () const
- **app_id_t getAppId** () const
- String **getName** () const
- void **setName** (String name)
- **SyncClient** ∗ **getSyncClient** () const
- **RoutineTracerThread** ∗ **getRoutineTracer** () const
- void **setVa2paFunc** (**va2pa_func_t** va2pa_func, **UInt64 m_va2pa_arg**)
- **UInt64 va2pa** (**UInt64** logical_address) const
- **SubsecondTime wait** (**Lock** &lock)
- void **signal** (**SubsecondTime** time, void ∗msg=NULL)
- **SubsecondTime getWakeupTime** () const
- void ∗ **getWakeupMsg** () const
- **Core** ∗ **getCore** () const
- void **setCore** (**Core** ∗core)
- bool **reschedule** (**SubsecondTime** &time, **Core** ∗current_core)
- bool **updateCoreTLS** (int threadIndex=-1)
- **SyscallMdl** ∗ **getSyscallMdl** ()

Public Attributes

- struct {
  pid_t **tid**
  IntPtr tid_ptr
  bool **clear_tid**
  } **m_os_info**

Private Attributes

- **thread_id_t m_thread_id**
- **app_id_t m_app_id**
- String **m_name**
- **ConditionVariable m_cond**
- **SubsecondTime m_wakeup_time**
- void ∗ **m_wakeup_msg**
- **Core** ∗ **m_core**
- **SyscallMdl** ∗ **m_syscall_model**
- **SyncClient** ∗ **m_sync_client**
- **RoutineTracerThread** ∗ **m_rtn_tracer**
- **va2pa_func_t m_va2pa_func**
- **UInt64 m_va2pa_arg**

6.364.1 Detailed Description

Definition at line 12 of file thread.h.

6.364.2 Member Typedef Documentation

6.364.2.1 va2pa_func_t

```
typedef  UInt64(* Thread::va2pa_func_t) ( UInt64 arg,  UInt64 va)
```

Definition at line 15 of file thread.h.

6.364.3 Constructor & Destructor Documentation

6.364.3.1 Thread()

```
Thread::Thread (
              thread_id_t thread_id,
              app_id_t app_id )
```

Definition at line 12 of file thread.cc.

References getRoutineTracer(), m_os_info, m_rtn_tracer, m_sync_client, and m_syscall_model.

6.364.3.2 ~Thread()

```
Thread::~Thread ( )
```

Definition at line 27 of file thread.cc.

References m_rtn_tracer, m_sync_client, and m_syscall_model.

6.364.4 Member Function Documentation

6.364.4.1 getAppId()

```
app_id_t Thread::getAppId ( ) const  [inline]
```

Definition at line 43 of file thread.h.

References m_app_id.

Referenced by TraceManager::endApplication(), getThreadAppid(), RoutineTracerOndemand::RtnThread::printStack(), TraceManager::signalDone(), and TraceThread::va2pa().

6.364.4.2 getCore()

```
Core* Thread::getCore ( ) const  [inline]
```

Definition at line 75 of file thread.h.

References m_core.

Referenced by SyncClient::__mutexLock(), TraceManager::accessMemory(), SyncClient::barrierWait(), SpinLoopDetector::commitBCT(), SyncClient::condBroadcast(), SyncClient::condSignal(), SyncClient::condWait(), SyscallServer::futexWakeOp(), TraceThread::getCurrentTime(), InstructionModeling::handleBasicBlock(), TraceThread::handleCacheOnlyFunc(), handleCheckScheduled(), TraceThread::handleEmuFunc(), SyscallMdl::handleFutexCall(), InstructionModeling::handleInstruction(), TraceThread::handleInstructionCountFunc(), handleMagic(), TraceThread::handleSyscallFunc(), ThreadManager::joinThread(), ThreadManager::moveThread(), SyncClient::mutexUnlock(), ThreadManager::onThreadExit(), ThreadManager::onThreadStart(), RoutineTracerOndemand::RtnThread::printStack(), TraceThread::run(), SyscallMdl::runEnter(), SyscallMdl::runExit(), ThreadManager::spawnThread(), TraceThread::unblock(), and TraceThread::va2pa().

6.364.4.3 getId()

```
thread_id_t Thread::getId ( ) const  [inline]
```

Definition at line 42 of file thread.h.

References m_thread_id.

Referenced by SyncClient::__mutexLock(), SmtTimer::barrier(), SyncClient::barrierInit(), SyncClient::barrierWait(), RoutineTracerFunctionStats::RtnMaster::ce_get_owner(), SyncClient::condBroadcast(), SyncClient::condInit(), SyncClient::condSignal(), SyncClient::condWait(), ThreadManager::createThread_unlocked(), FastForwardPerformanceManager::disable(), RoutineTracerFunctionStats::RtnMaster::getThreadHandler(), TraceThread::handleForkFunc(), SyscallMdl::handleFutexCall(), TraceThread::handleJoinFunc(), TraceThread::handleMagicFunc(), TraceThread::handleNewThreadFunc(), TraceThread::handleOutputFunc(), BarrierSyncServer::isCoreRunning(), SyncClient::mutexInit(), SyncClient::mutexUnlock(), TraceManager::newThread(), RoutineTracerOndemand::RtnThread::printStack(), reschedule(), TraceThread::run(), SyscallMdl::runEnter(), SyscallMdl::runExit(), setCore(), ThreadManager::spawnThread(), BarrierSyncServer::synchronize(), SyscallMdl::SyscallMdl(), and TraceThread::unblock().

6.364.4.4 getName()

```
String Thread::getName ( ) const  [inline]
```

Definition at line 45 of file thread.h.

References m_name.

Referenced by getThreadName().

6.364.4.5 getRoutineTracer()

```
RoutineTracerThread* Thread::getRoutineTracer ( ) const  [inline]
```

Definition at line 49 of file thread.h.

References m_rtn_tracer.

Referenced by TraceThread::handleRoutineChangeFunc(), RoutineTracerOndemand::RtnMaster::signalHandler(), and Thread().

6.364.4.6 getSyncClient()

```
SyncClient* Thread::getSyncClient ( ) const  [inline]
```

Definition at line 48 of file thread.h.

References m_sync_client.

Referenced by CarbonBarrierInit(), CarbonBarrierWait(), CarbonCondBroadcast(), CarbonCondInit(), CarbonCondSignal(), CarbonCondWait(), CarbonMutexInit(), CarbonMutexLock(), CarbonMutexTrylock(), and CarbonMutexUnlock().

6.364.4.7 getSyscallMdl()

```
SyscallMdl* Thread::getSyscallMdl ( )  [inline]
```

Definition at line 81 of file thread.h.

References m_syscall_model.

Referenced by lite::handleSyscall(), TraceThread::handleSyscallFunc(), RoutineTracerOndemand::RtnThread::printStack(), lite::syscallEnterRunModel(), lite::syscallExitRunModel(), and TraceThread::unblock().

6.364.4.8 getWakeupMsg()

```
void* Thread::getWakeupMsg ( ) const  [inline]
```

Definition at line 73 of file thread.h.

References m_wakeup_msg.

6.364.4.9 getWakeupTime()

```
SubsecondTime Thread::getWakeupTime ( ) const  [inline]
```

Definition at line 72 of file thread.h.

References m_wakeup_time.

Referenced by ThreadManager::stallThread().

6.364.4.10 reschedule()

```
bool Thread::reschedule (
            SubsecondTime & time,
            Core * current_core )
```

Definition at line 55 of file thread.cc.

References getId(), Core::getPerformanceModel(), m_core, PerformanceModel::queuePseudoInstruction(), ThreadManager::STALL_UNSCHEDULED, SyncInstruction::UNSCHEDULED, and updateCoreTLS().

Referenced by SyncClient::__mutexLock(), SyncClient::barrierWait(), SyncClient::condWait(), InstructionModeling::handleBasicBlock(), handleCheckScheduled(), SyscallMdl::handleFutexCall(), TraceThread::handleInstructionCountFunc(), ThreadManager::joinThread(), TraceThread::run(), SyscallMdl::runEnter(), SyscallMdl::runExit(), and TraceThread::unblock().

6.364.4.11 setCore()

```
void Thread::setCore (
            Core * core )
```

Definition at line 35 of file thread.cc.

References getId(), Core::getId(), Core::getThread(), LOG_ASSERT_ERROR, m_core, and Core::setThread().

Referenced by ThreadManager::createThread_unlocked(), ThreadManager::moveThread(), and ThreadManager::onThreadExit().

6.364.4.12 setName()

```
void Thread::setName (
             String name )  [inline]
```

Definition at line 46 of file thread.h.

References m_name.

6.364.4.13 setVa2paFunc()

```
void Thread::setVa2paFunc (
              va2pa_func_t va2pa_func,
              UInt64 m_va2pa_arg )
```

Definition at line 49 of file thread.cc.

References m_va2pa_arg, and m_va2pa_func.

Referenced by TraceThread::TraceThread().

6.364.4.14 signal()

```
void Thread::signal (
              SubsecondTime time,
             void * msg = NULL )  [inline]
```

Definition at line 66 of file thread.h.

References m_cond, m_wakeup_msg, m_wakeup_time, and ConditionVariable::signal().

Referenced by ThreadManager::onThreadStart(), and ThreadManager::resumeThread().

6.364.4.15 updateCoreTLS()

```
bool Thread::updateCoreTLS (
             int threadIndex = -1 )
```

Definition at line 76 of file thread.cc.

References Core::getId(), and m_core.

Referenced by ThreadManager::onThreadExit(), ThreadManager::onThreadStart(), and reschedule().

6.364.4.16 va2pa()

```
UInt64 Thread::va2pa (
            UInt64 logical_address ) const  [inline]
```

Definition at line 52 of file thread.h.

References m_va2pa_arg, and m_va2pa_func.

6.364.4.17 wait()

```
SubsecondTime Thread::wait (
            Lock & lock )  [inline]
```

Definition at line 60 of file thread.h.

References m_cond, m_wakeup_msg, m_wakeup_time, and ConditionVariable::wait().

Referenced by ThreadManager::stallThread().

6.364.5 Member Data Documentation

6.364.5.1 clear_tid

```
bool Thread::clear_tid
```

Definition at line 39 of file thread.h.

Referenced by lite::handleSyscall(), and ThreadManager::onThreadExit().

6.364.5.2 m_app_id

```
app_id_t Thread::m_app_id  [private]
```

Definition at line 19 of file thread.h.

Referenced by getAppId().

6.364.5.3 m_cond

```
ConditionVariable Thread::m_cond  [private]
```

Definition at line 22 of file thread.h.

Referenced by signal(), and wait().

6.364.5.4 m_core

```
Core* Thread::m_core  [private]
```

Definition at line 25 of file thread.h.

Referenced by getCore(), reschedule(), setCore(), and updateCoreTLS().

6.364.5.5 m_name

```
String Thread::m_name  [private]
```

Definition at line 20 of file thread.h.

Referenced by getName(), and setName().

6.364.5.6 m_os_info

```
struct { ... } Thread::m_os_info
```

Referenced by TraceThread::handleEmuFunc(), lite::handleSyscall(), ThreadManager::onThreadExit(), lite↩ ::syscallExitRunModel(), and Thread().

6.364.5.7 m_rtn_tracer

```
RoutineTracerThread* Thread::m_rtn_tracer  [private]
```

Definition at line 28 of file thread.h.

Referenced by getRoutineTracer(), Thread(), and ∼Thread().

6.364.5.8 m_sync_client

```
SyncClient* Thread::m_sync_client  [private]
```

Definition at line 27 of file thread.h.

Referenced by getSyncClient(), Thread(), and ~Thread().

6.364.5.9 m_syscall_model

```
SyscallMdl* Thread::m_syscall_model  [private]
```

Definition at line 26 of file thread.h.

Referenced by getSyscallMdl(), Thread(), and ~Thread().

6.364.5.10 m_thread_id

```
thread_id_t Thread::m_thread_id  [private]
```

Definition at line 18 of file thread.h.

Referenced by getId().

6.364.5.11 m_va2pa_arg

```
UInt64 Thread::m_va2pa_arg  [private]
```

Definition at line 30 of file thread.h.

Referenced by setVa2paFunc(), and va2pa().

6.364.5.12 m_va2pa_func

```
va2pa_func_t Thread::m_va2pa_func  [private]
```

Definition at line 29 of file thread.h.

Referenced by setVa2paFunc(), and va2pa().

6.364.5.13 m_wakeup_msg

```
void* Thread::m_wakeup_msg  [private]
```

Definition at line 24 of file thread.h.

Referenced by getWakeupMsg(), signal(), and wait().

6.364.5.14 m_wakeup_time

```
SubsecondTime Thread::m_wakeup_time  [private]
```

Definition at line 23 of file thread.h.

Referenced by getWakeupTime(), signal(), and wait().

6.364.5.15 tid

```
pid_t Thread::tid
```

Definition at line 37 of file thread.h.

Referenced by TraceThread::handleEmuFunc(), and lite::syscallExitRunModel().

6.364.5.16 tid_ptr

```
IntPtr Thread::tid_ptr
```

Definition at line 38 of file thread.h.

Referenced by lite::handleSyscall(), and ThreadManager::onThreadExit().

The documentation for this class was generated from the following files:

- common/core/ **thread.h**
- common/core/ **thread.cc**

6.365 HooksManager::ThreadCreate Struct Reference

```
#include <hooks_manager.h>
```

Public Attributes

- **thread_id_t thread_id**
- **thread_id_t creator_thread_id**

6.365.1 Detailed Description

Definition at line 73 of file hooks_manager.h.

6.365.2 Member Data Documentation

6.365.2.1 creator_thread_id

```
thread_id_t HooksManager::ThreadCreate::creator_thread_id
```

Definition at line 75 of file hooks_manager.h.

Referenced by hookCallbackThreadCreateType().

6.365.2.2 thread_id

```
thread_id_t HooksManager::ThreadCreate::thread_id
```

Definition at line 74 of file hooks_manager.h.

Referenced by ThreadStatsManager::hook_thread_create(), and hookCallbackThreadCreateType().

The documentation for this struct was generated from the following file:

- common/system/ **hooks_manager.h**

6.366 SchedulerPinnedBase::ThreadInfo Class Reference

```
#include <scheduler_pinned_base.h>
```

Public Member Functions

- **ThreadInfo** ()
- void **clearAffinity** ()
- void **setAffinitySingle** (**core_id_t** core_id)
- void **addAffinity** (**core_id_t** core_id)
- bool **hasAffinity** (**core_id_t** core_id) const
- String **getAffinityString** () const
- bool **hasAffinity** () const
- bool **hasExplicitAffinity** () const
- void **setExplicitAffinity** ()
- void **setCoreRunning** (**core_id_t** core_id)
- **core_id_t** **getCoreRunning** () const
- bool **isRunning** () const
- void **setLastScheduledIn** (**SubsecondTime** time)
- void **setLastScheduledOut** (**SubsecondTime** time)
- **SubsecondTime** **getLastScheduledIn** () const
- **SubsecondTime** **getLastScheduledOut** () const

Private Attributes

- bool **m_has_affinity**
- bool **m_explicit_affinity**
- std::vector< bool > **m_core_affinity**
- **core_id_t** **m_core_running**
- **SubsecondTime** **m_last_scheduled_in**
- **SubsecondTime** **m_last_scheduled_out**

6.366.1 Detailed Description

Definition at line 24 of file scheduler_pinned_base.h.

6.366.2 Constructor & Destructor Documentation

6.366.2.1 ThreadInfo()

```
SchedulerPinnedBase::ThreadInfo::ThreadInfo ( )  [inline]
```

Definition at line 27 of file scheduler_pinned_base.h.

6.366.3 Member Function Documentation

6.366.3.1 addAffinity()

```
void SchedulerPinnedBase::ThreadInfo::addAffinity (
             core_id_t core_id )  [inline]
```

Definition at line 46 of file scheduler_pinned_base.h.

References m_core_affinity, and m_has_affinity.

Referenced by setAffinitySingle().

6.366.3.2 clearAffinity()

```
void SchedulerPinnedBase::ThreadInfo::clearAffinity ( )  [inline]
```

Definition at line 36 of file scheduler_pinned_base.h.

References m_core_affinity.

Referenced by setAffinitySingle().

6.366.3.3 getAffinityString()

```
String SchedulerPinnedBase::ThreadInfo::getAffinityString ( ) const
```

Definition at line 292 of file scheduler_pinned_base.cc.

References hasAffinity().

6.366.3.4 getCoreRunning()

```
core_id_t SchedulerPinnedBase::ThreadInfo::getCoreRunning ( ) const  [inline]
```

Definition at line 54 of file scheduler_pinned_base.h.

References m_core_running.

6.366.3.5 getLastScheduledIn()

```
SubsecondTime SchedulerPinnedBase::ThreadInfo::getLastScheduledIn ( ) const  [inline]
```

Definition at line 59 of file scheduler_pinned_base.h.

References m_last_scheduled_in.

6.366.3.6 getLastScheduledOut()

```
SubsecondTime SchedulerPinnedBase::ThreadInfo::getLastScheduledOut ( ) const  [inline]
```

Definition at line 60 of file scheduler_pinned_base.h.

References m_last_scheduled_out.

6.366.3.7 hasAffinity() [1/2]

```
bool SchedulerPinnedBase::ThreadInfo::hasAffinity ( ) const  [inline]
```

Definition at line 50 of file scheduler_pinned_base.h.

References m_has_affinity.

Referenced by getAffinityString().

6.366.3.8 hasAffinity() [2/2]

```
bool SchedulerPinnedBase::ThreadInfo::hasAffinity (
            core_id_t core_id ) const  [inline]
```

Definition at line 47 of file scheduler_pinned_base.h.

References m_core_affinity.

6.366.3.9 hasExplicitAffinity()

```
bool SchedulerPinnedBase::ThreadInfo::hasExplicitAffinity ( ) const  [inline]
```

Definition at line 51 of file scheduler_pinned_base.h.

References m_explicit_affinity.

6.366.3.10 isRunning()

```
bool SchedulerPinnedBase::ThreadInfo::isRunning ( ) const  [inline]
```

Definition at line 55 of file scheduler_pinned_base.h.

References INVALID_CORE_ID, and m_core_running.

6.366.3.11 setAffinitySingle()

```
void SchedulerPinnedBase::ThreadInfo::setAffinitySingle (
              core_id_t core_id )  [inline]
```

Definition at line 41 of file scheduler_pinned_base.h.

References addAffinity(), and clearAffinity().

6.366.3.12 setCoreRunning()

```
void SchedulerPinnedBase::ThreadInfo::setCoreRunning (
              core_id_t core_id )  [inline]
```

Definition at line 53 of file scheduler_pinned_base.h.

References m_core_running.

6.366.3.13 setExplicitAffinity()

```
void SchedulerPinnedBase::ThreadInfo::setExplicitAffinity ( )  [inline]
```

Definition at line 52 of file scheduler_pinned_base.h.

References m_explicit_affinity.

6.366.3.14 setLastScheduledIn()

```
void SchedulerPinnedBase::ThreadInfo::setLastScheduledIn (
              SubsecondTime time )  [inline]
```

Definition at line 57 of file scheduler_pinned_base.h.

References m_last_scheduled_in.

6.366.3.15 setLastScheduledOut()

```
void SchedulerPinnedBase::ThreadInfo::setLastScheduledOut (
              SubsecondTime time )  [inline]
```

Definition at line 58 of file scheduler_pinned_base.h.

References m_last_scheduled_out.

6.366.4 Member Data Documentation

6.366.4.1 m_core_affinity

```
std::vector<bool> SchedulerPinnedBase::ThreadInfo::m_core_affinity  [private]
```

Definition at line 64 of file scheduler_pinned_base.h.

Referenced by addAffinity(), clearAffinity(), and hasAffinity().

6.366.4.2 m_core_running

```
core_id_t SchedulerPinnedBase::ThreadInfo::m_core_running  [private]
```

Definition at line 65 of file scheduler_pinned_base.h.

Referenced by getCoreRunning(), isRunning(), and setCoreRunning().

6.366.4.3 m_explicit_affinity

```
bool SchedulerPinnedBase::ThreadInfo::m_explicit_affinity  [private]
```

Definition at line 63 of file scheduler_pinned_base.h.

Referenced by hasExplicitAffinity(), and setExplicitAffinity().

6.366.4.4 m_has_affinity

```
bool SchedulerPinnedBase::ThreadInfo::m_has_affinity  [private]
```

Definition at line 62 of file scheduler_pinned_base.h.

Referenced by addAffinity(), and hasAffinity().

6.366.4.5 m_last_scheduled_in

```
SubsecondTime SchedulerPinnedBase::ThreadInfo::m_last_scheduled_in  [private]
```

Definition at line 66 of file scheduler_pinned_base.h.

Referenced by getLastScheduledIn(), and setLastScheduledIn().

6.366.4.6 m_last_scheduled_out

SubsecondTime SchedulerPinnedBase::ThreadInfo::m_last_scheduled_out [private]

Definition at line 67 of file scheduler_pinned_base.h.

Referenced by getLastScheduledOut(), and setLastScheduledOut().

The documentation for this class was generated from the following files:

- common/scheduler/ **scheduler_pinned_base.h**
- common/scheduler/ **scheduler_pinned_base.cc**

6.367 ThreadLocalStorage Struct Reference

#include <local_storage.h>

Public Attributes

- **Thread** * **thread**
- **DynamicInstruction** * **dynins**
- struct {
  thread_id_t **new_thread_id**
  } **clone**

- **SpinLoopDetectionState** **sld**
- struct {
  ADDRINT **eip**
  size_t **size**
  } **malloc**

- ADDRINT **lastCallSite**
- char * **scratch** [**NUM_SCRATCHPADS**]

Static Public Attributes

- static const unsigned int **NUM_SCRATCHPADS** = 3
- static const size_t **SCRATCHPAD_SIZE** = 1024

6.367.1 Detailed Description

Definition at line 15 of file local_storage.h.

6.367.2 Member Data Documentation

6.367.2.1 clone

```
struct { ...  } ThreadLocalStorage::clone
```

6.367.2.2 dynins

```
DynamicInstruction* ThreadLocalStorage::dynins
```

Definition at line 18 of file local_storage.h.

6.367.2.3 eip

```
ADDRINT ThreadLocalStorage::eip
```

Definition at line 31 of file local_storage.h.

6.367.2.4 lastCallSite

```
ADDRINT ThreadLocalStorage::lastCallSite
```

Definition at line 34 of file local_storage.h.

6.367.2.5 malloc

```
struct { ...  } ThreadLocalStorage::malloc
```

6.367.2.6 new_thread_id

```
thread_id_t ThreadLocalStorage::new_thread_id
```

Definition at line 24 of file local_storage.h.

6.367.2.7 NUM_SCRATCHPADS

```
const unsigned int ThreadLocalStorage::NUM_SCRATCHPADS = 3  [static]
```

Definition at line 37 of file local_storage.h.

Referenced by lite::addMemoryModeling(), and threadStartCallback().

6.367.2.8 scratch

```
char* ThreadLocalStorage::scratch[ NUM_SCRATCHPADS]
```

Definition at line 39 of file local_storage.h.

6.367.2.9 SCRATCHPAD_SIZE

```
const size_t ThreadLocalStorage::SCRATCHPAD_SIZE = 1024  [static]
```

Definition at line 38 of file local_storage.h.

Referenced by threadStartCallback().

6.367.2.10 size

```
size_t ThreadLocalStorage::size
```

Definition at line 32 of file local_storage.h.

6.367.2.11 sld

```
SpinLoopDetectionState ThreadLocalStorage::sld
```

Definition at line 27 of file local_storage.h.

6.367.2.12 thread

```
Thread* ThreadLocalStorage::thread
```

Definition at line 17 of file local_storage.h.

The documentation for this struct was generated from the following file:

- pin/ **local_storage.h**

6.368 ThreadManager Class Reference

`#include <thread_manager.h>`

Classes

- struct **ThreadSpawnRequest**
- struct **ThreadState**

Public Types

- enum **stall_type_t** {
  STALL_UNSCHEDULED, **STALL_BROKEN**, **STALL_JOIN**, **STALL_MUTEX**,
  STALL_COND, **STALL_BARRIER**, **STALL_FUTEX**, **STALL_PAUSE**,
  STALL_SLEEP, **STALL_SYSCALL**, **STALL_TYPES_MAX** }
- typedef void *(* **thread_func_t**) (void *)

Public Member Functions

- **ThreadManager** ()
- ~**ThreadManager** ()
- **Lock** & **getLock** ()
- **Scheduler** * **getScheduler** () const
- **Thread** * **createThread** (**app_id_t** app_id, **thread_id_t** creator_thread_id)
- **Thread** * **getThreadFromID** (**thread_id_t** thread_id)
- **Thread** * **getCurrentThread** (int threadIndex=-1)
- **UInt64** **getNumThreads** () const
- **Core::State** **getThreadState** (**thread_id_t** thread_id) const
- **stall_type_t** **getThreadStallReason** (**thread_id_t** thread_id) const
- **Thread** * **findThreadByTid** (pid_t tid)
- **thread_id_t** **spawnThread** (**thread_id_t** thread_id, **app_id_t** app_id)
- void **joinThread** (**thread_id_t** thread_id, **thread_id_t** join_thread_id)
- **thread_id_t** **getThreadToSpawn** (**SubsecondTime** &time)
- void **waitForThreadStart** (**thread_id_t** thread_id, **thread_id_t** wait_thread_id)
- void **onThreadStart** (**thread_id_t** thread_id, **SubsecondTime** time)
- void **onThreadExit** (**thread_id_t** thread_id)
- **SubsecondTime** **stallThread** (**thread_id_t** thread_id, **stall_type_t** reason, **SubsecondTime** time)
- void **stallThread_async** (**thread_id_t** thread_id, **stall_type_t** reason, **SubsecondTime** time)
- void **resumeThread** (**thread_id_t** thread_id, **thread_id_t** thread_id_by, **SubsecondTime** time, void *msg=NULL)
- void **resumeThread_async** (**thread_id_t** thread_id, **thread_id_t** thread_id_by, **SubsecondTime** time, void *msg=NULL)
- bool **isThreadRunning** (**thread_id_t** thread_id)
- bool **isThreadInitializing** (**thread_id_t** thread_id)
- bool **anyThreadRunning** ()
- void **moveThread** (**thread_id_t** thread_id, **core_id_t** core_id, **SubsecondTime** time)
- bool **areAllCoresRunning** ()

Static Public Attributes

- static const char * **stall_type_names** [ ]

Private Member Functions

- **Thread** * **createThread_unlocked** (**app_id_t** app_id, **thread_id_t** creator_thread_id)
- void **wakeUpWaiter** (**thread_id_t** thread_id, **SubsecondTime** time)

Private Attributes

- **Lock m_thread_lock**
- std::vector< **ThreadState** > **m_thread_state**
- std::queue< **ThreadSpawnRequest** > **m_thread_spawn_list**
- std::vector< **Thread** * > **m_threads**
- **TLS** * **m_thread_tls**
- **Scheduler** * **m_scheduler**

6.368.1 Detailed Description

Definition at line 17 of file thread_manager.h.

6.368.2 Member Typedef Documentation

6.368.2.1 thread_func_t

```
typedef void*(* ThreadManager::thread_func_t) (void *)
```

Definition at line 20 of file thread_manager.h.

6.368.3 Member Enumeration Documentation

6.368.3.1 stall_type_t

```
enum ThreadManager::stall_type_t
```

Enumerator

STALL_UNSCHEDULED	
STALL_BROKEN	
STALL_JOIN	
STALL_MUTEX	
STALL_COND	
STALL_BARRIER	
STALL_FUTEX	
STALL_PAUSE	
STALL_SLEEP	
STALL_SYSCALL	
STALL_TYPES_MAX	

Definition at line 22 of file thread_manager.h.

6.368.4 Constructor & Destructor Documentation

6.368.4.1 ThreadManager()

```
ThreadManager::ThreadManager ( )
```

Definition at line 27 of file thread_manager.cc.

6.368.4.2 ~ThreadManager()

```
ThreadManager::~ThreadManager ( )
```

Definition at line 33 of file thread_manager.cc.

References Core::IDLE, m_scheduler, m_thread_state, m_thread_tls, and m_threads.

6.368.5 Member Function Documentation

6.368.5.1 anyThreadRunning()

```
bool ThreadManager::anyThreadRunning ( )
```

Definition at line 396 of file thread_manager.cc.

References getNumThreads(), isThreadInitializing(), and isThreadRunning().

Referenced by stallThread().

6.368.5.2 areAllCoresRunning()

```
bool ThreadManager::areAllCoresRunning ( )
```

Definition at line 275 of file thread_manager.cc.

References Core::IDLE, and m_thread_state.

6.368.5.3 createThread()

```
Thread * ThreadManager::createThread (
            app_id_t app_id,
            thread_id_t creator_thread_id )
```

Definition at line 68 of file thread_manager.cc.

References createThread_unlocked(), and m_thread_lock.

6.368.5.4 createThread_unlocked()

```
Thread * ThreadManager::createThread_unlocked (
            app_id_t app_id,
            thread_id_t creator_thread_id )  [private]
```

Definition at line 74 of file thread_manager.cc.

References CLOG, StatsManager::EVENT_THREAD_CREATE, Thread::getId(), HookType::HOOK_THREAD_C↩ REATE, Core::INITIALIZING, INVALID_CORE_ID, m_scheduler, m_thread_state, m_threads, SubsecondTime::↩ MaxTime(), Thread::setCore(), Core::setState(), and Scheduler::threadCreate().

Referenced by createThread(), and spawnThread().

6.368.5.5 findThreadByTid()

```
Thread * ThreadManager::findThreadByTid (
           pid_t tid )
```

Definition at line 58 of file thread_manager.cc.

References m_threads.

6.368.5.6 getCurrentThread()

```
Thread * ThreadManager::getCurrentThread (
           int threadIndex = -1 )
```

Definition at line 53 of file thread_manager.cc.

References TLS::getPtr(), and m_thread_tls.

6.368.5.7 getLock()

```
Lock& ThreadManager::getLock ( )  [inline]
```

Definition at line 40 of file thread_manager.h.

References m_thread_lock.

Referenced by getThreadToSpawn(), joinThread(), spawnThread(), and waitForThreadStart().

6.368.5.8 getNumThreads()

```
UInt64 ThreadManager::getNumThreads ( ) const  [inline]
```

Definition at line 47 of file thread_manager.h.

References m_threads.

Referenced by anyThreadRunning().

6.368.5.9 getScheduler()

```
Scheduler* ThreadManager::getScheduler ( ) const  [inline]
```

Definition at line 41 of file thread_manager.h.

References m_scheduler.

6.368.5.10 getThreadFromID()

```
Thread * ThreadManager::getThreadFromID (
            thread_id_t thread_id )
```

Definition at line 48 of file thread_manager.cc.

References LOG_ASSERT_ERROR, and m_threads.

Referenced by joinThread(), moveThread(), onThreadExit(), onThreadStart(), resumeThread(), spawnThread(), stallThread(), and waitForThreadStart().

6.368.5.11 getThreadStallReason()

```
stall_type_t ThreadManager::getThreadStallReason (
            thread_id_t thread_id ) const  [inline]
```

Definition at line 49 of file thread_manager.h.

References m_thread_state.

6.368.5.12 getThreadState()

```
Core::State ThreadManager::getThreadState (
            thread_id_t thread_id ) const  [inline]
```

Definition at line 48 of file thread_manager.h.

References m_thread_state.

Referenced by moveThread().

6.368.5.13 getThreadToSpawn()

```
thread_id_t ThreadManager::getThreadToSpawn (
            SubsecondTime & time )
```

Definition at line 213 of file thread_manager.cc.

References getLock(), LOG_ASSERT_ERROR, m_thread_spawn_list, ThreadManager::ThreadSpawnRequest↩
::thread_id, and ThreadManager::ThreadSpawnRequest::time.

6.368.5.14 isThreadInitializing()

```
bool ThreadManager::isThreadInitializing (
            thread_id_t thread_id )
```

Definition at line 391 of file thread_manager.cc.

References Core::INITIALIZING, and m_thread_state.

Referenced by anyThreadRunning().

6.368.5.15 isThreadRunning()

```
bool ThreadManager::isThreadRunning (
            thread_id_t thread_id )
```

Definition at line 386 of file thread_manager.cc.

References m_thread_state, and Core::RUNNING.

Referenced by anyThreadRunning().

6.368.5.16 joinThread()

```
void ThreadManager::joinThread (
            thread_id_t thread_id,
            thread_id_t join_thread_id )
```

Definition at line 291 of file thread_manager.cc.

References Thread::getCore(), PerformanceModel::getElapsedTime(), getLock(), Core::getPerformanceModel(), getThreadFromID(), Core::IDLE, INVALID_THREAD_ID, SyncInstruction::JOIN, LOG_ASSERT_ERROR, LOG↩_PRINT, m_thread_state, PerformanceModel::queuePseudoInstruction(), Thread::reschedule(), STALL_JOIN, and stallThread().

6.368.5.17 moveThread()

```
void ThreadManager::moveThread (
            thread_id_t thread_id,
            core_id_t core_id,
            SubsecondTime time )
```

Definition at line 241 of file thread_manager.cc.

References CLOG, Thread::getCore(), Core::getId(), getThreadFromID(), getThreadState(), HookType::HOOK_↩THREAD_MIGRATE, Core::IDLE, INVALID_CORE_ID, INVALID_THREAD_ID, m_thread_state, resumeThread(), Core::RUNNING, Thread::setCore(), Core::setState(), STALL_UNSCHEDULED, and Core::STALLED.

Referenced by SchedulerDynamic::moveThread().

6.368.5.18 onThreadExit()

```
void ThreadManager::onThreadExit (
            thread_id_t thread_id )
```

Definition at line 148 of file thread_manager.cc.

References Core::accessMemory(), Thread::clear_tid, CLOG, StatsManager::EVENT_THREAD_EXIT, FUTE↩X_BITSET_MATCH_ANY, Thread::getCore(), PerformanceModel::getElapsedTime(), Core::getId(), Core::get↩PerformanceModel(), getThreadFromID(), HookType::HOOK_THREAD_EXIT, Core::IDLE, LOG_ASSERT_ERR↩OR, Thread::m_os_info, m_thread_lock, m_thread_state, m_thread_tls, SubsecondTime::MaxTime(), Core::NO↩NE, Core::RUNNING, TLS::set(), Thread::setCore(), Core::setState(), Thread::tid_ptr, Thread::updateCoreTLS(), wakeUpWaiter(), and Core::WRITE.

6.368.5.19 onThreadStart()

```
void ThreadManager::onThreadStart (
              thread_id_t thread_id,
              SubsecondTime time )
```

Definition at line 99 of file thread_manager.cc.

References CLOG, Thread::getCore(), PerformanceModel::getElapsedTime(), Core::getId(), Core::get↩ PerformanceModel(), getThreadFromID(), HookType::HOOK_THREAD_MIGRATE, HookType::HOOK_THRE↩ AD_START, INVALID_THREAD_ID, LOG_PRINT, m_thread_lock, m_thread_state, m_thread_tls, Performance↩ Model::queuePseudoInstruction(), Core::RUNNING, TLS::set(), Core::setState(), Thread::signal(), STALL_UNS↩ CHEDULED, Core::STALLED, and Thread::updateCoreTLS().

6.368.5.20 resumeThread()

```
void ThreadManager::resumeThread (
              thread_id_t thread_id,
              thread_id_t thread_id_by,
              SubsecondTime time,
             void * msg = NULL )
```

Definition at line 378 of file thread_manager.cc.

References getThreadFromID(), resumeThread_async(), and Thread::signal().

Referenced by moveThread(), and wakeUpWaiter().

6.368.5.21 resumeThread_async()

```
void ThreadManager::resumeThread_async (
              thread_id_t thread_id,
              thread_id_t thread_id_by,
              SubsecondTime time,
             void * msg = NULL )
```

Definition at line 368 of file thread_manager.cc.

References CLOG, HookType::HOOK_THREAD_RESUME, LOG_PRINT, m_thread_state, and Core::RUNNING.

Referenced by resumeThread().

6.368.5.22 spawnThread()

```
thread_id_t ThreadManager::spawnThread (
            thread_id_t thread_id,
            app_id_t app_id )
```

Definition at line 190 of file thread_manager.cc.

References createThread_unlocked(), Thread::getCore(), PerformanceModel::getElapsedTime(), Thread::getId(), getLock(), Core::getPerformanceModel(), getThreadFromID(), INVALID_THREAD_ID, LOG_PRINT, m_thread_↩ spawn_list, and SubsecondTime::Zero().

6.368.5.23 stallThread()

```
SubsecondTime ThreadManager::stallThread (
            thread_id_t thread_id,
            stall_type_t reason,
            SubsecondTime time )
```

Definition at line 350 of file thread_manager.cc.

References anyThreadRunning(), getThreadFromID(), Thread::getWakeupTime(), m_thread_lock, m_thread_state, Core::RUNNING, stallThread_async(), and Thread::wait().

Referenced by joinThread().

6.368.5.24 stallThread_async()

```
void ThreadManager::stallThread_async (
            thread_id_t thread_id,
            stall_type_t reason,
            SubsecondTime time )
```

Definition at line 339 of file thread_manager.cc.

References CLOG, HookType::HOOK_THREAD_STALL, LOG_PRINT, m_thread_state, stall_type_names, and Core::STALLED.

Referenced by stallThread().

6.368.5.25 waitForThreadStart()

```
void ThreadManager::waitForThreadStart (
            thread_id_t thread_id,
            thread_id_t wait_thread_id )
```

Definition at line 226 of file thread_manager.cc.

References getLock(), getThreadFromID(), Core::INITIALIZING, INVALID_THREAD_ID, LOG_ASSERT_ERROR, and m_thread_state.

6.368.5.26 wakeUpWaiter()

```
void ThreadManager::wakeUpWaiter (
              thread_id_t thread_id,
              SubsecondTime time )  [private]
```

Definition at line 325 of file thread_manager.cc.

References INVALID_THREAD_ID, itostr(), LOG_PRINT, m_thread_state, and resumeThread().

Referenced by onThreadExit().

6.368.6 Member Data Documentation

6.368.6.1 m_scheduler

```
Scheduler* ThreadManager::m_scheduler  [private]
```

Definition at line 102 of file thread_manager.h.

Referenced by createThread_unlocked(), getScheduler(), and ~ThreadManager().

6.368.6.2 m_thread_lock

```
Lock ThreadManager::m_thread_lock  [private]
```

Definition at line 94 of file thread_manager.h.

Referenced by createThread(), getLock(), onThreadExit(), onThreadStart(), and stallThread().

6.368.6.3 m_thread_spawn_list

```
std::queue< ThreadSpawnRequest> ThreadManager::m_thread_spawn_list  [private]
```

Definition at line 97 of file thread_manager.h.

Referenced by getThreadToSpawn(), and spawnThread().

6.368.6.4 m_thread_state

```
std::vector< ThreadState> ThreadManager::m_thread_state  [private]
```

Definition at line 96 of file thread_manager.h.

Referenced by areAllCoresRunning(), createThread_unlocked(), getThreadStallReason(), getThreadState(), isThreadInitializing(), isThreadRunning(), joinThread(), moveThread(), onThreadExit(), onThreadStart(), resumeThread_async(), stallThread(), stallThread_async(), waitForThreadStart(), wakeUpWaiter(), and ~ThreadManager().

6.368.6.5 m_thread_tls

```
TLS* ThreadManager::m_thread_tls  [private]
```

Definition at line 100 of file thread_manager.h.

Referenced by getCurrentThread(), onThreadExit(), onThreadStart(), and ~ThreadManager().

6.368.6.6 m_threads

```
std::vector< Thread*> ThreadManager::m_threads  [private]
```

Definition at line 99 of file thread_manager.h.

Referenced by createThread_unlocked(), findThreadByTid(), getNumThreads(), getThreadFromID(), and ~ThreadManager().

6.368.6.7 stall_type_names

```
const char * ThreadManager::stall_type_names  [static]
```

Initial value:

```
= {
   "unscheduled", "broken", "join", "mutex", "cond", "barrier", "futex", "pause", "sleep", "syscall"
}
```

Definition at line 35 of file thread_manager.h.

Referenced by hookCallbackThreadStallType(), RoutineTracerOndemand::RtnThread::printStack(), and stallThread_async().

The documentation for this class was generated from the following files:

- common/system/ **thread_manager.h**
- common/system/ **thread_manager.cc**

6.369 HooksManager::ThreadMigrate Struct Reference

`#include <hooks_manager.h>`

Public Attributes

- **thread_id_t thread_id**
- **core_id_t core_id**
- **subsecond_time_t time**

6.369.1 Detailed Description

Definition at line 91 of file hooks_manager.h.

6.369.2 Member Data Documentation

6.369.2.1 core_id

core_id_t HooksManager::ThreadMigrate::core_id

Definition at line 93 of file hooks_manager.h.

Referenced by hookCallbackThreadMigrateType(), and SmtTimer::threadMigrate().

6.369.2.2 thread_id

thread_id_t HooksManager::ThreadMigrate::thread_id

Definition at line 92 of file hooks_manager.h.

Referenced by hookCallbackThreadMigrateType(), BarrierSyncServer::threadMigrate(), and SmtTimer::threadMigrate().

6.369.2.3 time

subsecond_time_t HooksManager::ThreadMigrate::time

Definition at line 94 of file hooks_manager.h.

Referenced by hookCallbackThreadMigrateType().

The documentation for this struct was generated from the following file:

- common/system/ **hooks_manager.h**

6.370 HooksManager::ThreadResume Struct Reference

`#include <hooks_manager.h>`

Public Attributes

- **thread_id_t thread_id**
- **thread_id_t thread_by**
- **subsecond_time_t time**

6.370.1 Detailed Description

Definition at line 86 of file hooks_manager.h.

6.370.2 Member Data Documentation

6.370.2.1 thread_by

thread_id_t HooksManager::ThreadResume::thread_by

Definition at line 88 of file hooks_manager.h.

Referenced by SchedulerDynamic::hook_thread_resume(), ThreadStatsManager::hook_thread_resume(), and hookCallbackThreadResumeType().

6.370.2.2 thread_id

thread_id_t HooksManager::ThreadResume::thread_id

Definition at line 87 of file hooks_manager.h.

Referenced by SchedulerDynamic::hook_thread_resume(), ThreadStatsManager::hook_thread_resume(), hook↩ CallbackThreadResumeType(), and SmtTimer::threadResume().

6.370.2.3 time

subsecond_time_t HooksManager::ThreadResume::time

Definition at line 89 of file hooks_manager.h.

Referenced by SchedulerDynamic::hook_thread_resume(), ThreadStatsManager::hook_thread_resume(), and hookCallbackThreadResumeType().

The documentation for this struct was generated from the following file:

- common/system/ **hooks_manager.h**

6.371 ThreadManager::ThreadSpawnRequest Struct Reference

Public Attributes

- **thread_id_t thread_by**
- **thread_id_t thread_id**
- **SubsecondTime time**

6.371.1 Detailed Description

Definition at line 78 of file thread_manager.h.

6.371.2 Member Data Documentation

6.371.2.1 thread_by

```
thread_id_t ThreadManager::ThreadSpawnRequest::thread_by
```

Definition at line 80 of file thread_manager.h.

6.371.2.2 thread_id

```
thread_id_t ThreadManager::ThreadSpawnRequest::thread_id
```

Definition at line 81 of file thread_manager.h.

Referenced by ThreadManager::getThreadToSpawn().

6.371.2.3 time

```
SubsecondTime ThreadManager::ThreadSpawnRequest::time
```

Definition at line 82 of file thread_manager.h.

Referenced by ThreadManager::getThreadToSpawn().

The documentation for this struct was generated from the following file:

- common/system/ **thread_manager.h**

6.372 HooksManager::ThreadStall Struct Reference

`#include <hooks_manager.h>`

Public Attributes

- **thread_id_t thread_id**
- **ThreadManager::stall_type_t reason**
- **subsecond_time_t time**

6.372.1 Detailed Description

Definition at line 81 of file hooks_manager.h.

6.372.2 Member Data Documentation

6.372.2.1 reason

ThreadManager::stall_type_t HooksManager::ThreadStall::reason

Definition at line 83 of file hooks_manager.h.

Referenced by SchedulerDynamic::hook_thread_stall(), ThreadStatsManager::hook_thread_stall(), and hook↩ CallbackThreadStallType().

6.372.2.2 thread_id

thread_id_t HooksManager::ThreadStall::thread_id

Definition at line 82 of file hooks_manager.h.

Referenced by SchedulerDynamic::hook_thread_stall(), ThreadStatsManager::hook_thread_stall(), hook↩ CallbackThreadStallType(), BarrierSyncServer::threadStall(), and SmtTimer::threadStall().

6.372.2.3 time

subsecond_time_t HooksManager::ThreadStall::time

Definition at line 84 of file hooks_manager.h.

Referenced by SchedulerDynamic::hook_thread_stall(), ThreadStatsManager::hook_thread_stall(), and hook↩ CallbackThreadStallType().

The documentation for this struct was generated from the following file:

- common/system/ **hooks_manager.h**

6.373 RoutineTracerFunctionStats::ThreadStatAggregates Class Reference

`#include <routine_tracer_funcstats.h>`

Static Public Member Functions

- static void **registerStats** ()

Private Types

- enum **StatType** { **GLOBAL_INSTRUCTIONS**, **GLOBAL_NONIDLE_ELAPSED_TIME** }

Static Private Member Functions

- static **UInt64** **callback** (**ThreadStatsManager::ThreadStatType** type, **thread_id_t** thread_id, **Core** ∗core, **UInt64** user)

6.373.1 Detailed Description

Definition at line 106 of file routine_tracer_funcstats.h.

6.373.2 Member Enumeration Documentation

6.373.2.1 StatType

enum **RoutineTracerFunctionStats::ThreadStatAggregates::StatType** [private]

Enumerator

GLOBAL_INSTRUCTIONS	
GLOBAL_NONIDLE_ELAPSED_TIME	

Definition at line 111 of file routine_tracer_funcstats.h.

6.373.3 Member Function Documentation

6.373.3.1 callback()

```
UInt64 RoutineTracerFunctionStats::ThreadStatAggregates::callback (
            ThreadStatsManager::ThreadStatType type,
            thread_id_t thread_id,
            Core * core,
            UInt64 user ) [static], [private]
```

Definition at line 320 of file routine_tracer_funcstats.cc.

References LOG_PRINT_ERROR.

6.373.3.2 registerStats()

```
void RoutineTracerFunctionStats::ThreadStatAggregates::registerStats ( ) [static]
```

Definition at line 314 of file routine_tracer_funcstats.cc.

References ThreadStatsManager::DYNAMIC.

Referenced by RoutineTracerFunctionStats::RtnMaster::RtnMaster().

The documentation for this class was generated from the following files:

- common/system/ **routine_tracer_funcstats.h**
- common/system/ **routine_tracer_funcstats.cc**

6.374 ThreadStatAggregates Class Reference

Static Public Member Functions

- static void **registerStats** ()

Static Private Member Functions

- static **UInt64** **callback** (**ThreadStatsManager::ThreadStatType** type, **thread_id_t** thread_id, **Core** *core, **UInt64** user)

6.374.1 Detailed Description

Definition at line 306 of file routine_tracer_funcstats.cc.

6.374.2 Member Function Documentation

6.374.2.1 callback()

```
static  UInt64 ThreadStatAggregates::callback (
              ThreadStatsManager::ThreadStatType type,
              thread_id_t thread_id,
              Core * core,
              UInt64 user )  [static], [private]
```

6.374.2.2 registerStats()

```
static void ThreadStatAggregates::registerStats ( )  [static]
```

The documentation for this class was generated from the following file:

- common/system/ **routine_tracer_funcstats.cc**

6.375 RoutineTracerFunctionStats::ThreadStatCpiMem Class Reference

`#include <routine_tracer_funcstats.h>`

Static Public Member Functions

- static **ThreadStatsManager::ThreadStatType registerStat** ()

Private Member Functions

- **ThreadStatCpiMem** ()

Static Private Member Functions

- static **UInt64 callback** (**ThreadStatsManager::ThreadStatType** type, **thread_id_t** thread_id, **Core** *core, **UInt64** user)

Private Attributes

- std::vector< std::vector< **StatsMetricBase** * > > **m_stats**

6.375.1 Detailed Description

Definition at line 118 of file routine_tracer_funcstats.h.

6.375.2 Constructor & Destructor Documentation

6.375.2.1 ThreadStatCpiMem()

```
RoutineTracerFunctionStats::ThreadStatCpiMem::ThreadStatCpiMem ( )  [private]
```

Definition at line 350 of file routine_tracer_funcstats.cc.

References HitWhereIsValid(), HitWhereString(), LOG_ASSERT_ERROR, m_stats, HitWhere::NUM_HITWHERES, and HitWhere::WHERE_FIRST.

6.375.3 Member Function Documentation

6.375.3.1 callback()

```
UInt64 RoutineTracerFunctionStats::ThreadStatCpiMem::callback (
            ThreadStatsManager::ThreadStatType type,
            thread_id_t thread_id,
            Core * core,
            UInt64 user )  [static], [private]
```

Definition at line 370 of file routine_tracer_funcstats.cc.

References Core::getId(), and m_stats.

6.375.3.2 registerStat()

```
ThreadStatsManager::ThreadStatType RoutineTracerFunctionStats::ThreadStatCpiMem::registerStat
( )  [static]
```

Definition at line 344 of file routine_tracer_funcstats.cc.

References ThreadStatsManager::DYNAMIC.

Referenced by RoutineTracerFunctionStats::RtnMaster::RtnMaster().

6.375.4 Member Data Documentation

6.375.4.1 m_stats

```
std::vector<std::vector< StatsMetricBase*> > RoutineTracerFunctionStats::ThreadStatCpiMem↩
::m_stats  [private]
```

Definition at line 123 of file routine_tracer_funcstats.h.

Referenced by callback(), and ThreadStatCpiMem().

The documentation for this class was generated from the following files:

- common/system/ **routine_tracer_funcstats.h**
- common/system/ **routine_tracer_funcstats.cc**

6.376 ThreadManager::ThreadState Struct Reference

Public Member Functions

- **ThreadState** ()

Public Attributes

- **Core::State status**
- **stall_type_t stalled_reason**
- **thread_id_t waiter**

6.376.1 Detailed Description

Definition at line 85 of file thread_manager.h.

6.376.2 Constructor & Destructor Documentation

6.376.2.1 ThreadState()

```
ThreadManager::ThreadState::ThreadState ( )  [inline]
```

Definition at line 91 of file thread_manager.h.

6.376.3 Member Data Documentation

6.376.3.1 stalled_reason

```
stall_type_t ThreadManager::ThreadState::stalled_reason
```

Definition at line 88 of file thread_manager.h.

6.376.3.2 status

```
Core::State ThreadManager::ThreadState::status
```

Definition at line 87 of file thread_manager.h.

6.376.3.3 waiter

```
thread_id_t ThreadManager::ThreadState::waiter
```

Definition at line 89 of file thread_manager.h.

The documentation for this struct was generated from the following file:

- common/system/ **thread_manager.h**

6.377 ThreadStatNamedStat Class Reference

`#include <thread_stats_manager.h>`

Static Public Member Functions

- static **ThreadStatsManager::ThreadStatType registerStat** (const char *name, String objectName, String metricName)

Private Member Functions

- **ThreadStatNamedStat** (String objectName, String metricName)

Static Private Member Functions

- static **UInt64 callback** (**ThreadStatsManager::ThreadStatType** type, **thread_id_t** thread_id, **Core** *core, **UInt64** user)

Private Attributes

- std::vector< **StatsMetricBase** * > **m_stats**

6.377.1 Detailed Description

Definition at line 128 of file thread_stats_manager.h.

6.377.2 Constructor & Destructor Documentation

6.377.2.1 ThreadStatNamedStat()

```
ThreadStatNamedStat::ThreadStatNamedStat (
            String objectName,
            String metricName )  [private]
```

Definition at line 251 of file thread_stats_manager.cc.

6.377.3 Member Function Documentation

6.377.3.1 callback()

```
UInt64 ThreadStatNamedStat::callback (
            ThreadStatsManager::ThreadStatType type,
            thread_id_t thread_id,
            Core * core,
            UInt64 user )  [static], [private]
```

Definition at line 260 of file thread_stats_manager.cc.

References Core::getId(), and StatsMetricBase::recordMetric().

6.377.3.2 registerStat()

```
ThreadStatsManager::ThreadStatType ThreadStatNamedStat::registerStat (
            const char * name,
            String objectName,
            String metricName )  [static]
```

Definition at line 238 of file thread_stats_manager.cc.

References ThreadStatsManager::DYNAMIC, and ThreadStatsManager::INVALID.

Referenced by InstructionTracerFPStats::init(), registerPerThread(), RoutineTracerFunctionStats::RtnMaster::Rtn↩Master(), and SchedulerSequential::SchedulerSequential().

6.377.4 Member Data Documentation

6.377.4.1 m_stats

```
std::vector< StatsMetricBase*> ThreadStatNamedStat::m_stats  [private]
```

Definition at line 133 of file thread_stats_manager.h.

The documentation for this class was generated from the following files:

- common/system/ **thread_stats_manager.h**
- common/system/ **thread_stats_manager.cc**

6.378 ThreadStatsManager::ThreadStats Class Reference

```
#include <thread_stats_manager.h>
```

Public Member Functions

- **ThreadStats** (**thread_id_t** thread_id)
- void **update** (**SubsecondTime** time, bool init=false)

Public Attributes

- const **Thread** * **m_thread**
- **core_id_t** **m_core_id**
- std::vector< **UInt64** > **time_by_core**
- std::vector< **UInt64** > **insn_by_core**
- **SubsecondTime** **m_elapsed_time**
- **SubsecondTime** **m_unscheduled_time**

Private Attributes

- **SubsecondTime** **m_time_last**
- std::unordered_map< **ThreadStatType**, **UInt64** > **m_counts**
- std::unordered_map< **ThreadStatType**, **UInt64** > **m_last**

Friends

- class **ThreadStatsManager**

6.378.1 Detailed Description

Definition at line 25 of file thread_stats_manager.h.

6.378.2 Constructor & Destructor Documentation

6.378.2.1 ThreadStats()

```
ThreadStatsManager::ThreadStats::ThreadStats (
              thread_id_t thread_id )
```

Definition at line 170 of file thread_stats_manager.cc.

References ThreadStatsManager::getThreadStatName(), ThreadStatsManager::getThreadStatTypes(), insn_by_↩ core, itostr(), m_counts, m_elapsed_time, m_last, m_unscheduled_time, registerStatsMetric(), and time_by_core.

6.378.3 Member Function Documentation

6.378.3.1 update()

```
void ThreadStatsManager::ThreadStats::update (
              SubsecondTime time,
             bool init = false )
```

Definition at line 197 of file thread_stats_manager.cc.

References ThreadStatsManager::ELAPSED_NONIDLE_TIME, SubsecondTime::getFS(), Core::getId(), Performance↩ Model::getInstructionCount(), PerformanceModel::getNonIdleElapsedTime(), Core::getPerformanceModel(), Core::IDLE, PthreadEmu::init(), Core::INITIALIZING, ThreadStatsManager::INSTRUCTIONS, INVALID_COR↩ E_ID, and SubsecondTime::Zero().

6.378.4 Friends And Related Function Documentation

6.378.4.1 ThreadStatsManager

```
friend class ThreadStatsManager [friend]
```

Definition at line 43 of file thread_stats_manager.h.

6.378.5 Member Data Documentation

6.378.5.1 insn_by_core

std::vector< **UInt64** > ThreadStatsManager::ThreadStats::insn_by_core

Definition at line 31 of file thread_stats_manager.h.

Referenced by ThreadStats().

6.378.5.2 m_core_id

core_id_t ThreadStatsManager::ThreadStats::m_core_id

Definition at line 29 of file thread_stats_manager.h.

6.378.5.3 m_counts

std::unordered_map< **ThreadStatType**, **UInt64** > ThreadStatsManager::ThreadStats::m_counts [private]

Definition at line 40 of file thread_stats_manager.h.

Referenced by ThreadStats().

6.378.5.4 m_elapsed_time

SubsecondTime ThreadStatsManager::ThreadStats::m_elapsed_time

Definition at line 32 of file thread_stats_manager.h.

Referenced by ThreadStats().

6.378.5.5 m_last

std::unordered_map< **ThreadStatType**, **UInt64** > ThreadStatsManager::ThreadStats::m_last [private]

Definition at line 41 of file thread_stats_manager.h.

Referenced by ThreadStats().

6.378.5.6 m_thread

```
const  Thread* ThreadStatsManager::ThreadStats::m_thread
```

Definition at line 28 of file thread_stats_manager.h.

6.378.5.7 m_time_last

```
 SubsecondTime ThreadStatsManager::ThreadStats::m_time_last  [private]
```

Definition at line 39 of file thread_stats_manager.h.

6.378.5.8 m_unscheduled_time

```
 SubsecondTime ThreadStatsManager::ThreadStats::m_unscheduled_time
```

Definition at line 33 of file thread_stats_manager.h.

Referenced by ThreadStats().

6.378.5.9 time_by_core

```
std::vector< UInt64> ThreadStatsManager::ThreadStats::time_by_core
```

Definition at line 30 of file thread_stats_manager.h.

Referenced by ThreadStats().

The documentation for this class was generated from the following files:

- common/system/ **thread_stats_manager.h**
- common/system/ **thread_stats_manager.cc**

6.379 ThreadStatsManager Class Reference

```
#include <thread_stats_manager.h>
```

Classes

- struct **StatCallback**
- class **ThreadStats**

Public Types

- enum **ThreadStatTypeEnum** {
  INSTRUCTIONS, **ELAPSED_NONIDLE_TIME**, **WAITING_COST**, **NUM_THREAD_STAT_FIXED_TYP↩ ES**,
  DYNAMIC, **INVALID** }
- typedef **UInt32** **ThreadStatType**
- typedef std::vector< **ThreadStatType** > **ThreadStatTypeList**
- typedef **UInt64**(* **ThreadStatCallback**) (**ThreadStatType** type, **thread_id_t** thread_id, **Core** *core, **U↩ Int64** user)

Public Member Functions

- **ThreadStatsManager** ()
- ~**ThreadStatsManager** ()
- void **update** (**thread_id_t** thread_id= **INVALID_THREAD_ID**, **SubsecondTime** time= **SubsecondTime↩ ::MaxTime**())
- void **calculateWaitingCosts** (**SubsecondTime** time)
- const **ThreadStatTypeList** & **getThreadStatTypes** ()
- const char * **getThreadStatName** (**ThreadStatType** type)
- **UInt64** **getThreadStatistic** (**thread_id_t** thread_id, **ThreadStatType** type)
- **ThreadStatType** **registerThreadStatMetric** (**ThreadStatType** type, const char *name, **ThreadStat↩ Callback** func, **UInt64** user)

Private Member Functions

- **UInt64** **callThreadStatCallback** (**ThreadStatType** type, **thread_id_t** thread_id, **Core** *core)
- void **pre_stat_write** ()
- void **threadCreate** (**thread_id_t** thread_id)
- void **threadStart** (**thread_id_t** thread_id, **SubsecondTime** time)
- void **threadStall** (**thread_id_t** thread_id, **ThreadManager::stall_type_t** reason, **SubsecondTime** time)
- void **threadResume** (**thread_id_t** thread_id, **thread_id_t** thread_by, **SubsecondTime** time)
- void **threadExit** (**thread_id_t** thread_id, **SubsecondTime** time)

Static Private Member Functions

- static **UInt64** **metricCallback** (**ThreadStatType** type, **thread_id_t** thread_id, **Core** *core, **UInt64** user)
- static **SInt64** **hook_pre_stat_write** (**UInt64** ptr, **UInt64**)
- static **SInt64** **hook_thread_create** (**UInt64** ptr, **UInt64** _args)
- static **SInt64** **hook_thread_start** (**UInt64** ptr, **UInt64** _args)
- static **SInt64** **hook_thread_stall** (**UInt64** ptr, **UInt64** _args)
- static **SInt64** **hook_thread_resume** (**UInt64** ptr, **UInt64** _args)
- static **SInt64** **hook_thread_exit** (**UInt64** ptr, **UInt64** _args)

Private Attributes

- std::vector< **ThreadStats** * > **m_threads_stats**
- **ThreadStatTypeList** **m_thread_stat_types**
- std::unordered_map< **ThreadStatType**, **StatCallback** > **m_thread_stat_callbacks**
- **ThreadStatType** **m_next_dynamic_type**
- **BottleGraphManager** **m_bottlegraphs**
- **SubsecondTime** **m_waiting_time_last**

Static Private Attributes

- static const int **MAX_THREADS** = 4096

6.379.1 Detailed Description

Definition at line 11 of file thread_stats_manager.h.

6.379.2 Member Typedef Documentation

6.379.2.1 ThreadStatCallback

```
typedef UInt64(* ThreadStatsManager::ThreadStatCallback) ( ThreadStatType type, thread_id_t thread_id, Core *core, UInt64 user)
```

Definition at line 45 of file thread_stats_manager.h.

6.379.2.2 ThreadStatType

```
typedef UInt32 ThreadStatsManager::ThreadStatType
```

Definition at line 14 of file thread_stats_manager.h.

6.379.2.3 ThreadStatTypeList

```
typedef std::vector< ThreadStatType> ThreadStatsManager::ThreadStatTypeList
```

Definition at line 15 of file thread_stats_manager.h.

6.379.3 Member Enumeration Documentation

6.379.3.1 ThreadStatTypeEnum

```
enum ThreadStatsManager::ThreadStatTypeEnum
```

Enumerator

INSTRUCTIONS	
ELAPSED_NONIDLE_TIME	
WAITING_COST	
NUM_THREAD_STAT_FIXED_TYPES	
DYNAMIC	
INVALID	

Definition at line 16 of file thread_stats_manager.h.

6.379.4 Constructor & Destructor Documentation

6.379.4.1 ThreadStatsManager()

```
ThreadStatsManager::ThreadStatsManager ( )
```

Definition at line 10 of file thread_stats_manager.cc.

References ELAPSED_NONIDLE_TIME, HookType::HOOK_PRE_STAT_WRITE, hook_pre_stat_write(), HookType::HOOK_THREAD_CREATE, hook_thread_create(), HookType::HOOK_THREAD_EXIT, hook_thread_exit(), HookType::HOOK_THREAD_RESUME, hook_thread_resume(), HookType::HOOK_THREAD_STALL, hook_thread_stall(), HookType::HOOK_THREAD_START, hook_thread_start(), INSTRUCTIONS, metricCallback(), HooksManager::ORDER_NOTIFY_PRE, registerThreadStatMetric(), and WAITING_COST.

6.379.4.2 ~ThreadStatsManager()

```
ThreadStatsManager::~ThreadStatsManager ( )
```

Definition at line 31 of file thread_stats_manager.cc.

References m_threads_stats.

6.379.5 Member Function Documentation

6.379.5.1 calculateWaitingCosts()

```
void ThreadStatsManager::calculateWaitingCosts (
            SubsecondTime time )
```

Definition at line 74 of file thread_stats_manager.cc.

References SubsecondTime::getFS(), m_threads_stats, m_waiting_time_last, ThreadManager::STALL_UNSCHEDULED, and WAITING_COST.

Referenced by pre_stat_write(), threadExit(), threadResume(), threadStall(), and update().

6.379.5.2 callThreadStatCallback()

```
UInt64 ThreadStatsManager::callThreadStatCallback (
            ThreadStatType type,
            thread_id_t thread_id,
            Core * core ) [private]
```

Definition at line 50 of file thread_stats_manager.cc.

References m_thread_stat_callbacks.

6.379.5.3 getThreadStatistic()

```
UInt64 ThreadStatsManager::getThreadStatistic (
            thread_id_t thread_id,
            ThreadStatType type ) [inline]
```

Definition at line 57 of file thread_stats_manager.h.

References m_threads_stats.

Referenced by SchedulerSequential::results_on_file(), and SchedulerSequential::results_on_screen().

6.379.5.4 getThreadStatName()

```
const char* ThreadStatsManager::getThreadStatName (
            ThreadStatType type ) [inline]
```

Definition at line 56 of file thread_stats_manager.h.

References m_thread_stat_callbacks.

Referenced by ThreadStatsManager::ThreadStats::ThreadStats().

6.379.5.5 getThreadStatTypes()

```
const ThreadStatTypeList& ThreadStatsManager::getThreadStatTypes ( ) [inline]
```

Definition at line 55 of file thread_stats_manager.h.

References m_thread_stat_types.

Referenced by ThreadStatsManager::ThreadStats::ThreadStats().

6.379.5.6 hook_pre_stat_write()

```
static  SInt64 ThreadStatsManager::hook_pre_stat_write (
              UInt64 ptr,
              UInt64  )  [inline], [static], [private]
```

Definition at line 92 of file thread_stats_manager.h.

Referenced by ThreadStatsManager().

6.379.5.7 hook_thread_create()

```
static  SInt64 ThreadStatsManager::hook_thread_create (
              UInt64 ptr,
              UInt64 _args )  [inline], [static], [private]
```

Definition at line 94 of file thread_stats_manager.h.

References HooksManager::ThreadCreate::thread_id.

Referenced by ThreadStatsManager().

6.379.5.8 hook_thread_exit()

```
static  SInt64 ThreadStatsManager::hook_thread_exit (
              UInt64 ptr,
              UInt64 _args )  [inline], [static], [private]
```

Definition at line 118 of file thread_stats_manager.h.

References HooksManager::ThreadTime::thread_id, and HooksManager::ThreadTime::time.

Referenced by ThreadStatsManager().

6.379.5.9 hook_thread_resume()

```
static  SInt64 ThreadStatsManager::hook_thread_resume (
              UInt64 ptr,
              UInt64 _args )  [inline], [static], [private]
```

Definition at line 112 of file thread_stats_manager.h.

References HooksManager::ThreadResume::thread_by, HooksManager::ThreadResume::thread_id, and Hooks↩ Manager::ThreadResume::time.

Referenced by ThreadStatsManager().

6.379.5.10 hook_thread_stall()

```
static SInt64 ThreadStatsManager::hook_thread_stall (
              UInt64 ptr,
              UInt64 _args ) [inline], [static], [private]
```

Definition at line 106 of file thread_stats_manager.h.

References HooksManager::ThreadStall::reason, HooksManager::ThreadStall::thread_id, and HooksManager::↩ ThreadStall::time.

Referenced by ThreadStatsManager().

6.379.5.11 hook_thread_start()

```
static SInt64 ThreadStatsManager::hook_thread_start (
              UInt64 ptr,
              UInt64 _args ) [inline], [static], [private]
```

Definition at line 100 of file thread_stats_manager.h.

References HooksManager::ThreadTime::thread_id, and HooksManager::ThreadTime::time.

Referenced by ThreadStatsManager().

6.379.5.12 metricCallback()

```
UInt64 ThreadStatsManager::metricCallback (
              ThreadStatType type,
              thread_id_t thread_id,
              Core * core,
              UInt64 user ) [static], [private]
```

Definition at line 111 of file thread_stats_manager.cc.

References ELAPSED_NONIDLE_TIME, SubsecondTime::getFS(), PerformanceModel::getInstructionCount(), PerformanceModel::getNonIdleElapsedTime(), Core::getPerformanceModel(), INSTRUCTIONS, LOG_PRINT_E↩ RROR, and WAITING_COST.

Referenced by ThreadStatsManager().

6.379.5.13 pre_stat_write()

```
void ThreadStatsManager::pre_stat_write ( ) [private]
```

Definition at line 127 of file thread_stats_manager.cc.

References calculateWaitingCosts(), INVALID_THREAD_ID, m_bottlegraphs, BottleGraphManager::update(), and update().

6.379.5.14 registerThreadStatMetric()

```
ThreadStatsManager::ThreadStatType ThreadStatsManager::registerThreadStatMetric (
            ThreadStatType type,
           const char * name,
            ThreadStatCallback func,
            UInt64 user )
```

Definition at line 38 of file thread_stats_manager.cc.

References DYNAMIC, m_next_dynamic_type, m_thread_stat_callbacks, and m_thread_stat_types.

Referenced by ThreadStatsManager().

6.379.5.15 threadCreate()

```
void ThreadStatsManager::threadCreate (
            thread_id_t thread_id ) [private]
```

Definition at line 135 of file thread_stats_manager.cc.

References LOG_ASSERT_ERROR, m_threads_stats, and MAX_THREADS.

6.379.5.16 threadExit()

```
void ThreadStatsManager::threadExit (
            thread_id_t thread_id,
            SubsecondTime time ) [private]
```

Definition at line 163 of file thread_stats_manager.cc.

References calculateWaitingCosts(), m_bottlegraphs, m_threads_stats, and BottleGraphManager::update().

6.379.5.17 threadResume()

```
void ThreadStatsManager::threadResume (
            thread_id_t thread_id,
            thread_id_t thread_by,
            SubsecondTime time ) [private]
```

Definition at line 156 of file thread_stats_manager.cc.

References calculateWaitingCosts(), m_bottlegraphs, m_threads_stats, and BottleGraphManager::update().

6.379.5.18 threadStall()

```
void ThreadStatsManager::threadStall (
              thread_id_t thread_id,
              ThreadManager::stall_type_t reason,
              SubsecondTime time )  [private]
```

Definition at line 148 of file thread_stats_manager.cc.

References calculateWaitingCosts(), m_bottlegraphs, m_threads_stats, ThreadManager::STALL_UNSCHEDUL↩ ED, and BottleGraphManager::update().

6.379.5.19 threadStart()

```
void ThreadStatsManager::threadStart (
              thread_id_t thread_id,
              SubsecondTime time )  [private]
```

Definition at line 141 of file thread_stats_manager.cc.

References m_bottlegraphs, m_threads_stats, BottleGraphManager::threadStart(), and BottleGraphManager↩ ::update().

6.379.5.20 update()

```
void ThreadStatsManager::update (
              thread_id_t thread_id = INVALID_THREAD_ID,
              SubsecondTime time = SubsecondTime::MaxTime() )
```

Definition at line 55 of file thread_stats_manager.cc.

References calculateWaitingCosts(), INVALID_THREAD_ID, m_threads_stats, and SubsecondTime::MaxTime().

Referenced by pre_stat_write().

6.379.6 Member Data Documentation

6.379.6.1 m_bottlegraphs

```
BottleGraphManager ThreadStatsManager::m_bottlegraphs  [private]
```

Definition at line 78 of file thread_stats_manager.h.

Referenced by pre_stat_write(), threadExit(), threadResume(), threadStall(), and threadStart().

6.379.6.2 m_next_dynamic_type

ThreadStatType ThreadStatsManager::m_next_dynamic_type [private]

Definition at line 77 of file thread_stats_manager.h.

Referenced by registerThreadStatMetric().

6.379.6.3 m_thread_stat_callbacks

std::unordered_map< **ThreadStatType**, **StatCallback**> ThreadStatsManager::m_thread_stat_callbacks [private]

Definition at line 76 of file thread_stats_manager.h.

Referenced by callThreadStatCallback(), getThreadStatName(), and registerThreadStatMetric().

6.379.6.4 m_thread_stat_types

ThreadStatTypeList ThreadStatsManager::m_thread_stat_types [private]

Definition at line 75 of file thread_stats_manager.h.

Referenced by getThreadStatTypes(), and registerThreadStatMetric().

6.379.6.5 m_threads_stats

std::vector< **ThreadStats***> ThreadStatsManager::m_threads_stats [private]

Definition at line 74 of file thread_stats_manager.h.

Referenced by calculateWaitingCosts(), getThreadStatistic(), threadCreate(), threadExit(), threadResume(), threadStall(), threadStart(), update(), and ~ThreadStatsManager().

6.379.6.6 m_waiting_time_last

SubsecondTime ThreadStatsManager::m_waiting_time_last [private]

Definition at line 79 of file thread_stats_manager.h.

Referenced by calculateWaitingCosts().

6.379.6.7 MAX_THREADS

```
const int ThreadStatsManager::MAX_THREADS = 4096  [static], [private]
```

Definition at line 73 of file thread_stats_manager.h.

Referenced by threadCreate().

The documentation for this class was generated from the following files:

- common/system/ **thread_stats_manager.h**
- common/system/ **thread_stats_manager.cc**

6.380 HooksManager::ThreadTime Struct Reference

```
#include <hooks_manager.h>
```

Public Attributes

- **thread_id_t thread_id**
- **subsecond_time_t time**

6.380.1 Detailed Description

Definition at line 77 of file hooks_manager.h.

6.380.2 Member Data Documentation

6.380.2.1 thread_id

```
thread_id_t HooksManager::ThreadTime::thread_id
```

Definition at line 78 of file hooks_manager.h.

Referenced by SchedulerDynamic::hook_thread_exit(), ThreadStatsManager::hook_thread_exit(), SchedulerDynamic::hook_thread_start(), ThreadStatsManager::hook_thread_start(), hookCallbackThreadTimeType(), BarrierSyncServer::threadExit(), SmtTimer::threadExit(), and SmtTimer::threadStart().

6.380.2.2 time

```
subsecond_time_t HooksManager::ThreadTime::time
```

Definition at line 79 of file hooks_manager.h.

Referenced by SchedulerDynamic::hook_thread_exit(), ThreadStatsManager::hook_thread_exit(), SchedulerDynamic::hook_thread_start(), ThreadStatsManager::hook_thread_start(), and hookCallbackThreadTimeType().

The documentation for this struct was generated from the following file:

- common/system/ **hooks_manager.h**

6.381 TimeConverter< T > Struct Template Reference

```
#include <subsecond_time.h>
```

Static Public Member Functions

- static T **PStoFS** (T ps)
- static T **NStoPS** (T ns)
- static T **NStoFS** (T ns)
- static T **UStoNS** (T us)

6.381.1 Detailed Description

template<typename T>
struct TimeConverter< T >

Definition at line 19 of file subsecond_time.h.

6.381.2 Member Function Documentation

6.381.2.1 NStoFS()

```
template<typename T >
static T  TimeConverter< T >::NStoFS (
            T ns )  [inline], [static]
```

Definition at line 31 of file subsecond_time.h.

References TimeConverter< T >::NStoPS(), and TimeConverter< T >::PStoFS().

Referenced by DramPerfModelConstant::DramPerfModelConstant(), DramPerfModelNormal::DramPerfModelNormal(), and DramPerfModelReadWrite::DramPerfModelReadWrite().

6.381.2.2 NStoPS()

```
template<typename T >
static T  TimeConverter< T >::NStoPS (
            T ns )  [inline], [static]
```

Definition at line 26 of file subsecond_time.h.

Referenced by TimeConverter< T >::NStoFS().

6.381.2.3 PStoFS()

```
template<typename T >
static T  TimeConverter< T >::PStoFS (
            T ps )  [inline], [static]
```

Definition at line 21 of file subsecond_time.h.

Referenced by TimeConverter< T >::NStoFS().

6.381.2.4 UStoNS()

```
template<typename T >
static T  TimeConverter< T >::UStoNS (
            T us )  [inline], [static]
```

Definition at line 36 of file subsecond_time.h.

The documentation for this struct was generated from the following file:

- common/misc/ **subsecond_time.h**

6.382 TimeDistribution Class Reference

```
#include <distribution.h>
```

Inheritance diagram for TimeDistribution:

Public Member Functions

- virtual ~**TimeDistribution** ()
- virtual **SubsecondTime** **next** ()=0

6.382.1 Detailed Description

Definition at line 35 of file distribution.h.

6.382.2 Constructor & Destructor Documentation

6.382.2.1 ~TimeDistribution()

```
virtual TimeDistribution::~TimeDistribution ( )  [inline], [virtual]
```

Definition at line 38 of file distribution.h.

6.382.3 Member Function Documentation

6.382.3.1 next()

```
virtual  SubsecondTime TimeDistribution::next ( )  [pure virtual]
```

Implemented in **NormalTimeDistribution** (p. 877), and **ConstantTimeDistribution** (p. 369).

Referenced by DramPerfModelNormal::getAccessLatency().

The documentation for this class was generated from the following file:

- common/misc/ **distribution.h**

6.383 Timer Class Reference

```
#include <timer.h>
```

Public Member Functions

- **Timer** ()
- ~**Timer** ()
- void **start** (void)
- **UInt64** **getTime** (void)

Static Public Member Functions

- static **UInt64 now** (void)

Public Attributes

- bool **switched**

Private Types

- typedef **TFixedPoint**< 0x100 > **RdtscSpeed**

Private Attributes

- **UInt64 t_start**
- **UInt64 r_start**
- **UInt32 cpu_start**

Static Private Attributes

- static **RdtscSpeed rdtsc_speed** = 0

6.383.1 Detailed Description

Definition at line 11 of file timer.h.

6.383.2 Member Typedef Documentation

6.383.2.1 RdtscSpeed

`typedef` **`TFixedPoint`**`<0x100>` **`Timer::RdtscSpeed`** `[private]`

Definition at line 31 of file timer.h.

6.383.3 Constructor & Destructor Documentation

6.383.3.1 Timer()

```
Timer::Timer ( )
```

Definition at line 41 of file timer.cc.

References now(), r_start, rdtsc_and_cpuid(), rdtsc_speed, start(), and t_start.

6.383.3.2 ~Timer()

```
Timer::~Timer ( )
```

Definition at line 55 of file timer.cc.

6.383.4 Member Function Documentation

6.383.4.1 getTime()

```
UInt64 Timer::getTime (
            void  )
```

Return elapsed time in nanoseconds

Definition at line 77 of file timer.cc.

References cpu_start, TFixedPoint< one >::floor(), now(), r_start, rdtsc(), rdtsc_and_cpuid(), rdtsc_speed, switched, and t_start.

Referenced by MagicServer::disablePerformance(), MagicServer::setPerformance(), and ScopedTimer::~ScopedTimer().

6.383.4.2 now()

```
UInt64 Timer::now (
            void  )  [static]
```

Definition at line 59 of file timer.cc.

Referenced by getTime(), Core::initiateMemoryAccess(), SamplingManager::periodic(), start(), and Timer().

6.383.4.3 start()

```
void Timer::start (
            void  )
```

Start timing

Definition at line 66 of file timer.cc.

References cpu_start, now(), r_start, rdtsc(), rdtsc_and_cpuid(), switched, and t_start.

Referenced by MagicServer::enablePerformance(), MagicServer::setPerformance(), and Timer().

6.383.5 Member Data Documentation

6.383.5.1 cpu_start

```
UInt32 Timer::cpu_start  [private]
```

Definition at line 28 of file timer.h.

Referenced by getTime(), and start().

6.383.5.2 r_start

```
UInt64 Timer::r_start  [private]
```

Definition at line 27 of file timer.h.

Referenced by getTime(), start(), and Timer().

6.383.5.3 rdtsc_speed

```
Timer::RdtscSpeed Timer::rdtsc_speed = 0  [static], [private]
```

Definition at line 32 of file timer.h.

Referenced by getTime(), and Timer().

6.383.5.4 switched

```
bool Timer::switched
```

Definition at line 14 of file timer.h.

Referenced by getTime(), start(), and ScopedTimer::~ScopedTimer().

6.383.5.5 t_start

```
UInt64 Timer::t_start  [private]
```

Definition at line 26 of file timer.h.

Referenced by getTime(), start(), and Timer().

The documentation for this class was generated from the following files:

- common/misc/ **timer.h**
- common/misc/ **timer.cc**

6.384 ParametricDramDirectoryMSI::TLB Class Reference

```
#include <tlb.h>
```

Public Member Functions

- **TLB** (String name, String cfgname, **core_id_t** core_id, **UInt32** num_entries, **UInt32** associativity, **TLB** *next_level)
- bool **lookup** (**IntPtr** address, **SubsecondTime** now, bool allocate_on_miss=true)
- void **allocate** (**IntPtr** address, **SubsecondTime** now)

Private Attributes

- **UInt32 m_size**
- **UInt32 m_associativity**
- **Cache m_cache**
- **TLB** * **m_next_level**
- **UInt64 m_access**
- **UInt64 m_miss**

Static Private Attributes

- static const **UInt32 SIM_PAGE_SHIFT** = 12
- static const **IntPtr SIM_PAGE_SIZE** = (1L << **SIM_PAGE_SHIFT**)
- static const **IntPtr SIM_PAGE_MASK** = ~(**SIM_PAGE_SIZE** - 1)

6.384.1 Detailed Description

Definition at line 9 of file tlb.h.

6.384.2 Constructor & Destructor Documentation

6.384.2.1 TLB()

```
ParametricDramDirectoryMSI::TLB::TLB (
             String name,
             String cfgname,
              core_id_t core_id,
              UInt32 num_entries,
              UInt32 associativity,
              TLB * next_level )
```

Definition at line 7 of file tlb.cc.

References LOG_ASSERT_ERROR, m_access, m_miss, and registerStatsMetric().

6.384.3 Member Function Documentation

6.384.3.1 allocate()

```
void ParametricDramDirectoryMSI::TLB::allocate (
              IntPtr address,
              SubsecondTime now )
```

Definition at line 47 of file tlb.cc.

References allocate(), Cache::insertSingleLine(), m_cache, and m_next_level.

Referenced by allocate(), and lookup().

6.384.3.2 lookup()

```
bool ParametricDramDirectoryMSI::TLB::lookup (
              IntPtr address,
              SubsecondTime now,
             bool allocate_on_miss = true )
```

Definition at line 22 of file tlb.cc.

References Cache::accessSingleLine(), allocate(), CacheBase::LOAD, lookup(), m_access, m_cache, m_miss, and m_next_level.

Referenced by ParametricDramDirectoryMSI::MemoryManager::accessTLB(), and lookup().

6.384.4 Member Data Documentation

6.384.4.1 m_access

UInt64 ParametricDramDirectoryMSI::TLB::m_access [private]

Definition at line 22 of file tlb.h.

Referenced by lookup(), and TLB().

6.384.4.2 m_associativity

UInt32 ParametricDramDirectoryMSI::TLB::m_associativity [private]

Definition at line 17 of file tlb.h.

6.384.4.3 m_cache

Cache ParametricDramDirectoryMSI::TLB::m_cache [private]

Definition at line 18 of file tlb.h.

Referenced by allocate(), and lookup().

6.384.4.4 m_miss

UInt64 ParametricDramDirectoryMSI::TLB::m_miss [private]

Definition at line 22 of file tlb.h.

Referenced by lookup(), and TLB().

6.384.4.5 m_next_level

TLB* ParametricDramDirectoryMSI::TLB::m_next_level [private]

Definition at line 20 of file tlb.h.

Referenced by allocate(), and lookup().

6.384.4.6 m_size

UInt32 ParametricDramDirectoryMSI::TLB::m_size [private]

Definition at line 16 of file tlb.h.

6.384.4.7 SIM_PAGE_MASK

const **IntPtr** ParametricDramDirectoryMSI::TLB::SIM_PAGE_MASK = ~(**SIM_PAGE_SIZE** - 1) [static], [private]

Definition at line 14 of file tlb.h.

6.384.4.8 SIM_PAGE_SHIFT

const **UInt32** ParametricDramDirectoryMSI::TLB::SIM_PAGE_SHIFT = 12 [static], [private]

Definition at line 12 of file tlb.h.

6.384.4.9 SIM_PAGE_SIZE

const **IntPtr** ParametricDramDirectoryMSI::TLB::SIM_PAGE_SIZE = (1L << **SIM_PAGE_SHIFT**) [static], [private]

Definition at line 13 of file tlb.h.

The documentation for this class was generated from the following files:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **tlb.h**
- common/core/memory_subsystem/parametric_dram_directory_msi/ **tlb.cc**

6.385 TLBMissInstruction Class Reference

#include <instruction.h>

Inheritance diagram for TLBMissInstruction:

Public Member Functions

- **TLBMissInstruction** (**SubsecondTime** cost, bool is_ifetch)
- bool **isIfetch** () const

Private Attributes

- bool **m_is_ifetch**

Additional Inherited Members

6.385.1 Detailed Description

Definition at line 191 of file instruction.h.

6.385.2 Constructor & Destructor Documentation

6.385.2.1 TLBMissInstruction()

```
TLBMissInstruction::TLBMissInstruction (
              SubsecondTime cost,
             bool is_ifetch )  [inline]
```

Definition at line 195 of file instruction.h.

6.385.3 Member Function Documentation

6.385.3.1 isIfetch()

```
bool TLBMissInstruction::isIfetch ( ) const  [inline]
```

Definition at line 199 of file instruction.h.

References m_is_ifetch.

Referenced by MicroOpPerformanceModel::handleInstruction().

6.385.4 Member Data Documentation

6.385.4.1 m_is_ifetch

```
bool TLBMissInstruction::m_is_ifetch  [private]
```

Definition at line 193 of file instruction.h.

Referenced by isIfetch().

The documentation for this class was generated from the following file:

- common/performance_model/ **instruction.h**

6.386 TLock< T_LockCreator > Class Template Reference

```
#include <lock.h>
```

Inheritance diagram for TLock< T_LockCreator >:

Public Member Functions

- **TLock** ()
- ~**TLock** ()
- void **acquire** ()
- void **acquire_read** ()
- void **release** ()
- void **release_read** ()

Private Attributes

- **LockImplementation** * **_lock**

6.386.1 Detailed Description

template<class T_LockCreator>
class TLock< T_LockCreator >

Definition at line 45 of file lock.h.

6.386.2 Constructor & Destructor Documentation

6.386.2.1 TLock()

```
template<class T_LockCreator >
TLock< T_LockCreator >::  TLock ( )  [inline]
```

Definition at line 48 of file lock.h.

6.386.2.2 ~TLock()

```
template<class T_LockCreator >
TLock< T_LockCreator >::~ TLock ( )  [inline]
```

Definition at line 56 of file lock.h.

6.386.3 Member Function Documentation

6.386.3.1 acquire()

```
template<class T_LockCreator >
void  TLock< T_LockCreator >::acquire ( )  [inline], [virtual]
```

Implements **BaseLock** (p. 103).

Definition at line 64 of file lock.h.

Referenced by CheetahModel::accesses(), ConditionVariable::broadcast(), Semaphore::broadcast(), LockedHash::find(), lite::handleMemoryReadFaultinjection(), lite::handleMemoryReadFaultinjectionNondetailed(), ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDramDirectory(), Core::initiateMemoryAccess(), LockedHash::insert(), Log::log(), Core::nativeMemOp(), Network::netPullFromTransport(), Network::netRecv(), LockedHash::remove(), SmTransport::SmNode::send(), ConditionVariable::signal(), Semaphore::signal(), SimThreadManager::simThreadExitCallback(), SimThreadManager::simThreadStartCallback(), SmtTimer::simulate(), Barrier::wait(), ConditionVariable::wait(), and Semaphore::wait().

6.386.3.2 acquire_read()

```
template<class T_LockCreator >
void  TLock< T_LockCreator >::acquire_read ( )  [inline], [virtual]
```

Implements **BaseLock** (p. 104).

Definition at line 72 of file lock.h.

6.386.3.3 release()

```
template<class T_LockCreator >
void TLock< T_LockCreator >::release ( ) [inline], [virtual]
```

Implements **BaseLock** (p. 104).

Definition at line 80 of file lock.h.

Referenced by CheetahModel::accesses(), ConditionVariable::broadcast(), Semaphore::broadcast(), lite↩ ::completeMemoryWrite(), LockedHash::find(), ParametricDramDirectoryMSI::CacheCntlr::handleMsgFromDram↩ Directory(), Core::initiateMemoryAccess(), LockedHash::insert(), Log::log(), Core::nativeMemOp(), Network↩ ::netPullFromTransport(), Network::netRecv(), LockedHash::remove(), SmTransport::SmNode::send(), Condition↩ Variable::signal(), Semaphore::signal(), SimThreadManager::simThreadExitCallback(), SimThreadManager::sim↩ ThreadStartCallback(), SmtTimer::simulate(), Barrier::wait(), ConditionVariable::wait(), and Semaphore::wait().

6.386.3.4 release_read()

```
template<class T_LockCreator >
void TLock< T_LockCreator >::release_read ( ) [inline], [virtual]
```

Implements **BaseLock** (p. 104).

Definition at line 85 of file lock.h.

6.386.4 Member Data Documentation

6.386.4.1 _lock

```
template<class T_LockCreator >
LockImplementation* TLock< T_LockCreator >::_lock [private]
```

Definition at line 91 of file lock.h.

Referenced by TLock< LockCreator_Default >::acquire(), TLock< LockCreator_Default >::acquire_read(), T↩ Lock< LockCreator_Default >::release(), TLock< LockCreator_Default >::release_read(), TLock< LockCreator↩ _Default >::TLock(), and TLock< LockCreator_Default >::~TLock().

The documentation for this class was generated from the following file:

- common/misc/ **lock.h**

6.387 TLS Class Reference

`#include <tls.h>`

Inheritance diagram for TLS:

Public Member Functions

- virtual ~**TLS** ()
- virtual void * **get** (int thread_id=-1)=0
- virtual const void * **get** (int thread_id=-1) const =0
- template<class T >
  T & **get** (int thread_id=-1)
- template<class T >
  const T & **get** (int thread_id=-1) const
- template<class T >
  T * **getPtr** (int thread_id=-1)
- template<class T >
  const T * **getPtr** (int thread_id=-1) const
- **IntPtr** **getInt** (int thread_id=-1) const
- virtual void **set** (void *)=0
- template<class T >
  void **set** (T *p)
- void **setInt** (**IntPtr** i)

Static Public Member Functions

- static **TLS** * **create** ()

Protected Member Functions

- **TLS** ()

6.387.1 Detailed Description

Definition at line 6 of file tls.h.

6.387.2 Constructor & Destructor Documentation

6.387.2.1 ~TLS()

```
TLS::~TLS ( )  [virtual]
```

Definition at line 6 of file tls.cc.

6.387.2.2 TLS()

```
TLS::TLS ( )  [protected]
```

Definition at line 3 of file tls.cc.

6.387.3 Member Function Documentation

6.387.3.1 create()

```
TLS * TLS::create ( )  [static]
```

Definition at line 44 of file pin_tls.cc.

6.387.3.2 get() [1/4]

```
template<class T >
T& TLS::get (
            int thread_id = -1 )  [inline]
```

Definition at line 15 of file tls.h.

References get().

Referenced by get().

6.387.3.3 get() [2/4]

```
template<class T >
const T& TLS::get (
            int thread_id = -1 ) const  [inline]
```

Definition at line 18 of file tls.h.

References get().

Referenced by get().

6.387.3.4 get() [3/4]

```
virtual const void* TLS::get (
            int thread_id = -1 ) const  [pure virtual]
```

Implemented in **PthreadTLS** (p. 957), and **PinTLS** (p. 934).

6.387.3.5 get() [4/4]

```
virtual void* TLS::get (
            int thread_id = -1 )  [pure virtual]
```

Implemented in **PthreadTLS** (p. 956), and **PinTLS** (p. 934).

Referenced by getInt(), getPtr(), CoreManager::initializeThread(), and CoreManager::terminateThread().

6.387.3.6 getInt()

```
IntPtr TLS::getInt (
            int thread_id = -1 ) const  [inline]
```

Definition at line 26 of file tls.h.

References get().

Referenced by CoreManager::amiCoreThread(), CoreManager::amiSimThread(), and CoreManager::amiUser↩Thread().

6.387.3.7 getPtr() [1/2]

```
template<class T >
T* TLS::getPtr (
            int thread_id = -1 )  [inline]
```

Definition at line 21 of file tls.h.

References get().

Referenced by CoreManager::getCurrentCore(), and ThreadManager::getCurrentThread().

6.387.3.8 getPtr() [2/2]

```
template<class T >
const T* TLS::getPtr (
            int thread_id = -1 ) const  [inline]
```

Definition at line 24 of file tls.h.

References get().

6.387.3.9 set() [1/2]

```
template<class T >
void TLS::set (
            T * p )  [inline]
```

Definition at line 31 of file tls.h.

References set().

Referenced by set().

6.387.3.10 set() [2/2]

```
virtual void TLS::set (
            void *  )  [pure virtual]
```

Implemented in **PinTLS** (p. 934), and **PthreadTLS** (p. 957).

Referenced by CoreManager::initializeThread(), ThreadManager::onThreadExit(), ThreadManager::onThread↩Start(), CoreManager::registerSimThread(), setInt(), and CoreManager::terminateThread().

6.387.3.11 setInt()

```
void TLS::setInt (
             IntPtr i )  [inline]
```

Definition at line 33 of file tls.h.

References set().

Referenced by CoreManager::initializeThread(), and CoreManager::registerSimThread().

The documentation for this class was generated from the following files:

- common/misc/ **tls.h**
- common/misc/ **tls.cc**
- pin/ **pin_tls.cc**

6.388 Tools Class Reference

`#include <tools.h>`

Static Public Member Functions

- static bool **contains** (const uint64_t *array, const int length, const uint64_t value)
- static uint64_t **index** (const uint64_t *array, const int length, const uint64_t value)
- static void **swap** (uint64_t *array, const int idx1, const int idx2)

6.388.1 Detailed Description

Definition at line 12 of file tools.h.

6.388.2 Member Function Documentation

6.388.2.1 contains()

```
static bool Tools::contains (
            const uint64_t * array,
            const int length,
            const uint64_t value )  [inline], [static]
```

Definition at line 20 of file tools.h.

Referenced by DynamicMicroOp::addDependency().

6.388.2.2 index()

```
static uint64_t Tools::index (
            const uint64_t * array,
            const int length,
            const uint64_t value )  [inline], [static]
```

Definition at line 27 of file tools.h.

Referenced by DynamicMicroOp::removeDependency().

6.388.2.3 swap()

```
static void Tools::swap (
             uint64_t * array,
             const int idx1,
             const int idx2 )  [inline], [static]
```

Definition at line 34 of file tools.h.

Referenced by DynamicMicroOp::removeDependency().

The documentation for this class was generated from the following file:

- common/performance_model/performance_models/interval_performance_model/ **tools.h**

6.389 TopologyInfo Class Reference

```
#include <topology_info.h>
```

Public Member Functions

- **TopologyInfo** (**core_id_t** _core_id)
- void **setup** (**UInt32** smt_cores, **UInt32** llc_sharers)

Public Attributes

- **core_id_t core_id**
- **SInt32 apic_id**
- **SInt32 smt_index**
- **SInt32 smt_count**
- **SInt32 core_index**
- **SInt32 core_count**
- **SInt32 package**

Static Public Attributes

- static const **UInt32 SMT_SHIFT_BITS** = 4
- static const **UInt32 PACKAGE_SHIFT_BITS** = 16
- static **SInt32 s_package** = -1
- static **SInt32 s_cores_this_package** = 0
- static **SInt32 s_core_id_last** = -1

6.389.1 Detailed Description

Definition at line 6 of file topology_info.h.

6.389.2 Constructor & Destructor Documentation

6.389.2.1 TopologyInfo()

```
TopologyInfo::TopologyInfo (
              core_id_t _core_id )  [inline]
```

Definition at line 9 of file topology_info.h.

6.389.3 Member Function Documentation

6.389.3.1 setup()

```
void TopologyInfo::setup (
              UInt32 smt_cores,
              UInt32 llc_sharers )
```

Definition at line 11 of file topology_info.cc.

References apic_id, core_count, core_id, core_index, LOG_ASSERT_ERROR, package, PACKAGE_SHIFT_BITS, s_core_id_last, s_cores_this_package, s_package, smt_count, smt_index, and SMT_SHIFT_BITS.

Referenced by ParametricDramDirectoryMSI::MemoryManager::MemoryManager().

6.389.4 Member Data Documentation

6.389.4.1 apic_id

```
SInt32 TopologyInfo::apic_id
```

Definition at line 24 of file topology_info.h.

Referenced by Core::emulateCpuid(), and setup().

6.389.4.2 core_count

```
SInt32 TopologyInfo::core_count
```

Definition at line 26 of file topology_info.h.

Referenced by Core::emulateCpuid(), and setup().

6.389.4.3 core_id

```
core_id_t TopologyInfo::core_id
```

Definition at line 23 of file topology_info.h.

Referenced by setup().

6.389.4.4 core_index

```
SInt32 TopologyInfo::core_index
```

Definition at line 26 of file topology_info.h.

Referenced by setup().

6.389.4.5 package

```
SInt32 TopologyInfo::package
```

Definition at line 27 of file topology_info.h.

Referenced by setup().

6.389.4.6 PACKAGE_SHIFT_BITS

```
const UInt32 TopologyInfo::PACKAGE_SHIFT_BITS = 16 [static]
```

Definition at line 21 of file topology_info.h.

Referenced by Core::emulateCpuid(), and setup().

6.389.4.7 s_core_id_last

```
SInt32 TopologyInfo::s_core_id_last = -1 [static]
```

Definition at line 29 of file topology_info.h.

Referenced by setup().

6.389.4.8 s_cores_this_package

```
SInt32 TopologyInfo::s_cores_this_package = 0  [static]
```

Definition at line 29 of file topology_info.h.

Referenced by setup().

6.389.4.9 s_package

```
SInt32 TopologyInfo::s_package = -1  [static]
```

Definition at line 29 of file topology_info.h.

Referenced by setup().

6.389.4.10 smt_count

```
SInt32 TopologyInfo::smt_count
```

Definition at line 25 of file topology_info.h.

Referenced by Core::emulateCpuid(), and setup().

6.389.4.11 smt_index

```
SInt32 TopologyInfo::smt_index
```

Definition at line 25 of file topology_info.h.

Referenced by setup().

6.389.4.12 SMT_SHIFT_BITS

```
const  UInt32 TopologyInfo::SMT_SHIFT_BITS = 4  [static]
```

Definition at line 20 of file topology_info.h.

Referenced by Core::emulateCpuid(), and setup().

The documentation for this class was generated from the following files:

- common/core/ **topology_info.h**
- common/core/ **topology_info.cc**

6.390 TotalTimer Class Reference

`#include <timer.h>`

Public Member Functions

- **TotalTimer** (String _name, **UInt32** _ignore=0)
- ~**TotalTimer** ()
- void **add** (**UInt64** t, bool switched=false)
- void **report** (FILE *fp)

Static Public Member Functions

- static void **reports** (void)
- static **TotalTimer** * **getTimerByStacktrace** (String **name**)

Private Attributes

- String **name**
- **UInt64 total**
- **UInt64 n**
- **UInt64 max**
- **UInt64 n_switched**
- void * **backtrace_buffer** [**BACKTRACE_SIZE**]
- unsigned long **backtrace_n**
- unsigned long **backtrace_ignore**

Static Private Attributes

- static const unsigned long **BACKTRACE_SIZE** = 8

6.390.1 Detailed Description

Definition at line 35 of file timer.h.

6.390.2 Constructor & Destructor Documentation

6.390.2.1 TotalTimer()

```
TotalTimer::TotalTimer (
            String _name,
             UInt32 _ignore = 0 )
```

Definition at line 101 of file timer.cc.

References alltimers, backtrace_buffer, backtrace_n, BACKTRACE_SIZE, MAX_TIMERS, and numtimers.

Referenced by getTimerByStacktrace().

6.390.2.2 ~TotalTimer()

```
TotalTimer::~TotalTimer ( )
```

Definition at line 109 of file timer.cc.

6.390.3 Member Function Documentation

6.390.3.1 add()

```
void TotalTimer::add (
             UInt64 t,
            bool switched = false )  [inline]
```

Definition at line 41 of file timer.h.

References max, n, n_switched, and total.

Referenced by ScopedTimer::~ScopedTimer().

6.390.3.2 getTimerByStacktrace()

```
TotalTimer * TotalTimer::getTimerByStacktrace (
            String name )  [static]
```

Definition at line 150 of file timer.cc.

References getHashByStacktrace(), name, TotalTimer(), and totaltimershash.

Referenced by _SetLock::_SetLock(), ConditionVariable::ConditionVariable(), SELock::SELock(), and TLock< LockCreator_Default >::TLock().

6.390.3.3 report()

```
void TotalTimer::report (
            FILE * fp )
```

Definition at line 113 of file timer.cc.

References backtrace_buffer, backtrace_ignore, backtrace_n, max, n, n_switched, name, and total.

Referenced by reports().

6.390.3.4 reports()

```
void TotalTimer::reports (
            void  )  [static]
```

Definition at line 123 of file timer.cc.

References alltimers, numtimers, rdtsc(), and report().

Referenced by Simulator::∼Simulator().

6.390.4 Member Data Documentation

6.390.4.1 backtrace_buffer

```
void* TotalTimer::backtrace_buffer[ BACKTRACE_SIZE]  [private]
```

Definition at line 62 of file timer.h.

Referenced by report(), and TotalTimer().

6.390.4.2 backtrace_ignore

```
unsigned long TotalTimer::backtrace_ignore  [private]
```

Definition at line 63 of file timer.h.

Referenced by report().

6.390.4.3 backtrace_n

```
unsigned long TotalTimer::backtrace_n  [private]
```

Definition at line 63 of file timer.h.

Referenced by report(), and TotalTimer().

6.390.4.4 BACKTRACE_SIZE

```
const unsigned long TotalTimer::BACKTRACE_SIZE = 8  [static], [private]
```

Definition at line 61 of file timer.h.

Referenced by TotalTimer().

6.390.4.5 max

```
UInt64 TotalTimer::max  [private]
```

Definition at line 58 of file timer.h.

Referenced by add(), and report().

6.390.4.6 n

```
UInt64 TotalTimer::n  [private]
```

Definition at line 57 of file timer.h.

Referenced by add(), and report().

6.390.4.7 n_switched

```
UInt64 TotalTimer::n_switched  [private]
```

Definition at line 59 of file timer.h.

Referenced by add(), and report().

6.390.4.8 name

```
String TotalTimer::name  [private]
```

Definition at line 55 of file timer.h.

Referenced by getTimerByStacktrace(), and report().

6.390.4.9 total

```
UInt64 TotalTimer::total  [private]
```

Definition at line 56 of file timer.h.

Referenced by add(), and report().

The documentation for this class was generated from the following files:

- common/misc/ **timer.h**
- common/misc/ **timer.cc**

6.391 TraceManager Class Reference

`#include <trace_manager.h>`

Classes

- struct **app_info_t**
- class **Monitor**

Public Member Functions

- **TraceManager** ()
- ∼**TraceManager** ()
- void **init** ()
- void **start** ()
- void **stop** ()
- void **mark_done** ()
- void **wait** ()
- void **run** ()
- void **cleanup** ()
- void **setupTraceFiles** (int index)
- **thread_id_t** **createThread** (**app_id_t** app_id, **SubsecondTime** time, **thread_id_t** creator_thread_id)
- **app_id_t** **createApplication** (**SubsecondTime** time, **thread_id_t** creator_thread_id)
- void **signalStarted** ()
- void **signalDone** (**TraceThread** ∗thread, **SubsecondTime** time, bool aborted)
- void **endApplication** (**TraceThread** ∗thread, **SubsecondTime** time)
- void **accessMemory** (int core_id, **Core::lock_signal_t** lock_signal, **Core::mem_op_t** mem_op_type, **IntPtr** d_addr, char ∗data_buffer, **UInt32** data_size)
- **UInt64** **getProgressExpect** ()
- **UInt64** **getProgressValue** ()

Private Member Functions

- String **getFifoName** (**app_id_t** app_id, **UInt64** thread_num, bool response, bool create)
- **thread_id_t** **newThread** (**app_id_t** app_id, bool first, bool init_fifo, bool spawn, **SubsecondTime** time, **thread_id_t** creator_thread_id)

Private Attributes

- **Monitor** * **m_monitor**
- std::vector< **TraceThread** * > **m_threads**
- **UInt32** **m_num_threads_started**
- **UInt32** **m_num_threads_running**
- **Semaphore** **m_done**
- const bool **m_stop_with_first_app**
- const bool **m_app_restart**
- const bool **m_emulate_syscalls**
- **UInt32** **m_num_apps**
- **UInt32** **m_num_apps_nonfinish**
- std::vector< **app_info_t** > **m_app_info**
- std::vector< String > **m_tracefiles**
- std::vector< String > **m_responsefiles**
- String **m_trace_prefix**
- **Lock** **m_lock**

Friends

- class **Monitor**

6.391.1 Detailed Description

Definition at line 13 of file trace_manager.h.

6.391.2 Constructor & Destructor Documentation

6.391.2.1 TraceManager()

```
TraceManager::TraceManager ( )
```

Definition at line 14 of file trace_manager.cc.

References setupTraceFiles().

6.391.2.2 ~TraceManager()

```
TraceManager::~TraceManager ( )
```

Definition at line 236 of file trace_manager.cc.

References cleanup().

6.391.3 Member Function Documentation

6.391.3.1 accessMemory()

```
void TraceManager::accessMemory (
            int core_id,
             Core::lock_signal_t lock_signal,
             Core::mem_op_t mem_op_type,
             IntPtr d_addr,
            char * data_buffer,
             UInt32 data_size )
```

Definition at line 312 of file trace_manager.cc.

References Thread::getCore(), Core::getId(), TraceThread::getThread(), TraceThread::handleAccessMemory(), LOG_PRINT_ERROR, LOG_PRINT_WARNING_ONCE, TraceThread::m_stopped, and m_threads.

6.391.3.2 cleanup()

```
void TraceManager::cleanup ( )
```

Definition at line 224 of file trace_manager.cc.

References m_app_info, m_num_apps, m_num_apps_nonfinish, m_num_threads_running, and m_threads.

Referenced by ~TraceManager().

6.391.3.3 createApplication()

```
app_id_t TraceManager::createApplication (
             SubsecondTime time,
             thread_id_t creator_thread_id )
```

Definition at line 137 of file trace_manager.cc.

References m_app_info, m_lock, m_num_apps, m_num_apps_nonfinish, and newThread().

6.391.3.4 createThread()

```
thread_id_t TraceManager::createThread (
             app_id_t app_id,
             SubsecondTime time,
             thread_id_t creator_thread_id )
```

Definition at line 78 of file trace_manager.cc.

References m_lock, and newThread().

6.391.3.5 endApplication()

```
void TraceManager::endApplication (
              TraceThread * thread,
              SubsecondTime time )
```

Definition at line 209 of file trace_manager.cc.

References Thread::getAppId(), TraceThread::getThread(), m_threads, and signalDone().

6.391.3.6 getFifoName()

```
String TraceManager::getFifoName (
              app_id_t app_id,
              UInt64 thread_num,
             bool response,
             bool create )  [private]
```

Definition at line 70 of file trace_manager.cc.

References itostr(), and m_trace_prefix.

Referenced by newThread(), and setupTraceFiles().

6.391.3.7 getProgressExpect()

```
UInt64 TraceManager::getProgressExpect ( )
```

Definition at line 285 of file trace_manager.cc.

6.391.3.8 getProgressValue()

```
UInt64 TraceManager::getProgressValue ( )
```

Definition at line 290 of file trace_manager.cc.

References m_stop_with_first_app, and m_threads.

6.391.3.9 init()

```
void TraceManager::init ( )
```

Definition at line 62 of file trace_manager.cc.

References INVALID_THREAD_ID, m_num_apps, newThread(), and SubsecondTime::Zero().

Referenced by Simulator::start().

6.391.3.10 mark_done()

```
void TraceManager::mark_done ( )
```

Definition at line 268 of file trace_manager.cc.

References m_trace_prefix.

Referenced by stop().

6.391.3.11 newThread()

```
thread_id_t TraceManager::newThread (
            app_id_t app_id,
            bool first,
            bool init_fifo,
            bool spawn,
            SubsecondTime time,
            thread_id_t creator_thread_id ) [private]
```

Definition at line 86 of file trace_manager.cc.

References StatsManager::EVENT_APP_START, getFifoName(), Thread::getId(), HookType::HOOK_APPLICATION_START, INVALID_CORE_ID, INVALID_THREAD_ID, m_app_info, m_num_apps, m_num_threads_running, m_responsefiles, m_threads, m_tracefiles, SubsecondTime::MaxTime(), and TraceThread::spawn().

Referenced by createApplication(), createThread(), init(), and signalDone().

6.391.3.12 run()

```
void TraceManager::run ( )
```

Definition at line 279 of file trace_manager.cc.

References start(), and wait().

6.391.3.13 setupTraceFiles()

```
void TraceManager::setupTraceFiles (
            int index )
```

Definition at line 32 of file trace_manager.cc.

References getFifoName(), itostr(), m_emulate_syscalls, m_num_apps, m_responsefiles, m_trace_prefix, and m_tracefiles.

Referenced by TraceManager().

6.391.3.14 signalDone()

```
void TraceManager::signalDone (
             TraceThread * thread,
             SubsecondTime time,
            bool aborted )
```

Definition at line 159 of file trace_manager.cc.

References StatsManager::EVENT_APP_EXIT, Thread::getAppId(), TraceThread::getThread(), HookType::HOOK_APPLICATION_EXIT, INVALID_CORE_ID, INVALID_THREAD_ID, m_app_info, m_app_restart, m_lock, m_num_apps_nonfinish, m_num_threads_running, m_stop_with_first_app, TraceThread::m_stopped, SubsecondTime::MaxTime(), newThread(), and stop().

Referenced by endApplication().

6.391.3.15 signalStarted()

```
void TraceManager::signalStarted ( )
```

Definition at line 154 of file trace_manager.cc.

References m_num_threads_started.

6.391.3.16 start()

```
void TraceManager::start ( )
```

Definition at line 241 of file trace_manager.cc.

References m_monitor, m_threads, and TraceManager::Monitor::spawn().

Referenced by run().

6.391.3.17 stop()

```
void TraceManager::stop ( )
```

Definition at line 251 of file trace_manager.cc.

References m_done, m_threads, mark_done(), and Semaphore::signal().

Referenced by signalDone().

6.391.3.18 wait()

```
void TraceManager::wait ( )
```

Definition at line 274 of file trace_manager.cc.

References m_done, and Semaphore::wait().

Referenced by run().

6.391.4 Friends And Related Function Documentation

6.391.4.1 Monitor

```
friend class Monitor [friend]
```

Definition at line 59 of file trace_manager.h.

6.391.5 Member Data Documentation

6.391.5.1 m_app_info

```
std::vector< app_info_t> TraceManager::m_app_info [private]
```

Definition at line 50 of file trace_manager.h.

Referenced by cleanup(), createApplication(), newThread(), and signalDone().

6.391.5.2 m_app_restart

```
const bool TraceManager::m_app_restart  [private]
```

Definition at line 46 of file trace_manager.h.

Referenced by signalDone().

6.391.5.3 m_done

```
Semaphore TraceManager::m_done  [private]
```

Definition at line 44 of file trace_manager.h.

Referenced by stop(), and wait().

6.391.5.4 m_emulate_syscalls

```
const bool TraceManager::m_emulate_syscalls  [private]
```

Definition at line 47 of file trace_manager.h.

Referenced by setupTraceFiles().

6.391.5.5 m_lock

```
Lock TraceManager::m_lock  [private]
```

Definition at line 54 of file trace_manager.h.

Referenced by createApplication(), createThread(), and signalDone().

6.391.5.6 m_monitor

```
Monitor* TraceManager::m_monitor  [private]
```

Definition at line 40 of file trace_manager.h.

Referenced by start().

6.391.5.7 m_num_apps

```
UInt32 TraceManager::m_num_apps  [private]
```

Definition at line 48 of file trace_manager.h.

Referenced by cleanup(), createApplication(), init(), newThread(), and setupTraceFiles().

6.391.5.8 m_num_apps_nonfinish

```
UInt32 TraceManager::m_num_apps_nonfinish  [private]
```

Definition at line 49 of file trace_manager.h.

Referenced by cleanup(), createApplication(), and signalDone().

6.391.5.9 m_num_threads_running

```
UInt32 TraceManager::m_num_threads_running  [private]
```

Definition at line 43 of file trace_manager.h.

Referenced by cleanup(), newThread(), and signalDone().

6.391.5.10 m_num_threads_started

```
UInt32 TraceManager::m_num_threads_started  [private]
```

Definition at line 42 of file trace_manager.h.

Referenced by signalStarted().

6.391.5.11 m_responsefiles

```
std::vector<String> TraceManager::m_responsefiles  [private]
```

Definition at line 52 of file trace_manager.h.

Referenced by newThread(), and setupTraceFiles().

6.391.5.12 m_stop_with_first_app

```
const bool TraceManager::m_stop_with_first_app  [private]
```

Definition at line 45 of file trace_manager.h.

Referenced by getProgressValue(), and signalDone().

6.391.5.13 m_threads

```
std::vector< TraceThread *> TraceManager::m_threads  [private]
```

Definition at line 41 of file trace_manager.h.

Referenced by accessMemory(), cleanup(), endApplication(), getProgressValue(), newThread(), start(), and stop().

6.391.5.14 m_trace_prefix

```
String TraceManager::m_trace_prefix  [private]
```

Definition at line 53 of file trace_manager.h.

Referenced by getFifoName(), mark_done(), and setupTraceFiles().

6.391.5.15 m_tracefiles

```
std::vector<String> TraceManager::m_tracefiles  [private]
```

Definition at line 51 of file trace_manager.h.

Referenced by newThread(), and setupTraceFiles().

The documentation for this class was generated from the following files:

- common/trace_frontend/ **trace_manager.h**
- common/trace_frontend/ **trace_manager.cc**

6.392 TraceThread Class Reference

```
#include <trace_thread.h>
```

Inheritance diagram for TraceThread:

Public Member Functions

- **TraceThread** (**Thread** *thread, **SubsecondTime** time_start, String tracefile, String responsefile, **app_id_t** app_id, bool cleanup)
- ~**TraceThread** ()
- void **spawn** ()
- void **stop** ()
- **UInt64** **getProgressExpect** ()
- **UInt64** **getProgressValue** ()
- **Thread** * **getThread** () const
- void **handleAccessMemory** (**Core::lock_signal_t** lock_signal, **Core::mem_op_t** mem_op_type, **IntPtr** d_addr, char *data_buffer, **UInt32** data_size)

Public Attributes

- bool **m_stopped**

Private Member Functions

- **UInt64** **va2pa** (**UInt64** va, bool *noMapping=NULL)
- **UInt64** **remapAddress** (**UInt64** va_page)
- void **run** ()
- Sift::Mode **handleInstructionCountFunc** (uint32_t icount)
- void **handleCacheOnlyFunc** (uint8_t icount, Sift::CacheOnlyType type, uint64_t eip, uint64_t address)
- void **handleOutputFunc** (uint8_t fd, const uint8_t *data, uint32_t size)
- uint64_t **handleSyscallFunc** (uint16_t syscall_number, const uint8_t *data, uint32_t size)
- int32_t **handleNewThreadFunc** ()
- int32_t **handleForkFunc** ()
- int32_t **handleJoinFunc** (int32_t thread)
- uint64_t **handleMagicFunc** (uint64_t a, uint64_t b, uint64_t c)
- bool **handleEmuFunc** (Sift::EmuType type, Sift::EmuRequest &req, Sift::EmuReply &res)
- void **handleRoutineChangeFunc** (Sift::RoutineOpType event, uint64_t eip, uint64_t esp, uint64_t callEip)
- void **handleRoutineAnnounceFunc** (uint64_t eip, const char *name, const char *imgname, uint64_t offset, uint32_t line, uint32_t column, const char *filename)
- **Instruction** * **decode** (Sift::Instruction &inst)
- void **handleInstructionWarmup** (Sift::Instruction &inst, Sift::Instruction &next_inst, **Core** *core, bool do_icache_warmup, **UInt64** icache_warmup_addr, **UInt64** icache_warmup_size)
- void **handleInstructionDetailed** (Sift::Instruction &inst, Sift::Instruction &next_inst, **PerformanceModel** *prfmdl)
- void **addDetailedMemoryInfo** (**DynamicInstruction** *dynins, Sift::Instruction &inst, const dl::DecodedInst &decoded_inst, uint32_t mem_idx, **Operand::Direction** op_type, bool is_pretetch, **PerformanceModel** *prfmdl)
- void **unblock** ()
- **SubsecondTime** **getCurrentTime** () const
- const dl::DecodedInst * **staticDecode** (Sift::Instruction &inst)

Static Private Member Functions

- static **UInt64** **_va2pa** (**UInt64** self, **UInt64** va)
- static Sift::Mode **__handleInstructionCountFunc** (void *arg, uint32_t icount)
- static void **__handleCacheOnlyFunc** (void *arg, uint8_t icount, Sift::CacheOnlyType type, uint64_t eip, uint64_t address)
- static void **__handleOutputFunc** (void *arg, uint8_t fd, const uint8_t *data, uint32_t size)
- static uint64_t **__handleSyscallFunc** (void *arg, uint16_t syscall_number, const uint8_t *data, uint32_t size)
- static int32_t **__handleNewThreadFunc** (void *arg)
- static int32_t **__handleJoinFunc** (void *arg, int32_t join_thread_id)
- static uint64_t **__handleMagicFunc** (void *arg, uint64_t a, uint64_t b, uint64_t c)
- static bool **__handleEmuFunc** (void *arg, Sift::EmuType type, Sift::EmuRequest &req, Sift::EmuReply &res)
- static void **__handleRoutineChangeFunc** (void *arg, Sift::RoutineOpType event, uint64_t eip, uint64_t esp, uint64_t callEip)
- static void **__handleRoutineAnnounceFunc** (void *arg, uint64_t eip, const char *name, const char *imgname, uint64_t offset, uint32_t line, uint32_t column, const char *filename)
- static int32_t **__handleForkFunc** (void *arg)

Private Attributes

- **_Thread** * **m__thread**
- **Thread** * **m_thread**
- **SubsecondTime** **m_time_start**
- Sift::Reader **m_trace**
- bool **m_trace_has_pa**
- bool **m_address_randomization**
- bool **m_appid_from_coreid**
- uint8_t **m_address_randomization_table** [256]
- bool **m_stop**
- std::unordered_map< **IntPtr**, **Instruction** * > **m_icache**
- std::unordered_map< **IntPtr**, const dl::DecodedInst * > **m_decoder_cache**
- **UInt64** **m_bbv_base**
- **UInt64** **m_bbv_count**
- **UInt64** **m_bbv_last**
- bool **m_bbv_end**
- uint8_t **m_output_leftover** [160]
- uint16_t **m_output_leftover_size**
- String **m_tracefile**
- String **m_responsefile**
- **app_id_t** **m_app_id**
- bool **m_blocked**
- bool **m_cleanup**
- bool **m_started**
- dl::DecoderFactory * **m_factory**
- long long * **m_papi_counters**
- **Lock** **m_lock**

Static Private Attributes

- static const **UInt64** **pa_core_shift** = 48
- static const **UInt64** **pa_core_size** = 16
- static const **UInt64** **pa_va_mask** = ~(((**UInt64**(1) << **pa_core_size**) - 1) << **pa_core_shift**)
- static const **UInt64** **va_page_shift** = 12
- static const **UInt64** **va_page_mask** = (**UInt64**(1) << **va_page_shift**) - 1
- static int **m_isa** = 0

Additional Inherited Members

6.392.1 Detailed Description

Definition at line 32 of file trace_thread.h.

6.392.2 Constructor & Destructor Documentation

6.392.2.1 TraceThread()

```
TraceThread::TraceThread (
              Thread * thread,
              SubsecondTime time_start,
             String tracefile,
             String responsefile,
              app_id_t app_id,
             bool cleanup )
```

Definition at line 37 of file trace_thread.cc.

References __handleCacheOnlyFunc(), __handleEmuFunc(), __handleForkFunc(), __handleInstructionCount↩Func(), __handleJoinFunc(), __handleMagicFunc(), __handleNewThreadFunc(), __handleOutputFunc(), __↩handleRoutineAnnounceFunc(), __handleRoutineChangeFunc(), __handleSyscallFunc(), _va2pa(), m_address_↩randomization, m_address_randomization_table, m_trace, rng_next(), rng_seed(), and Thread::setVa2paFunc().

6.392.2.2 ~TraceThread()

```
TraceThread::~TraceThread ( )
```

Definition at line 101 of file trace_thread.cc.

References m__thread, m_cleanup, m_decoder_cache, m_responsefile, and m_tracefile.

6.392.3 Member Function Documentation

6.392.3.1 __handleCacheOnlyFunc()

```
static void TraceThread::__handleCacheOnlyFunc (
             void * arg,
             uint8_t icount,
             Sift::CacheOnlyType type,
             uint64_t eip,
             uint64_t address )  [inline], [static], [private]
```

Definition at line 85 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.2 __handleEmuFunc()

```
static bool TraceThread::__handleEmuFunc (
            void * arg,
            Sift::EmuType type,
            Sift::EmuRequest & req,
            Sift::EmuReply & res )  [inline], [static], [private]
```

Definition at line 97 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.3 __handleForkFunc()

```
static int32_t TraceThread::__handleForkFunc (
            void * arg )  [inline], [static], [private]
```

Definition at line 103 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.4 __handleInstructionCountFunc()

```
static Sift::Mode TraceThread::__handleInstructionCountFunc (
            void * arg,
            uint32_t icount )  [inline], [static], [private]
```

Definition at line 83 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.5 __handleJoinFunc()

```
static int32_t TraceThread::__handleJoinFunc (
            void * arg,
            int32_t join_thread_id )  [inline], [static], [private]
```

Definition at line 93 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.6 __handleMagicFunc()

```
static uint64_t TraceThread::__handleMagicFunc (
            void * arg,
            uint64_t a,
            uint64_t b,
            uint64_t c )  [inline], [static], [private]
```

Definition at line 95 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.7 __handleNewThreadFunc()

```
static int32_t TraceThread::__handleNewThreadFunc (
            void * arg )  [inline], [static], [private]
```

Definition at line 91 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.8 __handleOutputFunc()

```
static void TraceThread::__handleOutputFunc (
            void * arg,
            uint8_t fd,
            const uint8_t * data,
            uint32_t size )  [inline], [static], [private]
```

Definition at line 87 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.9 __handleRoutineAnnounceFunc()

```
static void TraceThread::__handleRoutineAnnounceFunc (
            void * arg,
            uint64_t eip,
            const char * name,
            const char * imgname,
            uint64_t offset,
            uint32_t line,
            uint32_t column,
            const char * filename )  [inline], [static], [private]
```

Definition at line 101 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.10 __handleRoutineChangeFunc()

```
static void TraceThread::__handleRoutineChangeFunc (
            void * arg,
            Sift::RoutineOpType event,
            uint64_t eip,
            uint64_t esp,
            uint64_t callEip )  [inline], [static], [private]
```

Definition at line 99 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.11 __handleSyscallFunc()

```
static uint64_t TraceThread::__handleSyscallFunc (
            void * arg,
            uint16_t syscall_number,
            const uint8_t * data,
            uint32_t size )  [inline], [static], [private]
```

Definition at line 89 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.12 _va2pa()

```
static  UInt64 TraceThread::_va2pa (
             UInt64 self,
             UInt64 va )  [inline], [static], [private]
```

Definition at line 49 of file trace_thread.h.

Referenced by TraceThread().

6.392.3.13 addDetailedMemoryInfo()

```
void TraceThread::addDetailedMemoryInfo (
             DynamicInstruction * dynins,
            Sift::Instruction & inst,
            const dl::DecodedInst & decoded_inst,
            uint32_t mem_idx,
             Operand::Direction op_type,
            bool is_pretetch,
             PerformanceModel * prfmdl )  [private]
```

Definition at line 699 of file trace_thread.cc.

References DynamicInstruction::addMemory(), HitWhere::PREFETCH_NO_MAPPING, HitWhere::UNKNOWN, va2pa(), and SubsecondTime::Zero().

Referenced by handleInstructionDetailed().

6.392.3.14 decode()

```
Instruction * TraceThread::decode (
            Sift::Instruction & inst )  [private]
```

Definition at line 394 of file trace_thread.cc.

References InstructionDecoder::decode(), m_decoder_cache, Operand::MEMORY, Operand::READ, Instruction::setAddress(), Instruction::setAtomic(), Instruction::setDisassembly(), Instruction::setMicroOps(), Instruction::setSize(), staticDecode(), va2pa(), and Operand::WRITE.

Referenced by handleInstructionDetailed().

6.392.3.15 getCurrentTime()

```
SubsecondTime TraceThread::getCurrentTime ( ) const  [private]
```

Definition at line 388 of file trace_thread.cc.

References Thread::getCore(), PerformanceModel::getElapsedTime(), Core::getPerformanceModel(), LOG_ASSERT_ERROR, and m_thread.

Referenced by handleEmuFunc(), handleForkFunc(), handleNewThreadFunc(), and handleSyscallFunc().

6.392.3.16 getProgressExpect()

```
UInt64 TraceThread::getProgressExpect ( )
```

Definition at line 883 of file trace_thread.cc.

References m_trace.

6.392.3.17 getProgressValue()

```
UInt64 TraceThread::getProgressValue ( )
```

Definition at line 888 of file trace_thread.cc.

References m_trace.

6.392.3.18 getThread()

```
Thread* TraceThread::getThread ( ) const  [inline]
```

Definition at line 146 of file trace_thread.h.

References m_thread.

Referenced by TraceManager::accessMemory(), TraceManager::endApplication(), and TraceManager::signalDone().

6.392.3.19 handleAccessMemory()

```
void TraceThread::handleAccessMemory (
              Core::lock_signal_t lock_signal,
              Core::mem_op_t mem_op_type,
              IntPtr d_addr,
             char * data_buffer,
              UInt32 data_size )
```

Definition at line 893 of file trace_thread.cc.

References Core::LOCK, m_trace, Core::NONE, Core::READ, Core::READ_EX, Core::UNLOCK, and Core::WRITE.

Referenced by TraceManager::accessMemory().

6.392.3.20 handleCacheOnlyFunc()

```
void TraceThread::handleCacheOnlyFunc (
             uint8_t icount,
             Sift::CacheOnlyType type,
             uint64_t eip,
             uint64_t address )  [private]
```

Definition at line 487 of file trace_thread.cc.

References Core::accessBranchPredictor(), Core::accessMemory(), Core::countInstructions(), Thread::getCore(), Core::getPerformanceModel(), PerformanceModel::handleBranchMispredict(), m_thread, Core::MEM_MODELED_COUNT, Core::NONE, Core::READ, Core::readInstructionMemory(), va2pa(), and Core::WRITE.

6.392.3.21 handleEmuFunc()

```
bool TraceThread::handleEmuFunc (
            Sift::EmuType type,
            Sift::EmuRequest & req,
            Sift::EmuReply & res )  [private]
```

Definition at line 288 of file trace_thread.cc.

References SubsecondTime::divideRounded(), cpuid_result_t::eax, cpuid_result_t::ebx, cpuid_result_t::ecx, cpuid_result_t::edx, Core::emulateCpuid(), PerformanceModel::getBranchPredictor(), Thread::getCore(), getCurrentTime(), Core::getId(), PerformanceModel::getInstructionCount(), SubsecondTime::getNS(), BranchPredictor::getNumIncorrectPredictions(), Core::getPerformanceModel(), LOG_ASSERT_ERROR, m_blocked, Thread::m_os_info, m_papi_counters, m_thread, NUM_PAPI_COUNTERS, PAPI_BR_MSP, PAPI_L1_DCM, PAPI_L2_DCM, PAPI_L3_TCM, PAPI_TOT_CYC, PAPI_TOT_INS, Thread::tid, and unblock().

6.392.3.22 handleForkFunc()

```
int32_t TraceThread::handleForkFunc ( )  [private]
```

Definition at line 254 of file trace_thread.cc.

References getCurrentTime(), Thread::getId(), and m_thread.

6.392.3.23 handleInstructionCountFunc()

```
Sift::Mode TraceThread::handleInstructionCountFunc (
            uint32_t icount )  [private]
```

Definition at line 438 of file trace_thread.cc.

References InstMode::CACHE_ONLY, Core::countInstructions(), InstMode::DETAILED, InstMode::FAST_FORWARD, Thread::getCore(), Core::getDvfsDomain(), PerformanceModel::getElapsedTime(), Core::getPerformanceModel(), ComponentPeriod::getPeriod(), InstMode::INVALID, PerformanceModel::iterate(), LOG_ASSERT_ERROR, m_blocked, m_started, m_thread, PerformanceModel::queuePseudoInstruction(), Thread::reschedule(), unblock(), and SyncInstruction::UNSCHEDULED.

6.392.3.24 handleInstructionDetailed()

```
void TraceThread::handleInstructionDetailed (
            Sift::Instruction & inst,
            Sift::Instruction & next_inst,
             PerformanceModel * prfmdl )  [private]
```

Definition at line 648 of file trace_thread.cc.

References DynamicInstruction::addBranch(), addDetailedMemoryInfo(), PerformanceModel::createDynamicInstruction(), decode(), PerformanceModel::iterate(), m_decoder_cache, m_icache, PerformanceModel::queueInstruction(), Operand::READ, va2pa(), and Operand::WRITE.

Referenced by run().

6.392.3.25 handleInstructionWarmup()

```
void TraceThread::handleInstructionWarmup (
             Sift::Instruction & inst,
             Sift::Instruction & next_inst,
              Core * core,
             bool do_icache_warmup,
              UInt64 icache_warmup_addr,
              UInt64 icache_warmup_size )  [private]
```

Definition at line 539 of file trace_thread.cc.

References Core::accessBranchPredictor(), Core::accessMemory(), Core::getPerformanceModel(), Performance↩ Model::handleBranchMispredict(), LOG_ASSERT_ERROR, Core::logMemoryHit(), m_decoder_cache, Core::M↩ EM_MODELED_COUNT, Core::NONE, Core::READ, Core::READ_EX, Core::readInstructionMemory(), static↩ Decode(), va2pa(), and Core::WRITE.

Referenced by run().

6.392.3.26 handleJoinFunc()

```
int32_t TraceThread::handleJoinFunc (
             int32_t thread )  [private]
```

Definition at line 259 of file trace_thread.cc.

References Thread::getId(), and m_thread.

6.392.3.27 handleMagicFunc()

```
uint64_t TraceThread::handleMagicFunc (
             uint64_t a,
             uint64_t b,
             uint64_t c )  [private]
```

Definition at line 265 of file trace_thread.cc.

References Thread::getId(), handleMagicInstruction(), and m_thread.

6.392.3.28 handleNewThreadFunc()

```
int32_t TraceThread::handleNewThreadFunc ( )  [private]
```

Definition at line 249 of file trace_thread.cc.

References getCurrentTime(), Thread::getId(), m_app_id, and m_thread.

6.392.3.29 handleOutputFunc()

```
void TraceThread::handleOutputFunc (
            uint8_t fd,
            const uint8_t * data,
            uint32_t size )  [private]
```

Definition at line 176 of file trace_thread.cc.

References Thread::getId(), m_output_leftover, m_output_leftover_size, and m_thread.

6.392.3.30 handleRoutineAnnounceFunc()

```
void TraceThread::handleRoutineAnnounceFunc (
            uint64_t eip,
            const char * name,
            const char * imgname,
            uint64_t offset,
            uint32_t line,
            uint32_t column,
            const char * filename )  [private]
```

Definition at line 383 of file trace_thread.cc.

6.392.3.31 handleRoutineChangeFunc()

```
void TraceThread::handleRoutineChangeFunc (
            Sift::RoutineOpType event,
            uint64_t eip,
            uint64_t esp,
            uint64_t callEip )  [private]
```

Definition at line 270 of file trace_thread.cc.

References Thread::getRoutineTracer(), LOG_PRINT_ERROR, m_thread, RoutineTracerThread::routineAssert(), RoutineTracerThread::routineEnter(), and RoutineTracerThread::routineExit().

6.392.3.32 handleSyscallFunc()

```
uint64_t TraceThread::handleSyscallFunc (
            uint16_t syscall_number,
            const uint8_t * data,
            uint32_t size )  [private]
```

Definition at line 214 of file trace_thread.cc.

References Thread::getCore(), getCurrentTime(), Thread::getSyscallMdl(), LOG_ASSERT_ERROR, m_blocked, m_thread, SyscallMdl::runEnter(), SyscallMdl::runExit(), and unblock().

6.392.3.33 remapAddress()

```
UInt64 TraceThread::remapAddress (
            UInt64 va_page ) [private]
```

Definition at line 160 of file trace_thread.cc.

References m_address_randomization_table.

Referenced by va2pa().

6.392.3.34 run()

```
void TraceThread::run ( ) [private], [virtual]
```

Implements **Runnable** (p. 1106).

Definition at line 769 of file trace_thread.cc.

References InstMode::CACHE_ONLY, Core::countInstructions(), InstMode::DETAILED, InstMode::FAST_FORWARD, Thread::getCore(), PerformanceModel::getElapsedTime(), Thread::getId(), Core::getPerformanceModel(), handleInstructionDetailed(), handleInstructionWarmup(), itostr(), LOG_PRINT_ERROR, m_bbv_base, m_bbv_count, m_bbv_end, m_bbv_last, m_blocked, m_started, m_stop, m_thread, m_time_start, m_trace, m_trace_has_pa, Thread::reschedule(), unblock(), and SubsecondTime::Zero().

6.392.3.35 spawn()

```
void TraceThread::spawn ( )
```

Definition at line 877 of file trace_thread.cc.

References _Thread::create(), m__thread, and _Thread::run().

Referenced by TraceManager::newThread().

6.392.3.36 staticDecode()

```
const dl::DecodedInst * TraceThread::staticDecode (
            Sift::Instruction & inst ) [private]
```

Definition at line 531 of file trace_thread.cc.

References m_factory.

Referenced by decode(), and handleInstructionWarmup().

6.392.3.37 stop()

```
void TraceThread::stop ( )  [inline]
```

Definition at line 143 of file trace_thread.h.

References m_stop.

6.392.3.38 unblock()

```
void TraceThread::unblock ( )  [private]
```

Definition at line 741 of file trace_thread.cc.

References Thread::getCore(), Thread::getId(), Core::getPerformanceModel(), Thread::getSyscallMdl(), INVALID_THREAD_ID, LOG_ASSERT_ERROR, m_blocked, m_thread, PerformanceModel::queuePseudoInstruction(), Thread::reschedule(), SyscallMdl::runExit(), SyncInstruction::SYSCALL, and SyncInstruction::UNSCHEDULED.

Referenced by handleEmuFunc(), handleInstructionCountFunc(), handleSyscallFunc(), and run().

6.392.3.39 va2pa()

```
UInt64 TraceThread::va2pa (
            UInt64 va,
            bool * noMapping = NULL )  [private]
```

Definition at line 115 of file trace_thread.cc.

References Thread::getAppId(), Thread::getCore(), Core::getId(), m_address_randomization, m_appid_from_coreid, m_thread, m_trace, m_trace_has_pa, pa_core_shift, pa_va_mask, remapAddress(), va_page_mask, and va_page_shift.

Referenced by addDetailedMemoryInfo(), decode(), handleCacheOnlyFunc(), handleInstructionDetailed(), and handleInstructionWarmup().

6.392.4 Member Data Documentation

6.392.4.1 m__thread

```
_Thread* TraceThread::m__thread  [private]
```

Definition at line 53 of file trace_thread.h.

Referenced by spawn(), and ~TraceThread().

6.392.4.2 m_address_randomization

```
bool TraceThread::m_address_randomization  [private]
```

Definition at line 58 of file trace_thread.h.

Referenced by TraceThread(), and va2pa().

6.392.4.3 m_address_randomization_table

```
uint8_t TraceThread::m_address_randomization_table[256]  [private]
```

Definition at line 60 of file trace_thread.h.

Referenced by remapAddress(), and TraceThread().

6.392.4.4 m_app_id

```
app_id_t TraceThread::m_app_id  [private]
```

Definition at line 77 of file trace_thread.h.

Referenced by handleNewThreadFunc().

6.392.4.5 m_appid_from_coreid

```
bool TraceThread::m_appid_from_coreid  [private]
```

Definition at line 59 of file trace_thread.h.

Referenced by va2pa().

6.392.4.6 m_bbv_base

```
UInt64 TraceThread::m_bbv_base  [private]
```

Definition at line 67 of file trace_thread.h.

Referenced by run().

6.392.4.7 m_bbv_count

```
UInt64 TraceThread::m_bbv_count  [private]
```

Definition at line 68 of file trace_thread.h.

Referenced by run().

6.392.4.8 m_bbv_end

```
bool TraceThread::m_bbv_end  [private]
```

Definition at line 70 of file trace_thread.h.

Referenced by run().

6.392.4.9 m_bbv_last

```
UInt64 TraceThread::m_bbv_last  [private]
```

Definition at line 69 of file trace_thread.h.

Referenced by run().

6.392.4.10 m_blocked

```
bool TraceThread::m_blocked  [private]
```

Definition at line 78 of file trace_thread.h.

Referenced by handleEmuFunc(), handleInstructionCountFunc(), handleSyscallFunc(), run(), and unblock().

6.392.4.11 m_cleanup

```
bool TraceThread::m_cleanup  [private]
```

Definition at line 79 of file trace_thread.h.

Referenced by ~TraceThread().

6.392.4.12 m_decoder_cache

```
std::unordered_map< IntPtr, const dl::DecodedInst *> TraceThread::m_decoder_cache  [private]
```

Definition at line 66 of file trace_thread.h.

Referenced by decode(), handleInstructionDetailed(), handleInstructionWarmup(), and ~TraceThread().

6.392.4.13 m_factory

```
dl::DecoderFactory* TraceThread::m_factory  [private]
```

Definition at line 128 of file trace_thread.h.

Referenced by staticDecode().

6.392.4.14 m_icache

```
std::unordered_map< IntPtr,  Instruction *> TraceThread::m_icache  [private]
```

Definition at line 62 of file trace_thread.h.

Referenced by handleInstructionDetailed().

6.392.4.15 m_isa

```
int TraceThread::m_isa = 0  [static], [private]
```

Definition at line 71 of file trace_thread.h.

6.392.4.16 m_lock

```
Lock TraceThread::m_lock  [private]
```

Definition at line 134 of file trace_thread.h.

6.392.4.17 m_output_leftover

```
uint8_t TraceThread::m_output_leftover[160]  [private]
```

Definition at line 73 of file trace_thread.h.

Referenced by handleOutputFunc().

6.392.4.18 m_output_leftover_size

```
uint16_t TraceThread::m_output_leftover_size  [private]
```

Definition at line 74 of file trace_thread.h.

Referenced by handleOutputFunc().

6.392.4.19 m_papi_counters

```
long long* TraceThread::m_papi_counters  [private]
```

Definition at line 132 of file trace_thread.h.

Referenced by handleEmuFunc().

6.392.4.20 m_responsefile

```
String TraceThread::m_responsefile  [private]
```

Definition at line 76 of file trace_thread.h.

Referenced by ~TraceThread().

6.392.4.21 m_started

```
bool TraceThread::m_started  [private]
```

Definition at line 80 of file trace_thread.h.

Referenced by handleInstructionCountFunc(), and run().

6.392.4.22 m_stop

```
bool TraceThread::m_stop  [private]
```

Definition at line 61 of file trace_thread.h.

Referenced by run(), and stop().

6.392.4.23 m_stopped

```
bool TraceThread::m_stopped
```

Definition at line 137 of file trace_thread.h.

Referenced by TraceManager::accessMemory(), and TraceManager::signalDone().

6.392.4.24 m_thread

```
Thread* TraceThread::m_thread  [private]
```

Definition at line 54 of file trace_thread.h.

Referenced by getCurrentTime(), getThread(), handleCacheOnlyFunc(), handleEmuFunc(), handleForkFunc(), handleInstructionCountFunc(), handleJoinFunc(), handleMagicFunc(), handleNewThreadFunc(), handleOutput↩Func(), handleRoutineChangeFunc(), handleSyscallFunc(), run(), unblock(), and va2pa().

6.392.4.25 m_time_start

```
SubsecondTime TraceThread::m_time_start  [private]
```

Definition at line 55 of file trace_thread.h.

Referenced by run().

6.392.4.26 m_trace

```
Sift::Reader TraceThread::m_trace  [private]
```

Definition at line 56 of file trace_thread.h.

Referenced by getProgressExpect(), getProgressValue(), handleAccessMemory(), run(), TraceThread(), and va2pa().

6.392.4.27 m_trace_has_pa

```
bool TraceThread::m_trace_has_pa  [private]
```

Definition at line 57 of file trace_thread.h.

Referenced by run(), and va2pa().

6.392.4.28 m_tracefile

```
String TraceThread::m_tracefile  [private]
```

Definition at line 75 of file trace_thread.h.

Referenced by ~TraceThread().

6.392.4.29 pa_core_shift

```
const  UInt64 TraceThread::pa_core_shift = 48  [static], [private]
```

Definition at line 41 of file trace_thread.h.

Referenced by va2pa().

6.392.4.30 pa_core_size

```
const  UInt64 TraceThread::pa_core_size = 16  [static], [private]
```

Definition at line 42 of file trace_thread.h.

6.392.4.31 pa_va_mask

```
const  UInt64 TraceThread::pa_va_mask = ~((( UInt64(1) <<  pa_core_size) - 1) <<  pa_core_↩
shift)  [static], [private]
```

Definition at line 43 of file trace_thread.h.

Referenced by va2pa().

6.392.4.32 va_page_mask

```
const UInt64 TraceThread::va_page_mask = ( UInt64(1) << va_page_shift) - 1 [static], [private]
```

Definition at line 47 of file trace_thread.h.

Referenced by va2pa().

6.392.4.33 va_page_shift

```
const UInt64 TraceThread::va_page_shift = 12 [static], [private]
```

Definition at line 46 of file trace_thread.h.

Referenced by va2pa().

The documentation for this class was generated from the following files:

- common/trace_frontend/ **trace_thread.h**
- common/trace_frontend/ **trace_thread.cc**

6.393 ParametricDramDirectoryMSI::Transition Class Reference

```
#include <cache_cntlr.h>
```

Public Types

- enum **reason_t** {
  REASON_FIRST = 0, **CORE_RD** = REASON_FIRST, **CORE_WR**, **CORE_RDEX**,
  UPGRADE, **EVICT**, **BACK_INVAL**, **COHERENCY**,
  NUM_REASONS }

6.393.1 Detailed Description

Definition at line 48 of file cache_cntlr.h.

6.393.2 Member Enumeration Documentation

6.393.2.1 reason_t

```
enum ParametricDramDirectoryMSI::Transition::reason_t
```

Enumerator

REASON_FIRST	
CORE_RD	
CORE_WR	
CORE_RDEX	
UPGRADE	
EVICT	
BACK_INVAL	
COHERENCY	
NUM_REASONS	

Definition at line 51 of file cache_cntlr.h.

The documentation for this class was generated from the following file:

- common/core/memory_subsystem/parametric_dram_directory_msi/ **cache_cntlr.h**

6.394 Transport Class Reference

`#include <transport.h>`

Inheritance diagram for Transport:

Classes

- class **Node**

Public Member Functions

- virtual ∼**Transport** ()
- virtual **Node** ∗ **createNode** (**core_id_t** core_id)=0
- virtual void **barrier** ()=0
- virtual **Node** ∗ **getGlobalNode** ()=0

Static Public Member Functions

- static **Transport** ∗ **create** ()
- static **Transport** ∗ **getSingleton** ()

Protected Member Functions

- **Transport** ()

Static Private Attributes

- static **Transport** ∗ **m_singleton**

6.394.1 Detailed Description

Definition at line 8 of file transport.h.

6.394.2 Constructor & Destructor Documentation

6.394.2.1 ∼Transport()

```
virtual Transport::∼Transport ( )  [inline], [virtual]
```

Definition at line 11 of file transport.h.

6.394.2.2 Transport()

```
Transport::Transport ( )  [protected]
```

Definition at line 15 of file transport.cc.

6.394.3 Member Function Documentation

6.394.3.1 barrier()

```
virtual void Transport::barrier ( )  [pure virtual]
```

Implemented in **SmTransport** (p. 1249).

Referenced by SimThreadManager::quitSimThreads(), and Simulator::∼Simulator().

6.394.3.2 create()

```
Transport * Transport::create ( )  [static]
```

Definition at line 19 of file transport.cc.

References m_singleton.

Referenced by Simulator::start().

6.394.3.3 createNode()

```
virtual  Node* Transport::createNode (
             core_id_t core_id )  [pure virtual]
```

Implemented in **SmTransport** (p. 1249).

Referenced by Network::Network().

6.394.3.4 getGlobalNode()

```
virtual  Node* Transport::getGlobalNode ( )  [pure virtual]
```

Implemented in **SmTransport** (p. 1249).

Referenced by SimThreadManager::quitSimThreads().

6.394.3.5 getSingleton()

```
Transport * Transport::getSingleton ( )  [static]
```

Definition at line 28 of file transport.cc.

References m_singleton.

Referenced by Network::Network(), and SimThreadManager::quitSimThreads().

6.394.4 Member Data Documentation

6.394.4.1 m_singleton

```
Transport * Transport::m_singleton  [static], [private]
```

Definition at line 43 of file transport.h.

Referenced by create(), and getSingleton().

The documentation for this class was generated from the following files:

- common/transport/ **transport.h**
- common/transport/ **transport.cc**

6.395 tree_node Struct Reference

```
#include <util.h>
```

Public Attributes

- intptr_t **addr**
- unsigned **inum**
- int **grpno**
- int **prty**
- int **rtwt**
- struct **tree_node** * **lft**
- struct **tree_node** * **rt**

6.395.1 Detailed Description

Definition at line 42 of file util.h.

6.395.2 Member Data Documentation

6.395.2.1 addr

```
intptr_t tree_node::addr
```

Definition at line 43 of file util.h.

6.395.2.2 grpno

```
int tree_node::grpno
```

Definition at line 45 of file util.h.

6.395.2.3 inum

```
unsigned tree_node::inum
```

Definition at line 44 of file util.h.

6.395.2.4 lft

```
struct tree_node* tree_node::lft
```

Definition at line 48 of file util.h.

Referenced by rotate_left(), rotate_right(), and splay().

6.395.2.5 prty

```
int tree_node::prty
```

Definition at line 46 of file util.h.

6.395.2.6 rt

```
struct tree_node * tree_node::rt
```

Definition at line 48 of file util.h.

Referenced by rotate_left(), rotate_right(), and splay().

6.395.2.7 rtwt

```
int tree_node::rtwt
```

Definition at line 47 of file util.h.

Referenced by rotate_left(), and rotate_right().

The documentation for this struct was generated from the following file:

- common/core/memory_subsystem/cheetah/ **util.h**

6.396 TypedAllocator< T, MaxItems > Class Template Reference

```
#include <allocator.h>
```

Inheritance diagram for TypedAllocator< T, MaxItems >:

Public Member Functions

- **TypedAllocator** ()
- virtual ~**TypedAllocator** ()
- virtual void * **alloc** (size_t bytes)
- virtual void **_dealloc** (void *ptr)

Private Attributes

- **UInt64 m_items**
- **FSBAllocator_ElemAllocator**< sizeof(**DataElement**)+sizeof(T), MaxItems, T > **m_alloc**
- **Lock m_lock**

Additional Inherited Members

6.396.1 Detailed Description

template<typename T, unsigned MaxItems = 0>
class TypedAllocator< T, MaxItems >

Definition at line 33 of file allocator.h.

6.396.2 Constructor & Destructor Documentation

6.396.2.1 TypedAllocator()

```
template<typename T , unsigned MaxItems = 0>
TypedAllocator< T, MaxItems >::  TypedAllocator ( )  [inline]
```

Definition at line 43 of file allocator.h.

6.396.2.2 ~TypedAllocator()

```
template<typename T , unsigned MaxItems = 0>
virtual  TypedAllocator< T, MaxItems >::~ TypedAllocator ( )  [inline], [virtual]
```

Definition at line 47 of file allocator.h.

References TypedAllocator< T, MaxItems >::m_items.

6.396.3 Member Function Documentation

6.396.3.1 _dealloc()

```
template<typename T , unsigned MaxItems = 0>
virtual void  TypedAllocator< T, MaxItems >::_dealloc (
            void * ptr )  [inline], [virtual]
```

Implements **Allocator** (p. 79).

Definition at line 68 of file allocator.h.

References TypedAllocator< T, MaxItems >::m_alloc, TypedAllocator< T, MaxItems >::m_items, and TypedAllocator< T, MaxItems >::m_lock.

6.396.3.2 alloc()

```
template<typename T , unsigned MaxItems = 0>
virtual void*  TypedAllocator< T, MaxItems >::alloc (
            size_t bytes )  [inline], [virtual]
```

Implements **Allocator** (p. 79).

Definition at line 58 of file allocator.h.

References Allocator::DataElement::allocator, Allocator::DataElement::data, TypedAllocator< T, MaxItems >::m_alloc, TypedAllocator< T, MaxItems >::m_items, and TypedAllocator< T, MaxItems >::m_lock.

6.396.4 Member Data Documentation

6.396.4.1 m_alloc

```
template<typename T , unsigned MaxItems = 0>
FSBAllocator_ElemAllocator<sizeof( DataElement) + sizeof(T), MaxItems, T>  TypedAllocator< T,
MaxItems >::m_alloc  [private]
```

Definition at line 37 of file allocator.h.

Referenced by TypedAllocator< T, MaxItems >::_dealloc(), and TypedAllocator< T, MaxItems >::alloc().

6.396.4.2 m_items

```
template<typename T , unsigned MaxItems = 0>
UInt64  TypedAllocator< T, MaxItems >::m_items  [private]
```

Definition at line 36 of file allocator.h.

Referenced by TypedAllocator< T, MaxItems >::_dealloc(), TypedAllocator< T, MaxItems >::alloc(), and Typed↩Allocator< T, MaxItems >::∼TypedAllocator().

6.396.4.3 m_lock

```
template<typename T , unsigned MaxItems = 0>
Lock  TypedAllocator< T, MaxItems >::m_lock  [private]
```

Definition at line 40 of file allocator.h.

Referenced by TypedAllocator< T, MaxItems >::_dealloc(), and TypedAllocator< T, MaxItems >::alloc().

The documentation for this class was generated from the following file:

- common/misc/ **allocator.h**

6.397 UnknownInstruction Class Reference

```
#include <instruction.h>
```

Inheritance diagram for UnknownInstruction:

Public Member Functions

- **UnknownInstruction** (**SubsecondTime** cost)

Additional Inherited Members

6.397.1 Detailed Description

Definition at line 236 of file instruction.h.

6.397.2 Constructor & Destructor Documentation

6.397.2.1 UnknownInstruction()

```
UnknownInstruction::UnknownInstruction (
              SubsecondTime cost )  [inline]
```

Definition at line 239 of file instruction.h.

The documentation for this class was generated from the following file:

- common/performance_model/ **instruction.h**

6.398 UnspecifiedType Class Reference

```
#include <FSBAllocator.hh>
```

6.398.1 Detailed Description

Definition at line 36 of file FSBAllocator.hh.

The documentation for this class was generated from the following file:

- common/misc/ **FSBAllocator.hh**

6.399 UnstructuredBuffer Class Reference

```
#include <packetize.h>
```

Public Member Functions

- **UnstructuredBuffer** ()
- const void * **getBuffer** ()
- void **clear** ()
- int **size** ()
- template<class T >
  void **put** (const T &data)
- template<class T >
  bool **get** (T &data)
- template<class T >
  void **put** (const T *data, int num)
- template<class T >
  bool **get** (T *data, int num)
- template<class T >
  UnstructuredBuffer & **operator**<< (const T &data)
- template<class T >
  UnstructuredBuffer & **operator**>> (T &data)
- template<class T , class I >
  UnstructuredBuffer & **operator**<< (std::pair< T *, I > buffer)
- template<class T , class I >
  UnstructuredBuffer & **operator**>> (std::pair< T *, I > buffer)
- **UnstructuredBuffer** & **operator**<< (std::pair< const void *, int > buffer)
- **UnstructuredBuffer** & **operator**>> (std::pair< void *, int > buffer)
- template<> void **put** (const **SubsecondTime** &_data)
- template<> bool **get** (**SubsecondTime** &_data)
- template<> **UnstructuredBuffer** & **operator**<< (const **SubsecondTime** &_data)
- template<> **UnstructuredBuffer** & **operator**>> (**SubsecondTime** &_data)

Private Attributes

- String **m_chars**

6.399.1 Detailed Description

Definition at line 16 of file packetize.h.

6.399.2 Constructor & Destructor Documentation

6.399.2.1 UnstructuredBuffer()

```
UnstructuredBuffer::UnstructuredBuffer ( )
```

Definition at line 3 of file packetize.cc.

6.399.3 Member Function Documentation

6.399.3.1 clear()

```
void UnstructuredBuffer::clear ( )
```

Definition at line 12 of file packetize.cc.

References m_chars.

6.399.3.2 get() [1/3]

```
template<>
bool UnstructuredBuffer::get (
              SubsecondTime & _data )  [inline]
```

Definition at line 88 of file packetize.h.

6.399.3.3 get() [2/3]

```
template<class T >
bool UnstructuredBuffer::get (
             T & data )
```

Definition at line 83 of file packetize.h.

6.399.3.4 get() [3/3]

```
template<class T >
bool UnstructuredBuffer::get (
             T * data,
             int num )
```

Definition at line 59 of file packetize.h.

References m_chars.

6.399.3.5 getBuffer()

```
const void * UnstructuredBuffer::getBuffer ( )
```

Definition at line 7 of file packetize.cc.

References m_chars.

6.399.3.6 operator<<() [1/4]

```
template<>
UnstructuredBuffer& UnstructuredBuffer::operator<< (
            const  SubsecondTime & _data )  [inline]
```

Definition at line 105 of file packetize.h.

6.399.3.7 operator<<() [2/4]

```
template<class T >
UnstructuredBuffer & UnstructuredBuffer::operator<< (
            const T & data )
```

Definition at line 97 of file packetize.h.

6.399.3.8 operator<<() [3/4]

```
 UnstructuredBuffer & UnstructuredBuffer::operator<< (
            std::pair< const void *, int > buffer )
```

Definition at line 24 of file packetize.cc.

References size().

6.399.3.9 operator<<() [4/4]

```
template<class T , class I >
UnstructuredBuffer & UnstructuredBuffer::operator<< (
            std::pair< T *, I > buffer )
```

Definition at line 132 of file packetize.h.

6.399.3.10 operator>>() [1/4]

```
template<class T , class I >
UnstructuredBuffer & UnstructuredBuffer::operator>> (
            std::pair< T *, I > buffer )
```

Definition at line 138 of file packetize.h.

6.399.3.11 operator>>() [2/4]

```
UnstructuredBuffer & UnstructuredBuffer::operator>> (
            std::pair< void *, int > buffer )
```

Definition at line 36 of file packetize.cc.

References __attribute__, and size().

6.399.3.12 operator>>() [3/4]

```
template<>
UnstructuredBuffer& UnstructuredBuffer::operator>> (
             SubsecondTime & _data )  [inline]
```

Definition at line 122 of file packetize.h.

References __attribute__.

6.399.3.13 operator>>() [4/4]

```
template<class T >
UnstructuredBuffer & UnstructuredBuffer::operator>> (
            T & data )
```

Definition at line 113 of file packetize.h.

References __attribute__.

6.399.3.14 put() [1/3]

```
template<>
void UnstructuredBuffer::put (
            const  SubsecondTime & _data )  [inline]
```

Definition at line 76 of file packetize.h.

6.399.3.15 put() [2/3]

```
template<class T >
void UnstructuredBuffer::put (
            const T & data )
```

Definition at line 71 of file packetize.h.

6.399.3.16 put() [3/3]

```
template<class T >
void UnstructuredBuffer::put (
            const T * data,
            int num )
```

Definition at line 53 of file packetize.h.

References m_chars.

6.399.3.17 size()

```
int UnstructuredBuffer::size ( )
```

Definition at line 17 of file packetize.cc.

References m_chars.

Referenced by operator<<(), and operator>>().

6.399.4 Member Data Documentation

6.399.4.1 m_chars

```
String UnstructuredBuffer::m_chars  [private]
```

Definition at line 20 of file packetize.h.

Referenced by clear(), get(), getBuffer(), put(), and size().

The documentation for this class was generated from the following files:

- common/misc/ **packetize.h**
- common/misc/ **packetize.cc**

6.400 InstructionTracer::uop_times_t Struct Reference

```
#include <instruction_tracer.h>
```

Public Attributes

- **SubsecondTime dispatch**
- **SubsecondTime issue**
- **SubsecondTime done**
- **SubsecondTime commit**

6.400.1 Detailed Description

Definition at line 18 of file instruction_tracer.h.

6.400.2 Member Data Documentation

6.400.2.1 commit

SubsecondTime InstructionTracer::uop_times_t::commit

Definition at line 19 of file instruction_tracer.h.

6.400.2.2 dispatch

SubsecondTime InstructionTracer::uop_times_t::dispatch

Definition at line 19 of file instruction_tracer.h.

6.400.2.3 done

SubsecondTime InstructionTracer::uop_times_t::done

Definition at line 19 of file instruction_tracer.h.

6.400.2.4 issue

SubsecondTime InstructionTracer::uop_times_t::issue

Definition at line 19 of file instruction_tracer.h.

Referenced by LoopTracer::traceInstruction().

The documentation for this struct was generated from the following file:

- common/performance_model/instruction_tracers/ **instruction_tracer.h**

6.401 SimFutex::Waiter Struct Reference

#include <syscall_server.h>

Public Member Functions

- **Waiter** (**thread_id_t** _thread_id, int _mask, **SubsecondTime** _timeout)

Public Attributes

- **thread_id_t thread_id**
- int **mask**
- **SubsecondTime timeout**

6.401.1 Detailed Description

Definition at line 22 of file syscall_server.h.

6.401.2 Constructor & Destructor Documentation

6.401.2.1 Waiter()

```
SimFutex::Waiter::Waiter (
              thread_id_t _thread_id,
             int _mask,
              SubsecondTime _timeout )  [inline]
```

Definition at line 24 of file syscall_server.h.

6.401.3 Member Data Documentation

6.401.3.1 mask

```
int SimFutex::Waiter::mask
```

Definition at line 28 of file syscall_server.h.

6.401.3.2 thread_id

```
thread_id_t SimFutex::Waiter::thread_id
```

Definition at line 27 of file syscall_server.h.

Referenced by SimFutex::requeueWaiter().

6.401.3.3 timeout

```
SubsecondTime SimFutex::Waiter::timeout
```

Definition at line 29 of file syscall_server.h.

The documentation for this struct was generated from the following file:

- common/system/ **syscall_server.h**

6.402 PentiumMBranchTargetBuffer::Way Class Reference

Public Member Functions

- **Way** ()

Public Attributes

- std::vector< **UInt32** > **m_tag_offset**
- std::vector< **UInt64** > **m_plru**

6.402.1 Detailed Description

Definition at line 20 of file pentium_m_branch_target_buffer.h.

6.402.2 Constructor & Destructor Documentation

6.402.2.1 Way()

```
PentiumMBranchTargetBuffer::Way::Way ( )  [inline]
```

Definition at line 23 of file pentium_m_branch_target_buffer.h.

6.402.3 Member Data Documentation

6.402.3.1 m_plru

```
std::vector< UInt64> PentiumMBranchTargetBuffer::Way::m_plru
```

Definition at line 29 of file pentium_m_branch_target_buffer.h.

6.402.3.2 m_tag_offset

```
std::vector< UInt32> PentiumMBranchTargetBuffer::Way::m_tag_offset
```

Definition at line 28 of file pentium_m_branch_target_buffer.h.

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **pentium_m_branch_target_buffer.h**

6.403 GlobalPredictor::Way Class Reference

Public Member Functions

- **Way** (**UInt32** entries, **UInt32** tag_bitwidth)

Public Attributes

- std::vector< bool > **m_valid**
- std::vector< uint8_t > **m_tags**
- std::vector< **SaturatingPredictor**< 2 > > **m_predictors**
- std::vector< **UInt64** > **m_lru**
- **UInt32 m_num_entries**
- **UInt32 m_tag_bitwidth**

6.403.1 Detailed Description

Definition at line 122 of file global_predictor.h.

6.403.2 Constructor & Destructor Documentation

6.403.2.1 Way()

```
GlobalPredictor::Way::Way (
            UInt32 entries,
            UInt32 tag_bitwidth )  [inline]
```

Definition at line 126 of file global_predictor.h.

6.403.3 Member Data Documentation

6.403.3.1 m_lru

```
std::vector< UInt64> GlobalPredictor::Way::m_lru
```

Definition at line 140 of file global_predictor.h.

6.403.3.2 m_num_entries

```
UInt32 GlobalPredictor::Way::m_num_entries
```

Definition at line 141 of file global_predictor.h.

6.403.3.3 m_predictors

```
std::vector< SaturatingPredictor<2> > GlobalPredictor::Way::m_predictors
```

Definition at line 139 of file global_predictor.h.

6.403.3.4 m_tag_bitwidth

```
UInt32 GlobalPredictor::Way::m_tag_bitwidth
```

Definition at line 142 of file global_predictor.h.

6.403.3.5 m_tags

```
std::vector<uint8_t> GlobalPredictor::Way::m_tags
```

Definition at line 138 of file global_predictor.h.

6.403.3.6 m_valid

```
std::vector<bool> GlobalPredictor::Way::m_valid
```

Definition at line 137 of file global_predictor.h.

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **global_predictor.h**

6.404 LoopBranchPredictor::Way Class Reference

Public Member Functions

- **Way** (**UInt32** entries, **UInt32** tag_bitwidth)

Public Attributes

- std::vector< uint8_t > **m_tags**
- std::vector< uint8_t > **m_previous_actual**
- std::vector< uint8_t > **m_enabled**
- std::vector< **SaturatingPredictor**< 1 > > **m_predictors**
- std::vector< **UInt64** > **m_lru**
- std::vector< **UInt32** > **m_count**
- std::vector< **UInt32** > **m_limit**
- **UInt32 m_num_entries**
- **UInt32 m_tag_bitwidth**

6.404.1 Detailed Description

Definition at line 264 of file lpb.h.

6.404.2 Constructor & Destructor Documentation

6.404.2.1 Way()

```
LoopBranchPredictor::Way::Way (
              UInt32 entries,
              UInt32 tag_bitwidth )  [inline]
```

Definition at line 268 of file lpb.h.

6.404.3 Member Data Documentation

6.404.3.1 m_count

```
std::vector< UInt32> LoopBranchPredictor::Way::m_count
```

Definition at line 287 of file lpb.h.

6.404.3.2 m_enabled

```
std::vector<uint8_t> LoopBranchPredictor::Way::m_enabled
```

Definition at line 284 of file lpb.h.

6.404.3.3 m_limit

```
std::vector< UInt32> LoopBranchPredictor::Way::m_limit
```

Definition at line 288 of file lpb.h.

6.404.3.4 m_lru

```
std::vector< UInt64> LoopBranchPredictor::Way::m_lru
```

Definition at line 286 of file lpb.h.

6.404.3.5 m_num_entries

```
UInt32 LoopBranchPredictor::Way::m_num_entries
```

Definition at line 289 of file lpb.h.

6.404.3.6 m_predictors

```
std::vector< SaturatingPredictor<1> > LoopBranchPredictor::Way::m_predictors
```

Definition at line 285 of file lpb.h.

6.404.3.7 m_previous_actual

```
std::vector<uint8_t> LoopBranchPredictor::Way::m_previous_actual
```

Definition at line 283 of file lpb.h.

6.404.3.8 m_tag_bitwidth

```
UInt32 LoopBranchPredictor::Way::m_tag_bitwidth
```

Definition at line 290 of file lpb.h.

6.404.3.9 m_tags

```
std::vector<uint8_t> LoopBranchPredictor::Way::m_tags
```

Definition at line 282 of file lpb.h.

The documentation for this class was generated from the following file:

- common/performance_model/branch_predictors/ **lpb.h**

6.405 Windows::WindowEntry Struct Reference

```
#include <windows.h>
```

Public Types

- enum { **ICACHE_OVERLAP** = 1, **BPRED_OVERLAP** = 2, **DCACHE_OVERLAP** = 4 }
- enum { **NO_DEP** = 0, **DATA_DEP** = 1, **INDEP_MISS** = 2 }

Public Member Functions

- void **initialize** (**DynamicMicroOp** *micro_op)
- uint32_t **getWindowIndex** () const
- void **setWindowIndex** (uint32_t index)
- **DynamicMicroOp** * **getDynMicroOp** ()
- const **DynamicMicroOp** * **getDynMicroOp** () const
- const **MicroOp** * **getMicroOp** () const
- uint64_t **getSequenceNumber** () const
- uint64_t **getExecTime** () const
- void **setExecTime** (uint64_t time)
- uint64_t **getDispatchTime** () const
- void **setDispatchTime** (uint64_t time)
- uint64_t **getFetchTime** () const
- void **setFetchTime** (uint64_t time)
- uint64_t **getCpContr** () const
- void **setCpContr** (uint64_t **cpContr**)
- void **addOverlapFlag** (uint32_t flag)
- bool **hasOverlapFlag** (uint32_t flag) const
- void **clearDependent** ()
- bool **isDependent** () const
- bool **isIndependent** () const
- void **setDataDependent** ()
- void **setIndependentMiss** ()

Public Attributes

- uint32_t **windowIndex**
- **DynamicMicroOp** ∗ **uop**
- uint64_t **execTime**
- uint64_t **dispatchTime**
- uint64_t **fetchTime**
- uint64_t **cpContr**
- uint32_t **overlapFlags**
- uint32_t **dependent**
- uint64_t **cphead**
- uint64_t **cptail**
- uint64_t **maxProducer**

6.405.1 Detailed Description

Definition at line 31 of file windows.h.

6.405.2 Member Enumeration Documentation

6.405.2.1 anonymous enum

```
anonymous enum
```

Enumerator

ICACHE_OVERLAP	
BPRED_OVERLAP	
DCACHE_OVERLAP	

Definition at line 51 of file windows.h.

6.405.2.2 anonymous enum

```
anonymous enum
```

Enumerator

NO_DEP	
DATA_DEP	
INDEP_MISS	

Definition at line 55 of file windows.h.

6.405.3 Member Function Documentation

6.405.3.1 addOverlapFlag()

```
void Windows::WindowEntry::addOverlapFlag (
            uint32_t flag )  [inline]
```

Definition at line 84 of file windows.h.

Referenced by IntervalTimer::blockWindow().

6.405.3.2 clearDependent()

```
void Windows::WindowEntry::clearDependent ( )  [inline]
```

Definition at line 87 of file windows.h.

References NO_DEP.

Referenced by IntervalTimer::blockWindow().

6.405.3.3 getCpContr()

```
uint64_t Windows::WindowEntry::getCpContr ( ) const  [inline]
```

Definition at line 81 of file windows.h.

References cpContr.

Referenced by Windows::addFunctionalUnitStats(), and Windows::removeFunctionalUnitStats().

6.405.3.4 getDispatchTime()

```
uint64_t Windows::WindowEntry::getDispatchTime ( ) const  [inline]
```

Definition at line 75 of file windows.h.

References dispatchTime.

6.405.3.5 getDynMicroOp() [1/2]

```
DynamicMicroOp* Windows::WindowEntry::getDynMicroOp ( )  [inline]
```

Definition at line 66 of file windows.h.

References uop.

Referenced by Windows::addFunctionalUnitStats(), IntervalTimer::blockWindow(), Windows::calculateBranch↩ ResolutionLatency(), IntervalTimer::dispatchInstruction(), IntervalTimer::dispatchWindow(), getCpContrType(), IntervalTimer::getMaxProducerExecTime(), getMicroOp(), getSequenceNumber(), IntervalTimer::issueMemOp(), Windows::removeFunctionalUnitStats(), and IntervalTimer::updateCriticalPath().

6.405.3.6 getDynMicroOp() [2/2]

```
const  DynamicMicroOp* Windows::WindowEntry::getDynMicroOp ( ) const  [inline]
```

Definition at line 67 of file windows.h.

References uop.

6.405.3.7 getExecTime()

```
uint64_t Windows::WindowEntry::getExecTime ( ) const  [inline]
```

Definition at line 72 of file windows.h.

References execTime.

Referenced by IntervalTimer::dispatchInstruction(), IntervalTimer::getMaxProducerExecTime(), Windows::long↩ LatencyOperationLatency(), IntervalTimer::updateCriticalPath(), and Windows::updateCriticalPathTail().

6.405.3.8 getFetchTime()

```
uint64_t Windows::WindowEntry::getFetchTime ( ) const  [inline]
```

Definition at line 78 of file windows.h.

References fetchTime.

Referenced by IntervalTimer::getMaxProducerExecTime().

6.405.3.9 getMicroOp()

```
const MicroOp* Windows::WindowEntry::getMicroOp ( ) const [inline]
```

Definition at line 69 of file windows.h.

References getDynMicroOp(), and DynamicMicroOp::getMicroOp().

Referenced by IntervalTimer::blockWindow(), IntervalTimer::dispatchInstruction(), IntervalTimer::dispatchWindow(), getCpContrType(), IntervalTimer::issueMemOp(), and Windows::toString().

6.405.3.10 getSequenceNumber()

```
uint64_t Windows::WindowEntry::getSequenceNumber ( ) const [inline]
```

Definition at line 70 of file windows.h.

References getDynMicroOp(), and DynamicMicroOp::getSequenceNumber().

Referenced by Windows::add(), Windows::getInstruction(), Windows::oldWindowContains(), Windows::toString(), and Windows::windowContains().

6.405.3.11 getWindowIndex()

```
uint32_t Windows::WindowEntry::getWindowIndex ( ) const [inline]
```

Definition at line 63 of file windows.h.

References windowIndex.

Referenced by Windows::calculateBranchResolutionLatency(), Windows::getInstruction(), and Windows::toString().

6.405.3.12 hasOverlapFlag()

```
bool Windows::WindowEntry::hasOverlapFlag (
            uint32_t flag ) const [inline]
```

Definition at line 85 of file windows.h.

Referenced by IntervalTimer::blockWindow(), and IntervalTimer::dispatchInstruction().

6.405.3.13 initialize()

```
void Windows::WindowEntry::initialize (
              DynamicMicroOp * micro_op )
```

Definition at line 18 of file windows.cc.

References cpContr, cphead, cptail, dependent, dispatchTime, execTime, fetchTime, maxProducer, NO_DEP, overlapFlags, and uop.

Referenced by Windows::add().

6.405.3.14 isDependent()

```
bool Windows::WindowEntry::isDependent ( ) const  [inline]
```

Definition at line 88 of file windows.h.

References DATA_DEP.

Referenced by IntervalTimer::blockWindow().

6.405.3.15 isIndependent()

```
bool Windows::WindowEntry::isIndependent ( ) const  [inline]
```

Definition at line 89 of file windows.h.

References INDEP_MISS.

Referenced by IntervalTimer::blockWindow().

6.405.3.16 setCpContr()

```
void Windows::WindowEntry::setCpContr (
             uint64_t cpContr )  [inline]
```

Definition at line 82 of file windows.h.

References cpContr.

Referenced by Windows::updateCriticalPathTail().

6.405.3.17 setDataDependent()

```
void Windows::WindowEntry::setDataDependent ( )  [inline]
```

Definition at line 90 of file windows.h.

References DATA_DEP.

Referenced by IntervalTimer::blockWindow().

6.405.3.18 setDispatchTime()

```
void Windows::WindowEntry::setDispatchTime (
            uint64_t time )  [inline]
```

Definition at line 76 of file windows.h.

6.405.3.19 setExecTime()

```
void Windows::WindowEntry::setExecTime (
            uint64_t time )  [inline]
```

Definition at line 73 of file windows.h.

Referenced by IntervalTimer::blockWindow(), and IntervalTimer::dispatchInstruction().

6.405.3.20 setFetchTime()

```
void Windows::WindowEntry::setFetchTime (
            uint64_t time )  [inline]
```

Definition at line 79 of file windows.h.

6.405.3.21 setIndependentMiss()

```
void Windows::WindowEntry::setIndependentMiss ( )  [inline]
```

Definition at line 91 of file windows.h.

References INDEP_MISS.

Referenced by IntervalTimer::blockWindow().

6.405.3.22 setWindowIndex()

```
void Windows::WindowEntry::setWindowIndex (
            uint32_t index )  [inline]
```

Definition at line 64 of file windows.h.

Referenced by Windows::Windows().

6.405.4 Member Data Documentation

6.405.4.1 cpContr

```
uint64_t Windows::WindowEntry::cpContr
```

The number of cycles this instruction contributes to the critical path.

Definition at line 49 of file windows.h.

Referenced by getCpContr(), initialize(), and setCpContr().

6.405.4.2 cphead

```
uint64_t Windows::WindowEntry::cphead
```

Definition at line 59 of file windows.h.

Referenced by IntervalTimer::dispatchInstruction(), and initialize().

6.405.4.3 cptail

```
uint64_t Windows::WindowEntry::cptail
```

Definition at line 60 of file windows.h.

Referenced by IntervalTimer::dispatchInstruction(), and initialize().

6.405.4.4 dependent

```
uint32_t Windows::WindowEntry::dependent
```

Used during the block window algorithm, shouldn't be here -> has to be int[doubleWindowSize] in **IntervalTimer** (p. 649) or **Windows** (p. 1494).

Definition at line 57 of file windows.h.

Referenced by initialize().

6.405.4.5 dispatchTime

```
uint64_t Windows::WindowEntry::dispatchTime
```

The cycle in which the instruction is dispatched.

Definition at line 45 of file windows.h.

Referenced by getDispatchTime(), and initialize().

6.405.4.6 execTime

```
uint64_t Windows::WindowEntry::execTime
```

The cycle in which the instruction is executed.

Definition at line 43 of file windows.h.

Referenced by getExecTime(), and initialize().

6.405.4.7 fetchTime

```
uint64_t Windows::WindowEntry::fetchTime
```

The cycle in which the instruction was fetched. This clock is not synchronized with the simulator clock.

Definition at line 47 of file windows.h.

Referenced by getFetchTime(), and initialize().

6.405.4.8 maxProducer

```
uint64_t Windows::WindowEntry::maxProducer
```

Definition at line 61 of file windows.h.

Referenced by IntervalTimer::dispatchInstruction(), and initialize().

6.405.4.9 overlapFlags

```
uint32_t Windows::WindowEntry::overlapFlags
```

The latency of the microInstruction can be overlapped by a long latency load. The flag states what is overlapped: a icache miss, a branch mispredict or a dcache miss.

Definition at line 53 of file windows.h.

Referenced by initialize().

6.405.4.10 uop

```
DynamicMicroOp* Windows::WindowEntry::uop
```

We own this, and have to delete it

Definition at line 40 of file windows.h.

Referenced by getDynMicroOp(), initialize(), Windows::Windows(), and Windows::~Windows().

6.405.4.11 windowIndex

```
uint32_t Windows::WindowEntry::windowIndex
```

The index of the microOperation in the window. Constant!

Definition at line 36 of file windows.h.

Referenced by getWindowIndex().

The documentation for this struct was generated from the following files:

- common/performance_model/performance_models/interval_performance_model/ **windows.h**
- common/performance_model/performance_models/interval_performance_model/ **windows.cc**

6.406 Windows Class Reference

`#include <windows.h>`

Classes

- class **Iterator**
- struct **WindowEntry**

Public Member Functions

- int **windowIndex** (int index) const
- int **incrementIndex** (int index) const
- int **decrementIndex** (int index) const
- **Windows** (int windowSize, bool doFunctionalUnitContention, **Core** *core, const **CoreModel** *core_model)
- ~**Windows** ()
- void **clear** ()
- bool **wIsFull** () const
- bool **wIsEmpty** () const
- void **add** (**DynamicMicroOp** *microOp)
- **WindowEntry** & **getInstruction** (uint64_t sequenceNumber) const
- **WindowEntry** & **getLastAdded** () const
- **WindowEntry** & **getInstructionToDispatch** () const
- **WindowEntry** & **getOldestInstruction** () const
- void **dispatchInstruction** ()
- void **clearOldWindow** (uint64_t newCpHead)
- bool **windowContains** (uint64_t sequenceNumber) const
- bool **oldWindowContains** (uint64_t sequenceNumber) const
- int **getOldWindowLength** () const
- uint64_t **getCriticalPathHead** () const
- uint64_t **getCriticalPathTail** () const
- int **getCriticalPathLength** () const
- uint64_t **longLatencyOperationLatency** (**WindowEntry** &uop)
- uint64_t **updateCriticalPathTail** (**WindowEntry** &uop)
- int **getMinimalFlushLatency** (int width) const
- uint64_t **getCpContrFraction** (**CpContrType** type, uint64_t effective_cp_length) const
- **Iterator** **getWindowIterator** () const
- **Iterator** **getOldWindowIterator** () const
- int **calculateBranchResolutionLatency** ()
- uint64_t **getEffectiveCriticalPathLength** (uint64_t critical_path_length, bool update_reason)
- String **toString** ()

Private Member Functions

- **WindowEntry** & **getInstructionByIndex** (int index) const
- void **addFunctionalUnitStats** (const **WindowEntry** &uop)
- void **removeFunctionalUnitStats** (const **WindowEntry** &uop)
- void **clearFunctionalUnitStats** ()

Private Attributes

- const **CoreModel** * **m_core_model**
- **IntervalContention** * **m_interval_contention**
- int **m_window_size**
- int **m_double_window_size**
- **WindowEntry** *const **m_double_window**
- uint32_t *const **m_exec_time_map**
- bool **m_do_functional_unit_contention**
- uint64_t **m_next_sequence_number**
- **RegisterDependencies** *const **m_register_dependencies**
- **MemoryDependencies** *const **m_memory_dependencies**
- int **m_window_head__old_window_tail**
- int **m_window_tail**
- int **m_old_window_head**
- int **m_window_length**
- int **m_old_window_length**
- uint64_t **m_critical_path_head**
- uint64_t **m_critical_path_tail**
- uint64_t **m_cpcontr_bytype** [**CPCONTR_TYPE_SIZE**]
- uint64_t **m_cpcontr_total**

Friends

- class **RegisterDependencies**
- class **MemoryDependencies**

6.406.1 Detailed Description

Definition at line 28 of file windows.h.

6.406.2 Constructor & Destructor Documentation

6.406.2.1 Windows()

```
Windows::Windows (
            int windowSize,
            bool doFunctionalUnitContention,
             Core * core,
            const  CoreModel * core_model )
```

Definition at line 38 of file windows.cc.

References clear(), clearFunctionalUnitStats(), m_double_window, m_double_window_size, m_window_size, Windows::WindowEntry::setWindowIndex(), and Windows::WindowEntry::uop.

6.406.2.2 ~Windows()

```
Windows::~Windows ( )
```

Definition at line 60 of file windows.cc.

References m_double_window, m_double_window_size, m_exec_time_map, m_memory_dependencies, m_↩ register_dependencies, and Windows::WindowEntry::uop.

6.406.3 Member Function Documentation

6.406.3.1 add()

```
void Windows::add (
            DynamicMicroOp * micro_op )
```

Add the microOperation to the window and calculate its dependencies.

Definition at line 141 of file windows.cc.

References getInstructionByIndex(), Windows::WindowEntry::getSequenceNumber(), incrementIndex(), Windows↩ ::WindowEntry::initialize(), LOG_ASSERT_ERROR, m_memory_dependencies, m_next_sequence_number, m↩ _old_window_head, m_register_dependencies, m_window_length, m_window_tail, RegisterDependencies::set↩ Dependencies(), MemoryDependencies::setDependencies(), DynamicMicroOp::setSequenceNumber(), and wIs↩ Full().

Referenced by IntervalTimer::simulate().

6.406.3.2 addFunctionalUnitStats()

```
void Windows::addFunctionalUnitStats (
            const  WindowEntry & uop )  [private]
```

Definition at line 99 of file windows.cc.

References IntervalContention::addFunctionalUnitStats(), Windows::WindowEntry::getCpContr(), getCpContr↩ Type(), Windows::WindowEntry::getDynMicroOp(), m_cpcontr_bytype, m_cpcontr_total, and m_interval_↩ contention.

Referenced by dispatchInstruction().

6.406.3.3 calculateBranchResolutionLatency()

```
int Windows::calculateBranchResolutionLatency ( )
```

Definition at line 351 of file windows.cc.

References DynamicMicroOp::getDependency(), Windows::WindowEntry::getDynMicroOp(), DynamicMicroOp↩ ::getExecLatency(), getInstruction(), getInstructionByIndex(), getInstructionToDispatch(), Windows::WindowEntry↩ ::getWindowIndex(), m_exec_time_map, m_old_window_length, m_window_head__old_window_tail, oldWindow↩ Contains(), and windowIndex().

Referenced by IntervalTimer::dispatchInstruction().

6.406.3.4 clear()

```
void Windows::clear ( )
```

Definition at line 71 of file windows.cc.

References RegisterDependencies::clear(), MemoryDependencies::clear(), clearFunctionalUnitStats(), m_critical↩ _path_head, m_critical_path_tail, m_double_window_size, m_exec_time_map, m_memory_dependencies, m_↩ next_sequence_number, m_old_window_head, m_old_window_length, m_register_dependencies, m_window_↩ head__old_window_tail, m_window_length, and m_window_tail.

Referenced by Windows().

6.406.3.5 clearFunctionalUnitStats()

```
void Windows::clearFunctionalUnitStats ( )  [private]
```

Definition at line 89 of file windows.cc.

References IntervalContention::clearFunctionalUnitStats(), CPCONTR_TYPE_SIZE, m_cpcontr_bytype, m_↩ cpcontr_total, and m_interval_contention.

Referenced by clear(), clearOldWindow(), and Windows().

6.406.3.6 clearOldWindow()

```
void Windows::clearOldWindow (
            uint64_t newCpHead )
```

Definition at line 231 of file windows.cc.

References clearFunctionalUnitStats(), m_critical_path_head, m_critical_path_tail, m_old_window_head, m_old↩ _window_length, and m_window_head__old_window_tail.

Referenced by IntervalTimer::dispatchInstruction(), and IntervalTimer::updateCriticalPath().

6.406.3.7 decrementIndex()

```
int Windows::decrementIndex (
            int index ) const
```

Definition at line 128 of file windows.cc.

References m_double_window_size.

Referenced by getLastAdded(), oldWindowContains(), and windowContains().

6.406.3.8 dispatchInstruction()

```
void Windows::dispatchInstruction ( )
```

Definition at line 209 of file windows.cc.

References addFunctionalUnitStats(), getInstructionByIndex(), incrementIndex(), m_critical_path_head, m_old↩_window_head, m_old_window_length, m_window_head__old_window_tail, m_window_length, m_window_size, and removeFunctionalUnitStats().

Referenced by IntervalTimer::dispatchWindow().

6.406.3.9 getCpContrFraction()

```
uint64_t Windows::getCpContrFraction (
             CpContrType type,
            uint64_t effective_cp_length ) const
```

Definition at line 318 of file windows.cc.

References m_cpcontr_bytype, and m_cpcontr_total.

Referenced by IntervalTimer::dispatchWindow().

6.406.3.10 getCriticalPathHead()

```
uint64_t Windows::getCriticalPathHead ( ) const
```

Definition at line 259 of file windows.cc.

References m_critical_path_head.

Referenced by IntervalTimer::blockWindow(), and IntervalTimer::dispatchInstruction().

6.406.3.11 getCriticalPathLength()

```
int Windows::getCriticalPathLength ( ) const
```

Definition at line 269 of file windows.cc.

References LOG_PRINT_WARNING_ONCE, m_critical_path_head, and m_critical_path_tail.

Referenced by IntervalTimer::calculateCurrentDispatchRate(), IntervalTimer::dispatchInstruction(), and IntervalTimer::dispatchWindow().

6.406.3.12 getCriticalPathTail()

```
uint64_t Windows::getCriticalPathTail ( ) const
```

Definition at line 264 of file windows.cc.

References m_critical_path_tail.

Referenced by IntervalTimer::dispatchInstruction().

6.406.3.13 getEffectiveCriticalPathLength()

```
uint64_t Windows::getEffectiveCriticalPathLength (
            uint64_t critical_path_length,
            bool update_reason )
```

Definition at line 197 of file windows.cc.

References IntervalContention::getEffectiveCriticalPathLength(), m_do_functional_unit_contention, and m_interval_contention.

Referenced by IntervalTimer::calculateCurrentDispatchRate(), and IntervalTimer::dispatchWindow().

6.406.3.14 getInstruction()

```
Windows::WindowEntry & Windows::getInstruction (
            uint64_t sequenceNumber ) const
```

Definition at line 164 of file windows.cc.

References getInstructionByIndex(), Windows::WindowEntry::getSequenceNumber(), Windows::WindowEntry::getWindowIndex(), m_window_head__old_window_tail, and windowIndex().

Referenced by IntervalTimer::blockWindow(), calculateBranchResolutionLatency(), and IntervalTimer::getMaxProducerExecTime().

6.406.3.15 getInstructionByIndex()

```
Windows::WindowEntry & Windows::getInstructionByIndex (
            int index ) const  [private]
```

Definition at line 158 of file windows.cc.

References LOG_ASSERT_ERROR, m_double_window, and m_double_window_size.

Referenced by add(), calculateBranchResolutionLatency(), dispatchInstruction(), getInstruction(), getInstruction↩ ToDispatch(), getLastAdded(), getOldestInstruction(), oldWindowContains(), toString(), and windowContains().

6.406.3.16 getInstructionToDispatch()

```
Windows::WindowEntry & Windows::getInstructionToDispatch ( ) const
```

Definition at line 187 of file windows.cc.

References getInstructionByIndex(), and m_window_head__old_window_tail.

Referenced by calculateBranchResolutionLatency(), and IntervalTimer::dispatchWindow().

6.406.3.17 getLastAdded()

```
Windows::WindowEntry & Windows::getLastAdded ( ) const
```

Definition at line 182 of file windows.cc.

References decrementIndex(), getInstructionByIndex(), and m_window_tail.

6.406.3.18 getMinimalFlushLatency()

```
int Windows::getMinimalFlushLatency (
            int width ) const
```

Definition at line 313 of file windows.cc.

References m_old_window_length.

Referenced by IntervalTimer::dispatchInstruction().

6.406.3.19 getOldestInstruction()

```
Windows::WindowEntry & Windows::getOldestInstruction ( ) const
```

Definition at line 192 of file windows.cc.

References getInstructionByIndex(), and m_old_window_head.

Referenced by IntervalTimer::getMaxProducerExecTime().

6.406.3.20 getOldWindowIterator()

```
Windows::Iterator Windows::getOldWindowIterator ( ) const
```

Definition at line 346 of file windows.cc.

References m_old_window_head, and m_window_head__old_window_tail.

6.406.3.21 getOldWindowLength()

```
int Windows::getOldWindowLength ( ) const
```

Definition at line 254 of file windows.cc.

References m_old_window_length.

Referenced by IntervalTimer::calculateCurrentDispatchRate().

6.406.3.22 getWindowIterator()

```
Windows::Iterator Windows::getWindowIterator ( ) const
```

Definition at line 341 of file windows.cc.

References m_window_head__old_window_tail, and m_window_tail.

Referenced by IntervalTimer::blockWindow().

6.406.3.23 incrementIndex()

```
int Windows::incrementIndex (
            int index ) const
```

Definition at line 123 of file windows.cc.

References m_double_window_size.

Referenced by add(), and dispatchInstruction().

6.406.3.24 longLatencyOperationLatency()

```
uint64_t Windows::longLatencyOperationLatency (
             WindowEntry & uop )
```

Definition at line 279 of file windows.cc.

References Windows::WindowEntry::getExecTime(), and m_critical_path_tail.

Referenced by IntervalTimer::updateCriticalPath().

6.406.3.25 oldWindowContains()

```
bool Windows::oldWindowContains (
            uint64_t sequenceNumber ) const
```

Definition at line 247 of file windows.cc.

References decrementIndex(), getInstructionByIndex(), Windows::WindowEntry::getSequenceNumber(), m_old_↩ window_head, and m_window_head__old_window_tail.

Referenced by calculateBranchResolutionLatency(), and IntervalTimer::getMaxProducerExecTime().

6.406.3.26 removeFunctionalUnitStats()

```
void Windows::removeFunctionalUnitStats (
            const  WindowEntry & uop )  [private]
```

Definition at line 106 of file windows.cc.

References Windows::WindowEntry::getCpContr(), getCpContrType(), Windows::WindowEntry::getDynMicroOp(), m_cpcontr_bytype, m_cpcontr_total, m_interval_contention, and IntervalContention::removeFunctionalUnitStats().

Referenced by dispatchInstruction().

6.406.3.27 toString()

```
String Windows::toString ( )
```

Definition at line 393 of file windows.cc.

References getInstructionByIndex(), Windows::WindowEntry::getMicroOp(), Windows::WindowEntry::getSequenceNumber(), Windows::WindowEntry::getWindowIndex(), m_double_window_size, m_old_window_head, m_window_head__old_window_tail, m_window_tail, and MicroOp::toString().

6.406.3.28 updateCriticalPathTail()

```
uint64_t Windows::updateCriticalPathTail (
              WindowEntry & uop )
```

Definition at line 298 of file windows.cc.

References Windows::WindowEntry::getExecTime(), CoreModel::getLongLatencyCutoff(), LOG_PRINT_WARNING_ONCE, m_core_model, m_critical_path_tail, and Windows::WindowEntry::setCpContr().

Referenced by IntervalTimer::updateCriticalPath().

6.406.3.29 windowContains()

```
bool Windows::windowContains (
             uint64_t sequenceNumber ) const
```

Definition at line 240 of file windows.cc.

References decrementIndex(), getInstructionByIndex(), Windows::WindowEntry::getSequenceNumber(), m_window_head__old_window_tail, and m_window_tail.

Referenced by IntervalTimer::blockWindow().

6.406.3.30 windowIndex()

```
int Windows::windowIndex (
             int index ) const
```

Definition at line 133 of file windows.cc.

References m_double_window_size.

Referenced by calculateBranchResolutionLatency(), and getInstruction().

6.406.3.31 wIsEmpty()

```
bool Windows::wIsEmpty ( ) const
```

Definition at line 118 of file windows.cc.

References m_window_length.

Referenced by IntervalTimer::dispatchWindow().

6.406.3.32 wIsFull()

```
bool Windows::wIsFull ( ) const
```

Definition at line 113 of file windows.cc.

References m_window_length, and m_window_size.

Referenced by add(), and IntervalTimer::simulate().

6.406.4 Friends And Related Function Documentation

6.406.4.1 MemoryDependencies

```
friend class MemoryDependencies [friend]
```

Definition at line 201 of file windows.h.

6.406.4.2 RegisterDependencies

```
friend class RegisterDependencies [friend]
```

Definition at line 200 of file windows.h.

6.406.5 Member Data Documentation

6.406.5.1 m_core_model

```
const CoreModel* Windows::m_core_model [private]
```

Definition at line 165 of file windows.h.

Referenced by updateCriticalPathTail().

6.406.5.2 m_cpcontr_bytype

```
uint64_t Windows::m_cpcontr_bytype[ CPCONTR_TYPE_SIZE] [private]
```

Definition at line 191 of file windows.h.

Referenced by addFunctionalUnitStats(), clearFunctionalUnitStats(), IntervalTimer::dispatchWindow(), getCp↩ ContrFraction(), and removeFunctionalUnitStats().

6.406.5.3 m_cpcontr_total

```
uint64_t Windows::m_cpcontr_total [private]
```

Definition at line 192 of file windows.h.

Referenced by addFunctionalUnitStats(), clearFunctionalUnitStats(), IntervalTimer::dispatchWindow(), getCp↩ ContrFraction(), and removeFunctionalUnitStats().

6.406.5.4 m_critical_path_head

```
uint64_t Windows::m_critical_path_head [private]
```

Definition at line 188 of file windows.h.

Referenced by clear(), clearOldWindow(), dispatchInstruction(), getCriticalPathHead(), and getCriticalPathLength().

6.406.5.5 m_critical_path_tail

```
uint64_t Windows::m_critical_path_tail [private]
```

Definition at line 189 of file windows.h.

Referenced by clear(), clearOldWindow(), getCriticalPathLength(), getCriticalPathTail(), longLatencyOperation↩ Latency(), and updateCriticalPathTail().

6.406.5.6 m_do_functional_unit_contention

```
bool Windows::m_do_functional_unit_contention  [private]
```

Definition at line 174 of file windows.h.

Referenced by getEffectiveCriticalPathLength().

6.406.5.7 m_double_window

```
WindowEntry* const Windows::m_double_window  [private]
```

Definition at line 171 of file windows.h.

Referenced by getInstructionByIndex(), Windows(), and ∼Windows().

6.406.5.8 m_double_window_size

```
int Windows::m_double_window_size  [private]
```

Definition at line 169 of file windows.h.

Referenced by clear(), decrementIndex(), getInstructionByIndex(), incrementIndex(), toString(), windowIndex(), Windows(), and ∼Windows().

6.406.5.9 m_exec_time_map

```
uint32_t* const Windows::m_exec_time_map  [private]
```

Definition at line 172 of file windows.h.

Referenced by calculateBranchResolutionLatency(), clear(), and ∼Windows().

6.406.5.10 m_interval_contention

```
IntervalContention* Windows::m_interval_contention  [private]
```

Definition at line 166 of file windows.h.

Referenced by addFunctionalUnitStats(), clearFunctionalUnitStats(), getEffectiveCriticalPathLength(), and removeFunctionalUnitStats().

6.406.5.11 m_memory_dependencies

```
MemoryDependencies* const Windows::m_memory_dependencies  [private]
```

Definition at line 179 of file windows.h.

Referenced by add(), clear(), and ~Windows().

6.406.5.12 m_next_sequence_number

```
uint64_t Windows::m_next_sequence_number  [private]
```

Definition at line 176 of file windows.h.

Referenced by add(), and clear().

6.406.5.13 m_old_window_head

```
int Windows::m_old_window_head  [private]
```

Definition at line 183 of file windows.h.

Referenced by add(), clear(), clearOldWindow(), dispatchInstruction(), getOldestInstruction(), getOldWindow↩Iterator(), oldWindowContains(), and toString().

6.406.5.14 m_old_window_length

```
int Windows::m_old_window_length  [private]
```

Definition at line 186 of file windows.h.

Referenced by calculateBranchResolutionLatency(), clear(), clearOldWindow(), dispatchInstruction(), getMinimal↩FlushLatency(), and getOldWindowLength().

6.406.5.15 m_register_dependencies

```
RegisterDependencies* const Windows::m_register_dependencies  [private]
```

Definition at line 178 of file windows.h.

Referenced by add(), clear(), and ~Windows().

6.406.5.16 m_window_head__old_window_tail

```
int Windows::m_window_head__old_window_tail  [private]
```

Definition at line 181 of file windows.h.

Referenced by calculateBranchResolutionLatency(), clear(), clearOldWindow(), dispatchInstruction(), getInstruction(), getInstructionToDispatch(), getOldWindowIterator(), getWindowIterator(), oldWindowContains(), toString(), and windowContains().

6.406.5.17 m_window_length

```
int Windows::m_window_length  [private]
```

Definition at line 185 of file windows.h.

Referenced by add(), clear(), dispatchInstruction(), wIsEmpty(), and wIsFull().

6.406.5.18 m_window_size

```
int Windows::m_window_size  [private]
```

Definition at line 168 of file windows.h.

Referenced by dispatchInstruction(), Windows(), and wIsFull().

6.406.5.19 m_window_tail

```
int Windows::m_window_tail  [private]
```

Definition at line 182 of file windows.h.

Referenced by add(), clear(), getLastAdded(), getWindowIterator(), toString(), and windowContains().

The documentation for this class was generated from the following files:

- common/performance_model/performance_models/interval_performance_model/ **windows.h**
- common/performance_model/performance_models/interval_performance_model/ **windows.cc**

Chapter 7

File Documentation

7.1 common/config/config.cpp File Reference

```
#include "config.hpp"
#include <boost/algorithm/string.hpp>
#include <cstdarg>
#include <cstdio>
```

Namespaces

- **config**

Functions

- void **config::Error** (const char *format,...)

7.2 common/config/config.d File Reference

7.3 common/misc/config.d File Reference

7.4 common/config/config.hpp File Reference

```
#include "fixed_types.h"
#include "key.hpp"
#include "section.hpp"
#include "config_exceptions.hpp"
#include <vector>
#include <map>
#include <iostream>
```

Classes

- class **config::Config**

  Config *(*p. *328): A class for managing the interface to persistent configuration entries defined at runtime. This class is used to manage a configuration interface. It is the base class for which different back ends will derive from.*

Namespaces

- **config**

Typedefs

- typedef std::vector< String > **config::PathElementList**
- typedef std::pair< String, String > **config::PathPair**

Functions

- **config::__attribute__** ((__noreturn__)) void Error(const char *msg

7.4.1 Detailed Description

This file contains most of the interface for the external->in memory configuration management interface. There is a **Config** (p. 341) class which is the main interface to the outside world for loading files, getting and setting values as well as saving the given configuration. This base class is then derived from for the different given backends. This **Config** (p. 341) class has a root 'Section'. Each section has a list of subsections and a list of keys. Each key actually contains the value.

7.5 common/config/config_exceptions.hpp File Reference

```
#include <iostream>
#include <exception>
```

Classes

- class **config::parserError**
- class **config::KeyNotFound**
- class **config::FileNotFound**
- class **config::SaveError**

Namespaces

- **config**

7.6 common/config/config_file.cpp File Reference

```
#include "config_file.hpp"
#include "config_exceptions.hpp"
#include "itostr.h"
#include <iostream>
#include <fstream>
#include <sstream>
#include <errno.h>
#include <boost/algorithm/string.hpp>
#include <boost/version.hpp>
#include <boost/lexical_cast.hpp>
```

Namespaces

- **config**

7.6.1 Detailed Description

The file interface to the config class.

7.7 common/config/config_file.d File Reference

7.8 common/config/config_file.hpp File Reference

```
#include "config.hpp"
#include "config_file_grammar.hpp"
#include <fstream>
```

Classes

- class **config::ConfigFile**

  ConfigFile *(*p. *362): A flat-file interface for the* ***Config*** *(*p. *328) Class. This file contains the class that is used to interface a flat file for config input / output. It uses boost::spirit for the config grammar.*

Namespaces

- **config**

7.8.1 Detailed Description

This file contains the class responsible for 'flat file' configurations. It accepts files very similar to the windows .reg file format, as in sub-sections are collapsed like the following:

```
---------------------
[main]
value_in_main = 1

[main/subsection]
value_in_sub = "asdf"
"value with spaces" = 4
---------------------
```

It uses boost::regex to perform the parsing. Note that comments are not preserved across saves.

7.9 common/config/config_file_grammar.hpp File Reference

```
#include "config_file.hpp"
#include <boost/version.hpp>
#include <boost/spirit/include/classic_core.hpp>
#include <boost/spirit/include/classic_parse_tree.hpp>
#include <boost/spirit/include/classic_ast.hpp>
#include <boost/spirit/include/classic_confix.hpp>
#include <boost/spirit/include/classic_escape_char.hpp>
#include <boost/spirit/include/classic_chset.hpp>
#include <iostream>
```

Classes

- struct **config::config_parser**
- struct **config::config_parser::Name**
- struct **config::config_parser::NodeValue**
- struct **config::config_parser::definition**< **ScannerT** >

Namespaces

- **config**

Typedefs

- typedef int_parser< **SInt64**, 10, 1, -1 > **config::long_parser_t**
- typedef tree_parse_info< config_parser::iterator_t, config_parser::factory_t > **config::parse_info_t**
- typedef tree_match< config_parser::iterator_t, config_parser::factory_t > **config::tree_match_t**
- typedef tree_match_t::node_t **config::node_t**
- typedef tree_match_t::const_tree_iterator **config::tree_iter_t**

Enumerations

- enum **config::RuleID** {
  config::defaultID, **config::sectionID**, **config::keyNameID**, **config::keyValueID**,
  config::keySeparatorID, **config::keyValueArrayID**, **config::keyValueSpanID**, **config::keyID**,
  config::sectionNameID, **config::stringID**, **config::configID** }

Variables

- static long_parser_t **config::long_p**

7.9.1 Detailed Description

This file represents the parser element of the bl::config::ConfigFile interface. It uses boost:spirit to create an AST of the loaded config text, then simply walks that tree adding the appropriate sections and keys to the ConfigFile class.

7.10 common/config/key.cpp File Reference

```
#include "key.hpp"
#include <iostream>
#include <boost/version.hpp>
#include <boost/lexical_cast.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/spirit/include/classic_core.hpp>
```

Namespaces

- **config**

Variables

- static long_parser_t **config::long_p**

7.11 common/config/key.d File Reference

7.12 common/config/key.hpp File Reference

```
#include "fixed_types.h"
```

Classes

- class **config::Key**

  Key *(p. 669): A configuration setting entry This class is used to hold a given setting from a configuration. It contains the actual data, as well as functions to get the type.*

Namespaces

- **config**

Enumerations

- enum **config::KeyType** { **config::TYPE_INT_VALID** = 0x01, **config::TYPE_FLOAT_VALID** = 0x02, **config::TYPE_STRING_VALID** = 0x04, **config::TYPE_BOOL_VALID** = 0x08 }

  Enumeration for the types a given key may be.

7.13 common/config/section.cpp File Reference

```
#include "section.hpp"
#include <boost/algorithm/string.hpp>
#include <iostream>
```

Namespaces

- **config**

7.14 common/config/section.d File Reference

7.15 common/config/section.hpp File Reference

```
#include "fixed_types.h"
#include "key.hpp"
#include <vector>
#include <map>
```

Classes

- class **config::Section**

  Section (p. 1163)*: A configuration section entry This class is used to hold a given section from a configuration. It contains both a list of subsections as well as a list of keys. The root of a configuration tree is of this class type.*

Namespaces

- **config**

Typedefs

- typedef std::map< String, Section * > **config::SectionList**
- typedef std::map< String, Key * > **config::KeyList**
- typedef std::map< String, std::vector< Key * > > **config::KeyArrayList**

7.16 common/core/bbv_count.cc File Reference

```
#include "bbv_count.h"
#include "simulator.h"
#include "config.hpp"
#include "stats.h"
#include "rng.h"
```

7.17 common/core/bbv_count.d File Reference

7.18 common/core/bbv_count.h File Reference

```
#include "fixed_types.h"
#include <vector>
```

Classes

- class **BbvCount**

7.19 common/core/core.cc File Reference

```
#include "core.h"
#include "network.h"
#include "syscall_model.h"
#include "branch_predictor.h"
#include "memory_manager_base.h"
#include "performance_model.h"
#include "instruction.h"
#include "clock_skew_minimization_object.h"
#include "core_manager.h"
#include "dvfs_manager.h"
#include "hooks_manager.h"
#include "trace_manager.h"
#include "simulator.h"
#include "log.h"
#include "config.hpp"
#include "stats.h"
#include "topology_info.h"
#include "cheetah_manager.h"
#include <cstring>
```

Macros

- #define **MYLOG**(...) {}
- #define **VERBOSE** 0

Functions

- const char * **ModeledString** (**Core::MemModeled** modeled)
- **MemoryResult** **makeMemoryResult** (**HitWhere::where_t** _hit_where, **SubsecondTime** _latency)
- **__attribute__** ((weak)) void **applicationMemCopy**(void *dest

Variables

- const char * **core_state_names** [ ]
- const void * **src**
- const void size_t **n**

7.19.1 Macro Definition Documentation

7.19.1.1 MYLOG

```
#define MYLOG(
              ...  ) {}
```

Definition at line 26 of file core.cc.

7.19.1.2 VERBOSE

```
#define VERBOSE 0
```

Definition at line 29 of file core.cc.

7.19.2 Function Documentation

7.19.2.1 __attribute__()

```
__attribute__ (
             (weak)  )
```

7.19.2.2 makeMemoryResult()

```
MemoryResult makeMemoryResult (
             HitWhere::where_t _hit_where,
             SubsecondTime _latency )
```

Definition at line 212 of file core.cc.

References MemoryResult::hit_where, MemoryResult::latency, LOG_ASSERT_ERROR, and HitWhere::NUM_H↩ ITWHERES.

Referenced by Core::accessMemory(), lite::handleMemoryReadDetailedIssue(), lite::handleMemoryWrite↩ DetailedIssue(), Core::initiateMemoryAccess(), PthreadEmu::MutexLock(), PthreadEmu::MutexUnlock(), Core↩ ::nativeMemOp(), and Core::readInstructionMemory().

7.19.2.3 ModeledString()

```
const char* ModeledString (
             Core::MemModeled modeled )
```

Definition at line 31 of file core.cc.

References Core::MEM_MODELED_COUNT, Core::MEM_MODELED_COUNT_TLBTIME, Core::MEM_MODEL↩ ED_FENCED, Core::MEM_MODELED_NONE, Core::MEM_MODELED_RETURN, and Core::MEM_MODELED↩ _TIME.

Referenced by Core::initiateMemoryAccess().

7.19.3 Variable Documentation

7.19.3.1 core_state_names

```
const char* core_state_names[]
```

Initial value:

```
= {
   "running",
   "initializing",
   "stalled",
   "sleeping",
   "waking_up",
   "idle",
   "broken",
}
```

Definition at line 45 of file core.cc.

Referenced by Core::CoreStateString().

7.19.3.2 n

```
const void size_t n
```

Initial value:

```
{
   memcpy(dest, src, n)
```

Definition at line 520 of file core.cc.

Referenced by applicationMemCopy(), ceilLog2(), countBits(), BarrierSyncServer::doRelease(), floorLog2(), getHashByStacktrace(), SimpleBimodalTable::ilog2(), isPower2(), config::config_parser::Name::operator()(), SaturatingPredictor< n >::reset(), and TraceManager::Monitor::run().

7.19.3.3 src

```
const void* src
```

Definition at line 519 of file core.cc.

Referenced by applicationMemCopy(), atomic_add_subsecondtime(), Network::netRecv(), Network::netRecv↩From(), and StableIterator< T >::operator=().

7.20 common/core/core.d File Reference

7.21 common/core/core.h File Reference

```
#include "mem_component.h"
#include "fixed_types.h"
#include "lock.h"
#include "packet_type.h"
#include "subsecond_time.h"
#include "bbv_count.h"
#include "cpuid.h"
#include "hit_where.h"
```

Classes

- struct **MemoryResult**
- class **Core**

Functions

- **MemoryResult** **makeMemoryResult** (**HitWhere::where_t** _hit_where, **SubsecondTime** _latency)
- void **applicationMemCopy** (void *dest, const void * **src**, size_t **n**)

7.21.1 Function Documentation

7.21.1.1 applicationMemCopy()

```
void applicationMemCopy (
            void * dest,
            const void * src,
            size_t n )
```

Definition at line 96 of file pin_sim.cc.

References n, and src.

Referenced by Core::nativeMemOp().

7.21.1.2 makeMemoryResult()

```
MemoryResult makeMemoryResult (
             HitWhere::where_t _hit_where,
             SubsecondTime _latency )
```

Definition at line 212 of file core.cc.

References MemoryResult::hit_where, MemoryResult::latency, LOG_ASSERT_ERROR, and HitWhere::NUM_H↩ ITWHERES.

Referenced by Core::accessMemory(), lite::handleMemoryReadDetailedIssue(), lite::handleMemoryWrite↩ DetailedIssue(), Core::initiateMemoryAccess(), PthreadEmu::MutexLock(), PthreadEmu::MutexUnlock(), Core↩ ::nativeMemOp(), and Core::readInstructionMemory().

7.22 common/core/memory_subsystem/address_home_lookup.cc File Reference

```
#include "address_home_lookup.h"
#include "log.h"
```

7.23 common/core/memory_subsystem/address_home_lookup.d File Reference

7.24 common/core/memory_subsystem/address_home_lookup.h File Reference

```
#include <vector>
#include "fixed_types.h"
```

Classes

- class **AddressHomeLookup**

7.25 common/core/memory_subsystem/cache/cache.cc File Reference

```
#include "simulator.h"
#include "cache.h"
#include "log.h"
```

7.26 common/core/memory_subsystem/cache/cache.d File Reference

7.27 common/core/memory_subsystem/cache/cache.h File Reference

```
#include "cache_base.h"
#include "cache_set.h"
#include "cache_block_info.h"
#include "utils.h"
#include "hash_map_set.h"
#include "cache_perf_model.h"
#include "shmem_perf_model.h"
#include "log.h"
#include "core.h"
#include "fault_injection.h"
```

Classes

- class **Cache**

Functions

- template<class T >
  UInt32 **moduloHashFn** (T key, **UInt32** hash_fn_param, **UInt32** num_buckets)

7.27.1 Function Documentation

7.27.1.1 moduloHashFn()

```
template<class T >
UInt32 moduloHashFn (
            T key,
             UInt32 hash_fn_param,
             UInt32 num_buckets )
```

Definition at line 74 of file cache.h.

7.28 common/core/memory_subsystem/cache/cache_base.cc File Reference

```
#include "cache_base.h"
#include "utils.h"
#include "log.h"
#include "rng.h"
#include "address_home_lookup.h"
```

7.29 common/core/memory_subsystem/cache/cache_base.d File Reference

7.30 common/core/memory_subsystem/cache/cache_base.h File Reference

```
#include "fixed_types.h"
```

Classes

- class **CacheBase**

Macros

- #define **k_KILO** 1024
- #define **k_MEGA** (**k_KILO**∗ **k_KILO**)
- #define **k_GIGA** (**k_KILO**∗ **k_MEGA**)

7.30.1 Macro Definition Documentation

7.30.1.1 k_GIGA

```
#define k_GIGA ( k_KILO∗ k_MEGA)
```

Definition at line 10 of file cache_base.h.

7.30.1.2 k_KILO

```
#define k_KILO 1024
```

Definition at line 8 of file cache_base.h.

7.30.1.3 k_MEGA

```
#define k_MEGA ( k_KILO* k_KILO)
```

Definition at line 9 of file cache_base.h.

7.31 common/core/memory_subsystem/cache/cache_block_info.cc File Reference

```
#include "cache_block_info.h"
#include "pr_l1_cache_block_info.h"
#include "pr_l2_cache_block_info.h"
#include "shared_cache_block_info.h"
#include "log.h"
```

7.32 common/core/memory_subsystem/cache/cache_block_info.d File Reference

7.33 common/core/memory_subsystem/cache/cache_block_info.h File Reference

```
#include "fixed_types.h"
#include "cache_state.h"
#include "cache_base.h"
```

Classes

- class **CacheBlockInfo**
- class **CacheCntlr**

7.34 common/core/memory_subsystem/cache/cache_set.cc File Reference

```
#include "cache_set.h"
#include "cache_set_lru.h"
#include "cache_set_mru.h"
#include "cache_set_nmru.h"
#include "cache_set_nru.h"
#include "cache_set_plru.h"
#include "cache_set_random.h"
#include "cache_set_round_robin.h"
#include "cache_set_srrip.h"
#include "cache_set_kruger.h"
#include "cache_base.h"
#include "log.h"
#include "simulator.h"
#include "config.h"
#include "config.hpp"
```

7.35 common/core/memory_subsystem/cache/cache_set.d File Reference

7.36 common/core/memory_subsystem/cache/cache_set.h File Reference

```
#include "fixed_types.h"
#include "cache_block_info.h"
#include "cache_base.h"
#include "lock.h"
#include "random.h"
#include "log.h"
#include <cstring>
```

Classes

- class **CacheSetInfo**
- class **CacheSet**

7.37 common/core/memory_subsystem/cache/cache_set_kruger.cc File Reference

```
#include "cache_set_kruger.h"
#include "log.h"
#include "stats.h"
```

7.38 common/core/memory_subsystem/cache/cache_set_kruger.d File Reference

7.39 common/core/memory_subsystem/cache/cache_set_kruger.h File Reference

```
#include "cache_set.h"
#include "cache_set_lru.h"
```

Classes

- class **CacheSetKruger**

7.40 common/core/memory_subsystem/cache/cache_set_kruger_2.cc File Reference

```
#include "cache_set_kruger_2.h"
#include "log.h"
```

7.41 common/core/memory_subsystem/cache/cache_set_kruger_2.d File Reference

7.42 common/core/memory_subsystem/cache/cache_set_kruger_2.h File Reference

```
#include "cache_set.h"
```

Classes

- class **CacheSetKruger**

7.43 common/core/memory_subsystem/cache/cache_set_lru.cc File Reference

```
#include "cache_set_lru.h"
#include "log.h"
#include "stats.h"
```

7.44 common/core/memory_subsystem/cache/cache_set_lru.d File Reference

7.45 common/core/memory_subsystem/cache/cache_set_lru.h File Reference

```
#include "cache_set.h"
```

Classes

- class **CacheSetInfoLRU**
- class **CacheSetLRU**

7.46 common/core/memory_subsystem/cache/cache_set_mru.cc File Reference

```
#include "cache_set_mru.h"
#include "log.h"
```

7.47 common/core/memory_subsystem/cache/cache_set_mru.d File Reference

7.48 common/core/memory_subsystem/cache/cache_set_mru.h File Reference

```
#include "cache_set.h"
```

Classes

- class **CacheSetMRU**

7.49 common/core/memory_subsystem/cache/cache_set_nmru.cc File Reference

```
#include "cache_set_nmru.h"
#include "log.h"
```

7.50 common/core/memory_subsystem/cache/cache_set_nmru.d File Reference

7.51 common/core/memory_subsystem/cache/cache_set_nmru.h File Reference

```
#include "cache_set.h"
```

Classes

- class **CacheSetNMRU**

7.52 common/core/memory_subsystem/cache/cache_set_nru.cc File Reference

```
#include "cache_set_nru.h"
#include "log.h"
```

7.53 common/core/memory_subsystem/cache/cache_set_nru.d File Reference

7.54 common/core/memory_subsystem/cache/cache_set_nru.h File Reference

```
#include "cache_set.h"
```

Classes

- class **CacheSetNRU**

7.55 common/core/memory_subsystem/cache/cache_set_plru.cc File Reference

```
#include "cache_set_plru.h"
#include "log.h"
```

7.56 common/core/memory_subsystem/cache/cache_set_plru.d File Reference

7.57 common/core/memory_subsystem/cache/cache_set_plru.h File Reference

```
#include "cache_set.h"
```

Classes

- class **CacheSetPLRU**

7.58 common/core/memory_subsystem/cache/cache_set_random.cc File Reference

```
#include "cache_set_random.h"
#include "log.h"
#include <time.h>
```

7.59 common/core/memory_subsystem/cache/cache_set_random.d File Reference

7.60 common/core/memory_subsystem/cache/cache_set_random.h File Reference

```
#include "cache_set.h"
```

Classes

- class **CacheSetRandom**

7.61 common/core/memory_subsystem/cache/cache_set_round_↩ robin.cc File Reference

```
#include "cache_set_round_robin.h"
```

7.62 common/core/memory_subsystem/cache/cache_set_round_robin.d File Reference

7.63 common/core/memory_subsystem/cache/cache_set_round_robin.h File Reference

```
#include "cache_set.h"
```

Classes

- class **CacheSetRoundRobin**

7.64 common/core/memory_subsystem/cache/cache_set_srrip.cc File Reference

```
#include "cache_set_srrip.h"
#include "simulator.h"
#include "config.hpp"
#include "log.h"
```

7.65 common/core/memory_subsystem/cache/cache_set_srrip.d File Reference

7.66 common/core/memory_subsystem/cache/cache_set_srrip.h File Reference

```
#include "cache_set.h"
#include "cache_set_lru.h"
```

Classes

- class **CacheSetSRRIP**

7.67 common/core/memory_subsystem/cache/cache_state.h File Reference

```
#include <cassert>
#include "fixed_types.h"
```

Classes

- class **CacheState**

7.68 common/core/memory_subsystem/cache/pr_l1_cache_block_info.h File Reference

```
#include "cache_state.h"
#include "cache_block_info.h"
```

Classes

- class **PrL1CacheBlockInfo**

7.69 common/core/memory_subsystem/cache/pr_l2_cache_block_↩ info.cc File Reference

```
#include "pr_l2_cache_block_info.h"
#include "log.h"
```

7.70 common/core/memory_subsystem/cache/pr_l2_cache_block_info.d File Reference

7.71 common/core/memory_subsystem/cache/pr_l2_cache_block_info.h File Reference

```
#include "cache_state.h"
#include "cache_block_info.h"
#include "mem_component.h"
```

Classes

- class **PrL2CacheBlockInfo**

7.72 common/core/memory_subsystem/cache/req_queue_list_↩ template.h File Reference

```
#include <map>
#include <queue>
#include "log.h"
```

Classes

- class **ReqQueueListTemplate**< **T_Req** >

7.73 common/core/memory_subsystem/cache/shared_cache_block_↩ info.cc File Reference

```
#include "shared_cache_block_info.h"
#include "log.h"
```

7.74 common/core/memory_subsystem/cache/shared_cache_block_↩ info.d File Reference

7.75 common/core/memory_subsystem/cache/shared_cache_block_↩ info.h File Reference

```
#include "cache_state.h"
#include "cache_block_info.h"
#include "mem_component.h"
```

Classes

- class **SharedCacheBlockInfo**

7.76 common/core/memory_subsystem/cheetah/cheetah_manager.cc File Reference

```
#include "cheetah_manager.h"
#include "cheetah_model.h"
#include "simulator.h"
#include "config.hpp"
#include "core_manager.h"
#include "hooks_manager.h"
#include "stats.h"
```

7.77 common/core/memory_subsystem/cheetah/cheetah_manager.d File Reference

7.78 common/core/memory_subsystem/cheetah/cheetah_manager.h File Reference

```
#include "fixed_types.h"
#include "core.h"
```

Classes

- class **CheetahManager**
- class **CheetahManager::CheetahStats**

7.79 common/core/memory_subsystem/cheetah/cheetah_model.cc File Reference

```
#include "cheetah_model.h"
```

7.80 common/core/memory_subsystem/cheetah/cheetah_model.d File Reference

7.81 common/core/memory_subsystem/cheetah/cheetah_model.h File Reference

```
#include "fixed_types.h"
#include "saclru.h"
#include "lock.h"
#include <vector>
```

Classes

- class **CheetahModel**

7.82 common/core/memory_subsystem/cheetah/saclru.cc File Reference

```
#include "saclru.h"
#include "util.h"
#include <cstdlib>
#include <cassert>
```

Macros

- #define **ONE** 1U
- #define **TWO** 2
- #define **B80000000** 0x80000000
- #define **INVALID** 0
- #define **MEM_AVAIL_HITARR** 2097152 /∗ Memory available for hitarr ∗/

7.82.1 Macro Definition Documentation

7.82.1.1 B80000000

```
#define B80000000 0x80000000
```

Definition at line 23 of file saclru.cc.

7.82.1.2 INVALID

```
#define INVALID 0
```

Definition at line 24 of file saclru.cc.

7.82.1.3 MEM_AVAIL_HITARR

```
#define MEM_AVAIL_HITARR 2097152 /* Memory available for hitarr */
```

Definition at line 25 of file saclru.cc.

7.82.1.4 ONE

```
#define ONE 1U
```

Definition at line 21 of file saclru.cc.

7.82.1.5 TWO

```
#define TWO 2
```

Definition at line 22 of file saclru.cc.

7.83 common/core/memory_subsystem/cheetah/saclru.d File Reference

7.84 common/core/memory_subsystem/cheetah/saclru.h File Reference

```
#include <cstdint>
#include <cstdio>
```

Classes

- class **CheetahSACLRU**

7.85 common/core/memory_subsystem/cheetah/util.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "util.h"
```

Functions

- void **fatal** (const char *msg)
- int **power** (int x, int y)
- uint64_t ** **idim2** (int row, int col)
- void **UHT_Add_to_free_list** (struct **hash_table** *free_ptr)
- struct **hash_table** * **UHT_Get_from_free_list** (void)
- void **rotate_left** (int y, struct **tree_node** **p_stack)
- void **rotate_right** (int y, struct **tree_node** **p_stack)
- void **splay** (int at, struct **tree_node** **p_stack)

Variables

- static struct **hash_table** * **head_free_list**

7.85.1 Function Documentation

7.85.1.1 fatal()

```
void fatal (
            const char * msg )
```

Definition at line 25 of file util.cc.

Referenced by idim2(), and CheetahSACLRU::init_saclru().

7.85.1.2 idim2()

```
uint64_t** idim2 (
            int row,
            int col )
```

Definition at line 62 of file util.cc.

References fatal().

Referenced by CheetahSACLRU::init_saclru().

7.85.1.3 power()

```
int power (
            int x,
            int y )
```

Definition at line 39 of file util.cc.

Referenced by CheetahSACLRU::init_saclru().

7.85.1.4 rotate_left()

```
void rotate_left (
            int y,
            struct  tree_node ** p_stack )
```

Definition at line 124 of file util.cc.

References tree_node::lft, tree_node::rt, and tree_node::rtwt.

Referenced by splay().

7.85.1.5 rotate_right()

```
void rotate_right (
            int y,
            struct  tree_node ** p_stack )
```

Definition at line 154 of file util.cc.

References tree_node::lft, tree_node::rt, and tree_node::rtwt.

Referenced by splay().

7.85.1.6 splay()

```
void splay (
            int at,
            struct  tree_node ** p_stack )
```

Definition at line 189 of file util.cc.

References tree_node::lft, rotate_left(), rotate_right(), and tree_node::rt.

7.85.1.7 UHT_Add_to_free_list()

```
void UHT_Add_to_free_list (
            struct  hash_table * free_ptr )
```

Definition at line 92 of file util.cc.

References head_free_list, and hash_table::nxt.

7.85.1.8 UHT_Get_from_free_list()

```
struct hash_table* UHT_Get_from_free_list (
            void )
```

Definition at line 99 of file util.cc.

References head_free_list, and hash_table::nxt.

7.85.2 Variable Documentation

7.85.2.1 head_free_list

```
struct hash_table* head_free_list [static]
```

Definition at line 89 of file util.cc.

Referenced by UHT_Add_to_free_list(), and UHT_Get_from_free_list().

7.86 common/core/memory_subsystem/cheetah/util.d File Reference

7.87 common/core/memory_subsystem/cheetah/util.h File Reference

```
#include <stdint.h>
```

Classes

- struct **hash_table**
- struct **tree_node**

Functions

- void **fatal** (const char *msg)
- int **power** (int x, int y)
- uint64_t ** **idim2** (int row, int col)
- void **UHT_Add_to_free_list** (struct **hash_table** *free_ptr)
- struct **hash_table** * **UHT_Get_from_free_list** (void)
- void **splay** (int at, struct **tree_node** **p_stack)

7.87.1 Function Documentation

7.87.1.1 fatal()

```
void fatal (
            const char * msg )
```

Definition at line 25 of file util.cc.

Referenced by idim2(), and CheetahSACLRU::init_saclru().

7.87.1.2 idim2()

```
uint64_t** idim2 (
            int row,
            int col )
```

Definition at line 62 of file util.cc.

References fatal().

Referenced by CheetahSACLRU::init_saclru().

7.87.1.3 power()

```
int power (
            int x,
            int y )
```

Definition at line 39 of file util.cc.

Referenced by CheetahSACLRU::init_saclru().

7.87.1.4 splay()

```
void splay (
            int at,
            struct  tree_node ** p_stack )
```

Definition at line 189 of file util.cc.

References tree_node::lft, rotate_left(), rotate_right(), and tree_node::rt.

7.87.1.5 UHT_Add_to_free_list()

```
void UHT_Add_to_free_list (
            struct  hash_table * free_ptr )
```

Definition at line 92 of file util.cc.

References head_free_list, and hash_table::nxt.

7.87.1.6 UHT_Get_from_free_list()

```
struct  hash_table* UHT_Get_from_free_list (
            void  )
```

Definition at line 99 of file util.cc.

References head_free_list, and hash_table::nxt.

7.88 common/core/memory_subsystem/directory_schemes/coherency_protocol.h File Reference

Classes

- class **CoherencyProtocol**

7.89 common/core/memory_subsystem/directory_schemes/directory.cc File Reference

```
#include "simulator.h"
#include "directory.h"
#include "directory_entry.h"
#include "directory_entry_limited_no_broadcast.h"
#include "directory_entry_limitless.h"
#include "stats.h"
#include "log.h"
#include "config.hpp"
```

7.90 common/core/memory_subsystem/directory_schemes/directory.d File Reference

7.91 common/core/memory_subsystem/directory_schemes/directory.h File Reference

```
#include "directory_entry.h"
#include "fixed_types.h"
#include "subsecond_time.h"
```

Classes

- class **Directory**

7.92 common/core/memory_subsystem/directory_schemes/directory_↩ block_info.h File Reference

```
#include "directory_state.h"
```

Classes

- class **DirectoryBlockInfo**

7.93 common/core/memory_subsystem/directory_schemes/directory_↩ entry.h File Reference

```
#include "fixed_types.h"
#include "directory_block_info.h"
#include "subsecond_time.h"
#include <vector>
#include <bitset>
#include <cassert>
```

Classes

- class **DirectorySharersBitset**< **Size** >
- class **DirectorySharersVector**
- class **DirectoryEntry**
- class **DirectoryEntrySized**< **DirectorySharers** >

7.94 common/core/memory_subsystem/directory_schemes/directory_↩ entry_limited_no_broadcast.h File Reference

```
#include "directory_entry.h"
#include "random.h"
#include "log.h"
```

Classes

- class **DirectoryEntryLimitedNoBroadcast**< **DirectorySharers** >

7.95 common/core/memory_subsystem/directory_schemes/directory_entry_limitless.h File Reference

```
#include "directory_entry.h"
#include "subsecond_time.h"
```

Classes

- class **DirectoryEntryLimitless**< **DirectorySharers** >

7.96 common/core/memory_subsystem/directory_schemes/directory_state.h File Reference

Classes

- class **DirectoryState**

7.97 common/core/memory_subsystem/dram/dram_cache.cc File Reference

```
#include "dram_cache.h"
#include "simulator.h"
#include "config.hpp"
#include "cache.h"
#include "stats.h"
#include "memory_manager_base.h"
#include "pr_l1_cache_block_info.h"
#include "queue_model.h"
#include "shmem_perf.h"
#include "prefetcher.h"
```

7.98 common/core/memory_subsystem/dram/dram_cache.d File Reference

7.99 common/core/memory_subsystem/dram/dram_cache.h File Reference

```
#include "dram_cntlr_interface.h"
#include "subsecond_time.h"
#include "cache.h"
#include "contention_model.h"
```

Classes

- class **DramCache**

7.100 common/core/memory_subsystem/dram/dram_cntlr_interface.cc File Reference

```
#include "dram_cntlr_interface.h"
#include "memory_manager.h"
#include "shmem_msg.h"
#include "shmem_perf.h"
#include "log.h"
```

7.101 common/core/memory_subsystem/dram/dram_cntlr_interface.d File Reference

7.102 common/core/memory_subsystem/dram/dram_cntlr_interface.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include "hit_where.h"
#include "shmem_msg.h"
#include "boost/tuple/tuple.hpp"
```

Classes

- class **DramCntlrInterface**

7.103 common/core/memory_subsystem/fast_nehalem/fast_cache.h File Reference

```
#include "memory_manager_fast.h"
```

Classes

- class **FastNehalem::Dram**
- class **FastNehalem::CacheSet**< **assoc** >
- class **FastNehalem::Cache**< **assoc, size_kb** >
- class **FastNehalem::CacheLocked**< **assoc, size_kb** >

Namespaces

- **FastNehalem**

7.104 common/core/memory_subsystem/fast_nehalem/memory_↩ manager.cc File Reference

```
#include "memory_manager.h"
#include "stats.h"
#include "log.h"
#include "utils.h"
#include "fast_cache.h"
```

Namespaces

- **FastNehalem**

7.105 common/core/memory_subsystem/parametric_dram_directory_↩ msi/memory_manager.cc File Reference

```
#include "core_manager.h"
#include "memory_manager.h"
#include "cache_base.h"
#include "nuca_cache.h"
#include "dram_cache.h"
#include "tlb.h"
#include "simulator.h"
#include "log.h"
#include "dvfs_manager.h"
#include "itostr.h"
#include "instruction.h"
#include "config.hpp"
#include "distribution.h"
#include "topology_info.h"
#include <algorithm>
```

Namespaces

- **ParametricDramDirectoryMSI**

Macros

- #define **MYLOG**(...) {}

7.105.1 Macro Definition Documentation

7.105.1.1 MYLOG

```
#define MYLOG(
              ...  ) {}
```

Definition at line 24 of file memory_manager.cc.

7.106 common/core/memory_subsystem/fast_nehalem/memory_↩ manager.d File Reference

7.107 common/core/memory_subsystem/parametric_dram_directory_↩ msi/memory_manager.d File Reference

7.108 common/core/memory_subsystem/fast_nehalem/memory_↩ manager.h File Reference

```
#include "memory_manager_fast.h"
```

Classes

- class **FastNehalem::CacheBase**
- class **FastNehalem::MemoryManager**

Namespaces

- **FastNehalem**

7.109 common/core/memory_subsystem/parametric_dram_directory_msi/memory_manager.h File Reference

```
#include "memory_manager_base.h"
#include "cache_base.h"
#include "cache_cntlr.h"
#include "../pr_l1_pr_l2_dram_directory_msi/dram_directory_cntlr.h"
#include "../pr_l1_pr_l2_dram_directory_msi/dram_cntlr.h"
#include "address_home_lookup.h"
#include "../pr_l1_pr_l2_dram_directory_msi/shmem_msg.h"
#include "mem_component.h"
#include "semaphore.h"
#include "fixed_types.h"
#include "shmem_perf_model.h"
#include "shared_cache_block_info.h"
#include "subsecond_time.h"
#include <map>
```

Classes

- class **ParametricDramDirectoryMSI::MemoryManager**

Namespaces

- **ParametricDramDirectoryMSI**

Typedefs

- typedef std::pair< **core_id_t**, **MemComponent::component_t** > **ParametricDramDirectoryMSI::CoreComponentType**
- typedef std::map< CoreComponentType, **CacheCntlr** * > **ParametricDramDirectoryMSI::CacheCntlrMap**

7.110 common/core/memory_subsystem/mem_component.cc File Reference

```
#include "mem_component.h"
```

Functions

- const char * **MemComponentString** (**MemComponent::component_t** mem_component)

7.110.1 Function Documentation

7.110.1.1 MemComponentString()

```
const char* MemComponentString (
            MemComponent::component_t mem_component )
```

Definition at line 3 of file mem_component.cc.

References MemComponent::CORE, MemComponent::DRAM, MemComponent::DRAM_CACHE, MemComponent::L1_DCACHE, MemComponent::L1_ICACHE, MemComponent::L2_CACHE, MemComponent::L3_CACHE, MemComponent::L4_CACHE, MemComponent::NUCA_CACHE, and MemComponent::TAG_DIR.

Referenced by FaultInjectorRandom::preRead().

7.111 common/core/memory_subsystem/mem_component.d File Reference

7.112 common/core/memory_subsystem/mem_component.h File Reference

Classes

- class **MemComponent**

Functions

- const char * **MemComponentString** (**MemComponent::component_t** mem_component)

7.112.1 Function Documentation

7.112.1.1 MemComponentString()

```
const char* MemComponentString (
            MemComponent::component_t mem_component )
```

Definition at line 3 of file mem_component.cc.

References MemComponent::CORE, MemComponent::DRAM, MemComponent::DRAM_CACHE, MemComponent::L1_DCACHE, MemComponent::L1_ICACHE, MemComponent::L2_CACHE, MemComponent::L3_CACHE, MemComponent::L4_CACHE, MemComponent::NUCA_CACHE, and MemComponent::TAG_DIR.

Referenced by FaultInjectorRandom::preRead().

7.113 common/core/memory_subsystem/memory_manager_base.cc File Reference

```
#include "simulator.h"
#include "config.h"
#include "memory_manager_base.h"
#include "parametric_dram_directory_msi/memory_manager.h"
#include "fast_nehalem/memory_manager.h"
#include "log.h"
#include "config.hpp"
```

Functions

- void **MemoryManagerNetworkCallback** (void *obj, **NetPacket** packet)

7.113.1 Function Documentation

7.113.1.1 MemoryManagerNetworkCallback()

```
void MemoryManagerNetworkCallback (
            void * obj,
             NetPacket packet )
```

Definition at line 40 of file memory_manager_base.cc.

References MemoryManagerBase::handleMsgFromNetwork(), LOG_PRINT_ERROR, SHARED_MEM_1, and NetPacket::type.

Referenced by ParametricDramDirectoryMSI::MemoryManager::MemoryManager().

7.114 common/core/memory_subsystem/memory_manager_base.d File Reference

7.115 common/core/memory_subsystem/memory_manager_base.h File Reference

```
#include "core.h"
#include "network.h"
#include "mem_component.h"
#include "performance_model.h"
#include "shmem_perf_model.h"
#include "pr_l1_pr_l2_dram_directory_msi/shmem_msg.h"
```

Classes

- class **MemoryManagerBase**

Functions

- void **MemoryManagerNetworkCallback** (void *obj, **NetPacket** packet)

7.115.1 Function Documentation

7.115.1.1 MemoryManagerNetworkCallback()

```
void MemoryManagerNetworkCallback (
            void * obj,
             NetPacket packet )
```

Definition at line 40 of file memory_manager_base.cc.

References MemoryManagerBase::handleMsgFromNetwork(), LOG_PRINT_ERROR, SHARED_MEM_1, and NetPacket::type.

Referenced by ParametricDramDirectoryMSI::MemoryManager::MemoryManager().

7.116 common/core/memory_subsystem/memory_manager_fast.h File Reference

```
#include "memory_manager_base.h"
#include "mem_component.h"
#include "fixed_types.h"
#include "subsecond_time.h"
```

Classes

- class **MemoryManagerFast**

7.117 common/core/memory_subsystem/parametric_dram_directory_msi/cache_atd.cc File Reference

```
#include "cache_atd.h"
#include "cache_set.h"
#include "pr_l1_cache_block_info.h"
#include "stats.h"
#include "config.hpp"
#include "rng.h"
```

7.118 common/core/memory_subsystem/parametric_dram_directory_↩ msi/cache_atd.d File Reference

7.119 common/core/memory_subsystem/parametric_dram_directory_↩ msi/cache_atd.h File Reference

```
#include "fixed_types.h"
#include "cache_base.h"
#include "cache_set.h"
#include "core.h"
#include <unordered_map>
```

Classes

- class **ATD**

7.120 common/core/memory_subsystem/parametric_dram_directory_↩ msi/cache_cntlr.cc File Reference

```
#include "cache_cntlr.h"
#include "log.h"
#include "memory_manager.h"
#include "core_manager.h"
#include "simulator.h"
#include "subsecond_time.h"
#include "config.hpp"
#include "fault_injection.h"
#include "hooks_manager.h"
#include "cache_atd.h"
#include "shmem_perf.h"
#include <cstring>
```

Namespaces

- **ParametricDramDirectoryMSI**

Macros

- #define **MYLOG**(...) {}

Functions

- char **ParametricDramDirectoryMSI::CStateString** (**CacheState::cstate_t** cstate)
- const char * **ParametricDramDirectoryMSI::ReasonString** (Transition::reason_t reason)
- MshrEntry **ParametricDramDirectoryMSI::make_mshr** (**SubsecondTime** t_issue, **SubsecondTime** t_↩ complete)

Variables

- **Lock iolock**

7.120.1 Macro Definition Documentation

7.120.1.1 MYLOG

```
#define MYLOG(
              ...  ) {}
```

Definition at line 32 of file cache_cntlr.cc.

7.120.2 Variable Documentation

7.120.2.1 iolock

```
Lock iolock
```

Definition at line 19 of file cache_cntlr.cc.

Referenced by Core::initiateMemoryAccess().

7.121 common/core/memory_subsystem/parametric_dram_directory_↩ msi/cache_cntlr.d File Reference

7.122 common/core/memory_subsystem/parametric_dram_directory_↩ msi/cache_cntlr.h File Reference

```
#include "core.h"
#include "cache.h"
#include "prefetcher.h"
#include "shared_cache_block_info.h"
#include "address_home_lookup.h"
#include "../pr_l1_pr_l2_dram_directory_msi/shmem_msg.h"
#include "mem_component.h"
#include "semaphore.h"
#include "lock.h"
#include "setlock.h"
#include "fixed_types.h"
#include "shmem_perf_model.h"
#include "contention_model.h"
#include "req_queue_list_template.h"
#include "stats.h"
#include "subsecond_time.h"
#include "shmem_perf.h"
#include "boost/tuple/tuple.hpp"
```

Classes

- class **ParametricDramDirectoryMSI::Transition**
- class **ParametricDramDirectoryMSI::Prefetch**
- class **ParametricDramDirectoryMSI::CacheParameters**
- class **ParametricDramDirectoryMSI::CacheCntlrList**
- class **ParametricDramDirectoryMSI::CacheDirectoryWaiter**
- struct **ParametricDramDirectoryMSI::MshrEntry**
- class **ParametricDramDirectoryMSI::CacheMasterCntlr**
- class **ParametricDramDirectoryMSI::CacheCntlr**

Namespaces

- **ParametricDramDirectoryMSI**

Macros

- #define **PREFETCH_MAX_QUEUE_LENGTH** 32
- #define **PREFETCH_INTERVAL** **SubsecondTime::NS**(1)

Typedefs

- typedef **ReqQueueListTemplate**< CacheDirectoryWaiter > **ParametricDramDirectoryMSI::Cache↩ DirectoryWaiterMap**
- typedef std::unordered_map< **IntPtr**, MshrEntry > **ParametricDramDirectoryMSI::Mshr**

7.122.1 Macro Definition Documentation

7.122.1.1 PREFETCH_INTERVAL

```
#define PREFETCH_INTERVAL  SubsecondTime::NS(1)
```

Definition at line 44 of file cache_cntlr.h.

7.122.1.2 PREFETCH_MAX_QUEUE_LENGTH

```
#define PREFETCH_MAX_QUEUE_LENGTH 32
```

Definition at line 42 of file cache_cntlr.h.

7.123 common/core/memory_subsystem/parametric_dram_directory_↩ msi/ghb_prefetcher.cc File Reference

```
#include "ghb_prefetcher.h"
#include "simulator.h"
#include "config.hpp"
#include <algorithm>
```

7.124 common/core/memory_subsystem/parametric_dram_directory_↩ msi/ghb_prefetcher.d File Reference

7.125 common/core/memory_subsystem/parametric_dram_directory_↩ msi/ghb_prefetcher.h File Reference

```
#include "prefetcher.h"
```

Classes

- class **GhbPrefetcher**
- struct **GhbPrefetcher::GHBEntry**
- struct **GhbPrefetcher::TableEntry**

7.126 common/core/memory_subsystem/parametric_dram_directory_↩ msi/nuca_cache.cc File Reference

```
#include "nuca_cache.h"
#include "memory_manager_base.h"
#include "pr_l1_cache_block_info.h"
#include "config.hpp"
#include "stats.h"
#include "queue_model.h"
#include "shmem_perf.h"
```

7.127 common/core/memory_subsystem/parametric_dram_directory_↩ msi/nuca_cache.d File Reference

7.128 common/core/memory_subsystem/parametric_dram_directory_↩ msi/nuca_cache.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include "hit_where.h"
#include "cache_cntlr.h"
#include "boost/tuple/tuple.hpp"
```

Classes

- class **NucaCache**

7.129 common/core/memory_subsystem/parametric_dram_directory_↩ msi/prefetcher.cc File Reference

```
#include "prefetcher.h"
#include "simulator.h"
#include "config.hpp"
#include "log.h"
#include "simple_prefetcher.h"
#include "ghb_prefetcher.h"
```

7.130 common/core/memory_subsystem/parametric_dram_directory_msi/prefetcher.d File Reference

7.131 common/core/memory_subsystem/parametric_dram_directory_msi/prefetcher.h File Reference

```
#include "fixed_types.h"
#include <vector>
```

Classes

- class **Prefetcher**

7.132 common/core/memory_subsystem/parametric_dram_directory_msi/simple_prefetcher.cc File Reference

```
#include "simple_prefetcher.h"
#include "simulator.h"
#include "config.hpp"
#include <cstdlib>
```

Variables

- const **IntPtr** **PAGE_SIZE** = 4096
- const **IntPtr** **PAGE_MASK** = ∼(**PAGE_SIZE**-1)

7.132.1 Variable Documentation

7.132.1.1 PAGE_MASK

```
const  IntPtr PAGE_MASK = ∼( PAGE_SIZE-1)
```

Definition at line 8 of file simple_prefetcher.cc.

Referenced by SimplePrefetcher::getNextAddress().

7.132.1.2 PAGE_SIZE

```
const IntPtr PAGE_SIZE = 4096
```

Definition at line 7 of file simple_prefetcher.cc.

Referenced by SimplePrefetcher::getNextAddress(), MemGuard::isAligned(), MemGuard::pagePointer(), MemGuard::protect(), and MemGuard::unprotect().

7.133 common/core/memory_subsystem/parametric_dram_directory_msi/simple_prefetcher.d File Reference

7.134 common/core/memory_subsystem/parametric_dram_directory_msi/simple_prefetcher.h File Reference

```
#include "prefetcher.h"
```

Classes

- class **SimplePrefetcher**

7.135 common/core/memory_subsystem/parametric_dram_directory_msi/tlb.cc File Reference

```
#include "tlb.h"
#include "stats.h"
```

Namespaces

- **ParametricDramDirectoryMSI**

7.136 common/core/memory_subsystem/parametric_dram_directory_msi/tlb.d File Reference

7.137 common/core/memory_subsystem/parametric_dram_directory_msi/tlb.h File Reference

```
#include "fixed_types.h"
#include "cache.h"
```

Classes

- class **ParametricDramDirectoryMSI::TLB**

Namespaces

- **ParametricDramDirectoryMSI**

7.138 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/dram_cntlr.cc File Reference

```
#include "dram_cntlr.h"
#include "memory_manager.h"
#include "core.h"
#include "log.h"
#include "subsecond_time.h"
#include "stats.h"
#include "fault_injection.h"
#include "shmem_perf.h"
```

Namespaces

- **PrL1PrL2DramDirectoryMSI**

Macros

- #define **MYLOG**(...) {}

7.138.1 Macro Definition Documentation

7.138.1.1 MYLOG

```
#define MYLOG(
             ...  ) {}
```

Definition at line 16 of file dram_cntlr.cc.

7.139 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/dram_cntlr.d File Reference

7.140 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/dram_cntlr.h File Reference

```
#include <unordered_map>
#include "dram_perf_model.h"
#include "shmem_msg.h"
#include "shmem_perf.h"
#include "fixed_types.h"
#include "memory_manager_base.h"
#include "dram_cntlr_interface.h"
#include "subsecond_time.h"
```

Classes

- class **PrL1PrL2DramDirectoryMSI::DramCntlr**

Namespaces

- **PrL1PrL2DramDirectoryMSI**

7.141 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/dram_directory_cache.cc File Reference

```
#include "dram_directory_cache.h"
#include "log.h"
#include "utils.h"
```

Namespaces

- **PrL1PrL2DramDirectoryMSI**

7.142 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/dram_directory_cache.d File Reference

7.143 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/dram_directory_cache.h File Reference

```
#include <vector>
#include "directory.h"
#include "shmem_perf_model.h"
#include "subsecond_time.h"
```

Classes

- class **PrL1PrL2DramDirectoryMSI::DramDirectoryCache**

Namespaces

- **PrL1PrL2DramDirectoryMSI**

7.144 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/dram_directory_cntlr.cc File Reference

```
#include "dram_directory_cntlr.h"
#include "log.h"
#include "memory_manager.h"
#include "stats.h"
#include "nuca_cache.h"
#include "shmem_perf.h"
#include "coherency_protocol.h"
#include "config.hpp"
```

Namespaces

- **PrL1PrL2DramDirectoryMSI**

Macros

- #define **MYLOG**(...) {}

Functions

- char **PrL1PrL2DramDirectoryMSI::DStateString** (**DirectoryState::dstate_t** state)

7.144.1 Macro Definition Documentation

7.144.1.1 MYLOG

```
#define MYLOG(
              ...  ) {}
```

Definition at line 16 of file dram_directory_cntlr.cc.

7.145 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_↩ msi/dram_directory_cntlr.d File Reference

7.146 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_↩ msi/dram_directory_cntlr.h File Reference

```
#include "dram_directory_cache.h"
#include "req_queue_list.h"
#include "dram_cntlr.h"
#include "address_home_lookup.h"
#include "shmem_req.h"
#include "shmem_msg.h"
#include "shmem_perf.h"
#include "mem_component.h"
#include "memory_manager_base.h"
#include "coherency_protocol.h"
```

Classes

- class **PrL1PrL2DramDirectoryMSI::DramDirectoryCntlr**

Namespaces

- **PrL1PrL2DramDirectoryMSI**

7.147 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_↩ msi/req_queue_list.h File Reference

```
#include <map>
#include <queue>
#include "shmem_req.h"
#include "req_queue_list_template.h"
```

Namespaces

- **PrL1PrL2DramDirectoryMSI**

Typedefs

- typedef **ReqQueueListTemplate**< ShmemReq > **PrL1PrL2DramDirectoryMSI::ReqQueueList**

7.148 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_↩ msi/shmem_msg.cc File Reference

```
#include <string.h>
#include "shmem_msg.h"
#include "shmem_perf.h"
#include "log.h"
```

Namespaces

- **PrL1PrL2DramDirectoryMSI**

7.149 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_↩ msi/shmem_msg.d File Reference

7.150 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_↩ msi/shmem_msg.h File Reference

```
#include "mem_component.h"
#include "fixed_types.h"
#include "hit_where.h"
```

Classes

- class **PrL1PrL2DramDirectoryMSI::ShmemMsg**

Namespaces

- **PrL1PrL2DramDirectoryMSI**

7.151 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_↩ msi/shmem_perf.cc File Reference

```
#include "shmem_perf.h"
#include "network.h"
#include "log.h"
```

Functions

- const char * **ShmemReasonString** (**ShmemPerf::shmem_times_type_t** reason)

Variables

- const char * **shmem_reason_names** [ ]

7.151.1 Function Documentation

7.151.1.1 ShmemReasonString()

```
const char* ShmemReasonString (
              ShmemPerf::shmem_times_type_t reason )
```

Definition at line 33 of file shmem_perf.cc.

References LOG_ASSERT_ERROR, ShmemPerf::NUM_SHMEM_TIMES, and shmem_reason_names.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr().

7.151.2 Variable Documentation

7.151.2.1 shmem_reason_names

```
const char* shmem_reason_names[]
```

Initial value:

```
= {
   "noc-base",
   "noc-queue",
   "td-access",
   "inv-imbalance",
   "remote-cache-inv",
   "remote-cache-fwd",
   "remote-cache-wb",
   "pending-hit",
   "nuca-tags",
   "nuca-bus",
   "nuca-queue",
   "nuca-data",
   "dram-cache",
   "dram-cache-tags",
   "dram-cache-queue",
   "dram-cache-bus",
   "dram-cache-data",
   "dram",
   "dram-queue",
   "dram-bus",
   "dram-device",
   "unknown",
}
```

Definition at line 5 of file shmem_perf.cc.

Referenced by ShmemReasonString().

7.152 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/shmem_perf.d File Reference

7.153 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/shmem_perf.h File Reference

```
#include "subsecond_time.h"
#include <vector>
```

Classes

- class **ShmemPerf**

Functions

- const char * **ShmemReasonString** (**ShmemPerf::shmem_times_type_t** reason)

7.153.1 Function Documentation

7.153.1.1 ShmemReasonString()

```
const char* ShmemReasonString (
              ShmemPerf::shmem_times_type_t reason )
```

Definition at line 33 of file shmem_perf.cc.

References LOG_ASSERT_ERROR, ShmemPerf::NUM_SHMEM_TIMES, and shmem_reason_names.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr().

7.154 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/shmem_req.cc File Reference

```
#include "shmem_req.h"
#include "log.h"
```

Namespaces

- **PrL1PrL2DramDirectoryMSI**

7.155 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/shmem_req.d File Reference

7.156 common/core/memory_subsystem/pr_l1_pr_l2_dram_directory_msi/shmem_req.h File Reference

```
#include "shmem_msg.h"
#include "fixed_types.h"
#include "subsecond_time.h"
```

Classes

- class **PrL1PrL2DramDirectoryMSI::ShmemReq**

Namespaces

- **PrL1PrL2DramDirectoryMSI**

7.157 common/core/syscall_model.cc File Reference

```
#include "syscall_model.h"
#include "config.h"
#include "simulator.h"
#include "syscall_server.h"
#include "thread.h"
#include "core.h"
#include "core_manager.h"
#include "thread_manager.h"
#include "performance_model.h"
#include "instruction.h"
#include "pthread_emu.h"
#include "scheduler.h"
#include "hooks_manager.h"
#include "stats.h"
#include "syscall_strings.h"
#include "circular_log.h"
#include <errno.h>
#include <sys/syscall.h>
#include <linux/futex.h>
#include "os_compat.h"
#include <boost/algorithm/string.hpp>
```

7.158 common/core/syscall_model.d File Reference

7.159 common/core/syscall_model.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include <iostream>
```

Classes

- class **SyscallMdl**
- struct **SyscallMdl::syscall_args_t**
- struct **SyscallMdl::HookSyscallEnter**
- struct **SyscallMdl::HookSyscallExit**
- struct **SyscallMdl::futex_counters_t**

7.160 common/core/thread.cc File Reference

```
#include "thread.h"
#include "simulator.h"
#include "syscall_model.h"
#include "thread_manager.h"
#include "core_manager.h"
```

```
#include "sync_client.h"
#include "performance_model.h"
#include "instruction.h"
#include "routine_tracer.h"
#include "config.hpp"
```

7.161 common/core/thread.d File Reference

7.162 common/core/thread.h File Reference

```
#include "cond.h"
#include "subsecond_time.h"
```

Classes

- class **Thread**

7.163 common/core/topology_info.cc File Reference

```
#include "topology_info.h"
#include "simulator.h"
#include "config.h"
#include "config.hpp"
#include "core_manager.h"
```

Macros

- #define **VERBOSE** 0

7.163.1 Macro Definition Documentation

7.163.1.1 VERBOSE

```
#define VERBOSE 0
```

Definition at line 7 of file topology_info.cc.

7.164 common/core/topology_info.d File Reference

7.165 common/core/topology_info.h File Reference

```
#include "fixed_types.h"
```

Classes

- class **TopologyInfo**

7.166 common/fault_injection/fault_injection.cc File Reference

```
#include "fault_injection.h"
#include "fault_injector_random.h"
#include "simulator.h"
#include "config.hpp"
#include "log.h"
```

7.167 common/fault_injection/fault_injection.d File Reference

7.168 common/fault_injection/fault_injection.h File Reference

```
#include "fixed_types.h"
#include "core.h"
```

Classes

- class **FaultinjectionManager**
- class **FaultInjector**

7.169 common/fault_injection/fault_injector_random.cc File Reference

```
#include "fault_injector_random.h"
#include "rng.h"
```

7.170 common/fault_injection/fault_injector_random.d File Reference

7.171 common/fault_injection/fault_injector_random.h File Reference

```
#include "fault_injection.h"
```

Classes

- class **FaultInjectorRandom**

7.172 common/misc/_thread.h File Reference

Classes

- class **Runnable**
- class **_Thread**

7.173 common/misc/allocator.h File Reference

```
#include "fixed_types.h"
#include "FSBAllocator.hh"
#include <typeinfo>
#include <cxxabi.h>
```

Classes

- class **Allocator**
- struct **Allocator::DataElement**
- class **TypedAllocator**< **T, MaxItems** >

7.174 common/misc/average.h File Reference

Functions

- template<class T >
  T::value_type **arithmetic_mean** (T &list)

7.174.1 Function Documentation

7.174.1.1 arithmetic_mean()

```
template<class T >
T::value_type arithmetic_mean (
            T & list )
```

Definition at line 6 of file average.h.

Referenced by PeriodicSampling::callbackDetailed().

7.175 common/misc/barrier.cc File Reference

```
#include "barrier.h"
#include "log.h"
```

7.176 common/misc/barrier.d File Reference

7.177 common/misc/barrier.h File Reference

```
#include "fixed_types.h"
#include "cond.h"
```

Classes

- class **Barrier**

7.178 common/misc/basic_hash.cc File Reference

```
#include "basic_hash.h"
```

7.179 common/misc/basic_hash.d File Reference

7.180 common/misc/basic_hash.h File Reference

```
#include "fixed_types.h"
#include <unordered_map>
#include <utility>
#include <iostream>
#include <assert.h>
```

Classes

- class **BasicHash**

7.181 common/misc/bit_vector.cc File Reference

```
#include "bit_vector.h"
```

7.182 common/misc/bit_vector.d File Reference

7.183 common/misc/bit_vector.h File Reference

```
#include "fixed_types.h"
#include <assert.h>
#include <vector>
```

Classes

- class **BitVector**

Macros

- #define **BITVECT_DEBUG** 0

7.183.1 Macro Definition Documentation

7.183.1.1 BITVECT_DEBUG

```
#define BITVECT_DEBUG 0
```

Definition at line 15 of file bit_vector.h.

7.184 common/misc/bottlegraph.cc File Reference

```
#include "bottlegraph.h"
#include "thread_manager.h"
#include "stats.h"
```

7.185 common/misc/bottlegraph.d File Reference

7.186 common/misc/bottlegraph.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include <vector>
```

Classes

- class **BottleGraphManager**

7.187 common/misc/callstack.cc File Reference

```
#include "callstack.h"
```

Classes

- struct **stack_frame**

Functions

- int **get_call_stack_from** (void **retaddrs, int max_size, void *sp, void *bp)
- int **get_call_stack** (void **retaddrs, int max_size)

Variables

- void * **__libc_stack_end**

7.187.1 Function Documentation

7.187.1.1 get_call_stack()

```
int get_call_stack (
            void ** retaddrs,
            int max_size )
```

Definition at line 26 of file callstack.cc.

References get_call_stack_from().

7.187.1.2 get_call_stack_from()

```
int get_call_stack_from (
             void ** retaddrs,
             int max_size,
             void * sp,
             void * bp )
```

Definition at line 9 of file callstack.cc.

References __libc_stack_end, stack_frame::next, and stack_frame::ret.

Referenced by exceptionHandler(), and get_call_stack().

7.187.2 Variable Documentation

7.187.2.1 __libc_stack_end

```
void* __libc_stack_end
```

Referenced by get_call_stack_from().

7.188 common/misc/callstack.d File Reference

7.189 common/misc/callstack.h File Reference

Functions

- int **get_call_stack** (void ∗∗retaddrs, int max_size)
- int **get_call_stack_from** (void ∗∗retaddrs, int max_size, void ∗sp, void ∗bp)

7.189.1 Function Documentation

7.189.1.1 get_call_stack()

```
int get_call_stack (
             void ** retaddrs,
             int max_size )
```

Definition at line 26 of file callstack.cc.

References get_call_stack_from().

7.189.1.2 get_call_stack_from()

```
int get_call_stack_from (
            void ** retaddrs,
            int max_size,
            void * sp,
            void * bp )
```

Definition at line 9 of file callstack.cc.

References __libc_stack_end, stack_frame::next, and stack_frame::ret.

Referenced by exceptionHandler(), and get_call_stack().

7.190 common/misc/checksum.cc File Reference

```
#include "checksum.h"
#include "log.h"
```

Functions

- **UInt64 computeCheckSum** (const **Byte** *buffer, **UInt32** length)

7.190.1 Function Documentation

7.190.1.1 computeCheckSum()

```
UInt64 computeCheckSum (
            const  Byte * buffer,
             UInt32 length )
```

Definition at line 4 of file checksum.cc.

7.191 common/misc/checksum.d File Reference

7.192 common/misc/checksum.h File Reference

```
#include "fixed_types.h"
```

Functions

- **UInt64 computeCheckSum** (const **Byte** *buffer, **UInt32** length)

7.192.1 Function Documentation

7.192.1.1 computeCheckSum()

```
UInt64 computeCheckSum (
            const  Byte * buffer,
             UInt32 length )
```

Definition at line 4 of file checksum.cc.

7.193 common/misc/circular_log.cc File Reference

```
#include "circular_log.h"
#include "log.h"
#include "simulator.h"
#include "hooks_manager.h"
#include <stdarg.h>
```

7.194 common/misc/circular_log.d File Reference

7.195 common/misc/circular_log.h File Reference

```
#include "fixed_types.h"
#include "lock.h"
#include "timer.h"
```

Classes

- class **CircularLog**
- struct **CircularLog::event_t**

Macros

- #define **CLOG**(type, ...)

7.195.1 Macro Definition Documentation

7.195.1.1 CLOG

```
#define CLOG(
              type,
              ...  )
```

Value:

```
      do { \
      if (CircularLog::g_singleton) \
         CircularLog::g_singleton->insert(type, __VA_ARGS__); \
   } while(0)
```

Definition at line 56 of file circular_log.h.

7.196 common/misc/circular_queue.h File Reference

```
#include <assert.h>
#include <string.h>
```

Classes

- class **CircularQueue< T >**
- class **CircularQueue< T >::iterator**

7.197 common/misc/cond.cc File Reference

```
#include "cond.h"
#include "os_compat.h"
#include <unistd.h>
#include <sys/syscall.h>
#include <linux/futex.h>
#include <asm/errno.h>
#include <limits.h>
```

7.198 common/misc/cond.d File Reference

7.199 common/misc/cond.h File Reference

```
#include "fixed_types.h"
#include "lock.h"
```

Classes

- class **ConditionVariable**

7.200 common/misc/config.cc File Reference

```
#include "config.h"
#include "network_model.h"
#include "packet_type.h"
#include "simulator.h"
#include "utils.h"
#include "config.hpp"
#include <unistd.h>
#include <sstream>
```

Macros

- #define **DEBUG**

7.200.1 Macro Definition Documentation

7.200.1.1 DEBUG

```
#define DEBUG
```

Definition at line 12 of file config.cc.

7.201 common/misc/config.h File Reference

```
#include "fixed_types.h"
#include "clock_skew_minimization_object.h"
#include "cache_efficiency_tracker.h"
#include <vector>
#include <set>
#include <unordered_map>
#include <iostream>
#include <cassert>
```

Classes

- class **Config**

7.202 common/misc/cpuid.h File Reference

Classes

- struct **cpuid_result_t**

Functions

- static void **cpuid** (**UInt32** eax, **UInt32** ecx, **cpuid_result_t** &res)

7.202.1 Function Documentation

7.202.1.1 cpuid()

```
static void cpuid (
              UInt32 eax,
              UInt32 ecx,
              cpuid_result_t & res )  [inline], [static]
```

Definition at line 9 of file cpuid.h.

References cpuid_result_t::eax, cpuid_result_t::ebx, cpuid_result_t::ecx, and cpuid_result_t::edx.

Referenced by Core::emulateCpuid().

7.203 common/misc/distribution.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include <tr1/random>
```

Classes

- class **FloatDistribution**
- class **NormalFloatDistribution**
- class **TimeDistribution**
- class **ConstantTimeDistribution**
- class **NormalTimeDistribution**

7.204 common/misc/fixed_point.h File Reference

```
#include "fixed_types.h"
#include <cassert>
#include <cstdlib>
```

Classes

- class **TFixedPoint**< **one** >

Typedefs

- typedef **TFixedPoint**< __UINT64_C(0x4000)> **FixedPoint**

Functions

- template<SInt64 one>
  TFixedPoint< one > **operator**/ (**SInt64** i, **TFixedPoint**< one > &fp)
- template<SInt64 one>
  std::ostream & **operator**<< (std::ostream &os, const **TFixedPoint**< one > &f)

7.204.1 Typedef Documentation

7.204.1.1 FixedPoint

```
typedef TFixedPoint<__UINT64_C(0x4000)> FixedPoint
```

Definition at line 48 of file fixed_point.h.

7.204.2 Function Documentation

7.204.2.1 operator/()

```
template<SInt64 one>
TFixedPoint<one> operator/ (
            SInt64 i,
            TFixedPoint< one > & fp ) [inline]
```

Definition at line 45 of file fixed_point.h.

7.204.2.2 operator<<()

```
template<SInt64 one>
std::ostream& operator<< (
            std::ostream & os,
            const TFixedPoint< one > & f )
```

Definition at line 51 of file fixed_point.h.

7.205 common/misc/fixed_types.h File Reference

```
#include <stdint.h>
#include <inttypes.h>
```

Macros

- #define **__STDC_LIMIT_MACROS**
- #define **__STDC_CONSTANT_MACROS**
- #define **__STDC_FORMAT_MACROS**
- #define **INVALID_THREAD_ID** ((**thread_id_t**) -1)
- #define **INVALID_APP_ID** ((**app_id_t**) -1)
- #define **INVALID_CORE_ID** ((**core_id_t**) -1)
- #define **INVALID_ADDRESS** ((**IntPtr**) -1)

Typedefs

- typedef uint64_t **UInt64**
- typedef uint32_t **UInt32**
- typedef uint16_t **UInt16**
- typedef uint8_t **UInt8**
- typedef int64_t **SInt64**
- typedef int32_t **SInt32**
- typedef int16_t **SInt16**
- typedef int8_t **SInt8**
- typedef **UInt8 Byte**
- typedef **UInt8 Boolean**
- typedef uintptr_t **IntPtr**
- typedef uintptr_t **carbon_reg_t**
- typedef **SInt32 thread_id_t**
- typedef **SInt32 app_id_t**
- typedef **SInt32 core_id_t**
- typedef **SInt32 carbon_thread_t**

7.205.1 Macro Definition Documentation

7.205.1.1 __STDC_CONSTANT_MACROS

```
#define __STDC_CONSTANT_MACROS
```

Definition at line 8 of file fixed_types.h.

7.205.1.2 __STDC_FORMAT_MACROS

```
#define __STDC_FORMAT_MACROS
```

Definition at line 11 of file fixed_types.h.

7.205.1.3 __STDC_LIMIT_MACROS

```
#define __STDC_LIMIT_MACROS
```

Definition at line 5 of file fixed_types.h.

7.205.1.4 INVALID_ADDRESS

```
#define INVALID_ADDRESS (( IntPtr) -1)
```

Definition at line 53 of file fixed_types.h.

7.205.1.5 INVALID_APP_ID

```
#define INVALID_APP_ID (( app_id_t) -1)
```

Definition at line 51 of file fixed_types.h.

7.205.1.6 INVALID_CORE_ID

```
#define INVALID_CORE_ID (( core_id_t) -1)
```

Definition at line 52 of file fixed_types.h.

7.205.1.7 INVALID_THREAD_ID

```
#define INVALID_THREAD_ID (( thread_id_t) -1)
```

Definition at line 50 of file fixed_types.h.

7.205.2 Typedef Documentation

7.205.2.1 app_id_t

```
typedef SInt32 app_id_t
```

Definition at line 46 of file fixed_types.h.

7.205.2.2 Boolean

```
typedef UInt8 Boolean
```

Definition at line 38 of file fixed_types.h.

7.205.2.3 Byte

```
typedef UInt8 Byte
```

Definition at line 37 of file fixed_types.h.

7.205.2.4 carbon_reg_t

```
typedef uintptr_t carbon_reg_t
```

Definition at line 42 of file fixed_types.h.

7.205.2.5 carbon_thread_t

```
typedef SInt32 carbon_thread_t
```

Definition at line 48 of file fixed_types.h.

7.205.2.6 core_id_t

```
typedef SInt32 core_id_t
```

Definition at line 47 of file fixed_types.h.

7.205.2.7 IntPtr

```
typedef uintptr_t IntPtr
```

Definition at line 40 of file fixed_types.h.

7.205.2.8 SInt16

```
typedef int16_t SInt16
```

Definition at line 34 of file fixed_types.h.

7.205.2.9 SInt32

```
typedef int32_t SInt32
```

Definition at line 33 of file fixed_types.h.

7.205.2.10 SInt64

```
typedef int64_t SInt64
```

Definition at line 32 of file fixed_types.h.

7.205.2.11 SInt8

```
typedef int8_t SInt8
```

Definition at line 35 of file fixed_types.h.

7.205.2.12 thread_id_t

```
typedef SInt32 thread_id_t
```

Definition at line 45 of file fixed_types.h.

7.205.2.13 UInt16

```
typedef uint16_t UInt16
```

Definition at line 29 of file fixed_types.h.

7.205.2.14 UInt32

```
typedef uint32_t UInt32
```

Definition at line 28 of file fixed_types.h.

7.205.2.15 UInt64

```
typedef uint64_t UInt64
```

Definition at line 27 of file fixed_types.h.

7.205.2.16 UInt8

```
typedef uint8_t UInt8
```

Definition at line 30 of file fixed_types.h.

7.206 common/misc/FSBAllocator.hh File Reference

```
#include "log.h"
#include <new>
#include <vector>
#include <typeinfo>
#include <cxxabi.h>
```

Classes

- class **UnspecifiedType**
- class **FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >**
- class **FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::MemBlock**
- struct **FSBAllocator_ElemAllocator< ElemSize, MaxElem, TypeToAlloc >::BlocksVector**

7.207 common/misc/fxsupport.cc File Reference

```
#include <stdlib.h>
#include "fxsupport.h"
#include "core_manager.h"
#include "simulator.h"
```

7.208 common/misc/fxsupport.d File Reference

7.209 common/misc/fxsupport.h File Reference

```
#include "fixed_types.h"
```

Classes

- class **Fxsupport**
- class **ScopedFxsave**

7.210 common/misc/handle_args.cc File Reference

```
#include "handle_args.h"
#include <stdio.h>
#include <stdlib.h>
#include <vector>
#include <stdarg.h>
#include <boost/algorithm/string/classification.hpp>
#include <boost/algorithm/string/predicate.hpp>
#include <boost/algorithm/string/split.hpp>
```

Functions

- void **handle_generic_arg** (const String &str, **config::ConfigFile** & **cfg**)
- void **handle_args** (const **string_vec** &args)
- void **usage_error** (const char *error_msg,...)
- void **parse_args** (**string_vec** &args, String &config_path, int argc, char **argv)
- void **handle_args** (const **string_vec** &args, **config::ConfigFile** & **cfg**)

Variables

- static char * **prog_name**

7.210.1 Function Documentation

7.210.1.1 handle_args() [1/2]

```
void handle_args (
            const  string_vec & args )
```

Referenced by main().

7.210.1.2 handle_args() [2/2]

```
void handle_args (
            const  string_vec & args,
             config::ConfigFile & cfg )
```

Definition at line 85 of file handle_args.cc.

References cfg, and handle_generic_arg().

7.210.1.3 handle_generic_arg()

```
void handle_generic_arg (
            const String & str,
             config::ConfigFile & cfg )
```

Definition at line 95 of file handle_args.cc.

References cfg, config::Config::load(), config::ConfigFile::loadConfigFromString(), and usage_error().

Referenced by handle_args().

7.210.1.4 parse_args()

```
void parse_args (
              string_vec & args,
             String & config_path,
             int argc,
             char ** argv )
```

Definition at line 32 of file handle_args.cc.

References prog_name, and usage_error().

Referenced by main().

7.210.1.5 usage_error()

```
void usage_error (
             const char * error_msg,
              ...  )
```

Definition at line 19 of file handle_args.cc.

References prog_name.

Referenced by handle_generic_arg(), and parse_args().

7.210.2 Variable Documentation

7.210.2.1 prog_name

```
char* prog_name  [static]
```

Definition at line 11 of file handle_args.cc.

Referenced by parse_args(), and usage_error().

7.211 common/misc/handle_args.d File Reference

7.212 common/misc/handle_args.h File Reference

```
#include "config_file.hpp"
```

Typedefs

- typedef std::vector< String > **string_vec**

Functions

- void **parse_args** (**string_vec** &args, String &config_path, int argc, char **argv)
- void **handle_args** (const **string_vec** &args, **config::ConfigFile** & **cfg**)

7.212.1 Typedef Documentation

7.212.1.1 string_vec

```
typedef std::vector< String >  string_vec
```

Definition at line 5 of file handle_args.h.

7.212.2 Function Documentation

7.212.2.1 handle_args()

```
void handle_args (
            const  string_vec & args,
             config::ConfigFile & cfg )
```

Definition at line 85 of file handle_args.cc.

References cfg, and handle_generic_arg().

7.212.2.2 parse_args()

```
void parse_args (
             string_vec & args,
            String & config_path,
            int argc,
            char ** argv )
```

Definition at line 32 of file handle_args.cc.

References prog_name, and usage_error().

Referenced by main().

7.213 common/misc/hash_map_set.h File Reference

```
#include <set>
#include "fixed_types.h"
```

Classes

- class **HashMapSet**< **T** >

7.214 common/misc/itostr.h File Reference

```
#include "fixed_types.h"
#include <sstream>
```

Functions

- template<typename T >
  String **itostr** (T val)

7.214.1 Function Documentation

7.214.1.1 itostr()

```
template<typename T >
String itostr (
            T val )
```

Definition at line 13 of file itostr.h.

Referenced by _SetLock::_SetLock(), BarrierSyncServer::barrierRelease(), BbvCount::BbvCount(), Cache↩SetInfoLRU::CacheSetInfoLRU(), NetworkModelEMeshHopByHop::computeLatency(), QueueModelBasic↩::computeQueueDelay(), QueueModelHistoryList::computeUsingHistoryList(), ConditionVariable::Condition↩Variable(), SyscallMdl::formatSyscall(), PrL1PrL2DramDirectoryMSI::DramCntlr::getDataFromDram(), Trace↩Manager::getFifoName(), QueueModelHistoryList::getQueueUtilization(), MicroOpPerformanceModel::handle↩Instruction(), ShmemPerfModel::incrElapsedTime(), ParametricDramDirectoryMSI::MemoryManager::incr↩ElapsedTime(), HooksPy::init(), RobSmtTimer::initializeThread(), Core::initiateMemoryAccess(), IntervalTimer↩::IntervalTimer(), ParametricDramDirectoryMSI::MemoryManager::MemoryManager(), MicroOpPerformance↩Model::MicroOpPerformanceModel(), Network::netPullFromTransport(), Network::netRecv(), Network::netSend(), NetworkModel::NetworkModel(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrev↩Cache(), RobTimer::RobTimer(), CoreThread::run(), SimThread::run(), TraceThread::run(), config::ConfigFile↩::saveAs(), SELock::SELock(), ShmemPerfModel::setElapsedTime(), TraceManager::setupTraceFiles(), Barrier↩SyncServer::synchronize(), threadStartCallback(), ThreadStatsManager::ThreadStats::ThreadStats(), TLock< LockCreator_Default >::TLock(), ShmemPerfModel::updateElapsedTime(), and ThreadManager::wakeUpWaiter().

7.215 common/misc/lock.h File Reference

```
#include "itostr.h"
```

Classes

- class **LockImplementation**
- class **LockCreator**
- class **BaseLock**
- class **TLock< T_LockCreator >**
- class **LockCreator_Default**
- class **LockCreator_RwLock**
- class **LockCreator_Spinlock**
- class **LockCreator_NullLock**
- class **ScopedLock**
- class **ScopedReadLock**

Typedefs

- typedef **TLock< LockCreator_Default > Lock**
- typedef **TLock< LockCreator_RwLock > RwLock**
- typedef **TLock< LockCreator_Spinlock > SpinLock**
- typedef **TLock< LockCreator_NullLock > NullLock**

7.215.1 Typedef Documentation

7.215.1.1 Lock

```
typedef TLock< LockCreator_Default> Lock
```

Definition at line 119 of file lock.h.

7.215.1.2 NullLock

```
typedef TLock< LockCreator_NullLock> NullLock
```

Definition at line 122 of file lock.h.

7.215.1.3 RwLock

```
typedef TLock< LockCreator_RwLock> RwLock
```

Definition at line 120 of file lock.h.

7.215.1.4 SpinLock

```
typedef TLock< LockCreator_Spinlock> SpinLock
```

Definition at line 121 of file lock.h.

7.216 common/misc/locked_hash.cc File Reference

```
#include "locked_hash.h"
```

7.217 common/misc/locked_hash.d File Reference

7.218 common/misc/locked_hash.h File Reference

```
#include "fixed_types.h"
#include "lock.h"
#include <unordered_map>
```

Classes

- class **LockedHash**

7.219 common/misc/lockfree_hash.cc File Reference

```
#include "lockfree_hash.h"
```

7.220 common/misc/lockfree_hash.d File Reference

7.221 common/misc/lockfree_hash.h File Reference

```
#include "fixed_types.h"
#include "basic_hash.h"
#include <map>
#include <utility>
#include <iostream>
#include <assert.h>
```

Classes

- class **LockFreeHash**

7.222 common/misc/log.cc File Reference

```
#include <unistd.h>
#include <sys/time.h>
#include <sys/syscall.h>
#include <stdarg.h>
#include <string.h>
#include <signal.h>
#include "log.h"
#include "config.h"
#include "simulator.h"
#include "core_manager.h"
#include "config.hpp"
#include "circular_log.h"
```

Functions

- static String **formatFileName** (const char *s)

7.222.1 Function Documentation

7.222.1.1 formatFileName()

```
static String formatFileName (
            const char * s )  [static]
```

Definition at line 23 of file log.cc.

Referenced by Log::getFile(), and Log::Log().

7.223 common/misc/log.d File Reference

7.224 common/misc/log.h File Reference

```
#include "fixed_types.h"
#include "lock.h"
#include <stdio.h>
#include <stdlib.h>
#include <set>
#include <map>
```

Classes

- class **Log**
- class **FunctionTracer**

Macros

- #define **likely**(x) __builtin_expect((x), 1)
- #define **unlikely**(x) __builtin_expect((x), 0)
- #define **__LOG_PRINT**(err, file, line, ...)
- #define **_LOG_PRINT**(err, ...)
- #define **LOG_PRINT**(...) **_LOG_PRINT**(**Log::None**, __VA_ARGS__); \
- #define **LOG_PRINT_WARNING**(...) **_LOG_PRINT**(**Log::Warning**, __VA_ARGS__);
- #define **LOG_PRINT_WARNING_ONCE**(...)
- #define **LOG_PRINT_ERROR**(...)
- #define **LOG_ASSERT_WARNING**(expr, ...)
- #define **LOG_ASSERT_WARNING_ONCE**(expr, ...)
- #define **LOG_ASSERT_ERROR**(expr, ...)
- #define **LOG_FUNC_TRACE**() **FunctionTracer** func_tracer(__FILE__,__LINE__,__PRETTY_FUNCTIO↩ N__);

7.224.1 Macro Definition Documentation

7.224.1.1 __LOG_PRINT

```
#define __LOG_PRINT(
              err,
              file,
              line,
              ...  )
```

Value:

```
   {                                                                    \
      if (unlikely(Log::getSingleton()->isLoggingEnabled()) || err != Log::None)  \
      {                                                                 \
         String module = Log::getSingleton()->getModule(file);     \
         if (err != Log::None ||                                        \
             Log::getSingleton()->isEnabled(module.c_str()))            \
         {                                                              \
            Log::getSingleton()->log(err, module.c_str(), line, __VA_ARGS__); \
         }                                                              \
      }                                                                 \
   }                                                                    \
```

Definition at line 96 of file log.h.

7.224.1.2 _LOG_PRINT

```
#define _LOG_PRINT(
              err,
              ...  )
```

Value:

```
   {                                                                    \
   __LOG_PRINT(err, __FILE__, __LINE__, __VA_ARGS__);                   \
   }                                                                    \
```

Definition at line 109 of file log.h.

7.224.1.3 likely

```
#define likely(
              x ) __builtin_expect((x), 1)
```

Definition at line 93 of file log.h.

7.224.1.4 LOG_ASSERT_ERROR

```
#define LOG_ASSERT_ERROR(
              expr,
              ...  )
```

Value:

```
   {                                                \
      if (!(expr))                                  \
      {                                             \
         LOG_PRINT_ERROR(__VA_ARGS__);              \
      }                                             \
   }                                                \
```

Definition at line 160 of file log.h.

7.224.1.5 LOG_ASSERT_WARNING

```
#define LOG_ASSERT_WARNING(
              expr,
              ...  )
```

Value:

```
   {                                                \
      if (!(expr))                                  \
      {                                             \
         LOG_PRINT_WARNING(__VA_ARGS__);            \
      }                                             \
   }                                                \
```

Definition at line 144 of file log.h.

7.224.1.6 LOG_ASSERT_WARNING_ONCE

```
#define LOG_ASSERT_WARNING_ONCE(
              expr,
              ... )
```

Value:

```
   {                                                \
      if (!(expr))                                  \
      {                                             \
         LOG_PRINT_WARNING_ONCE(__VA_ARGS__);       \
      }                                             \
   }                                                \
```

Definition at line 152 of file log.h.

7.224.1.7 LOG_FUNC_TRACE

```
#define LOG_FUNC_TRACE( )  FunctionTracer func_tracer(__FILE__,__LINE__,__PRETTY_FUNCTION__);
```

Definition at line 194 of file log.h.

7.224.1.8 LOG_PRINT

```
#define LOG_PRINT(
              ... )   _LOG_PRINT( Log::None, __VA_ARGS__); \
```

Definition at line 114 of file log.h.

7.224.1.9 LOG_PRINT_ERROR

```
#define LOG_PRINT_ERROR(
              ... )
```

Value:

```
   {                                          \
      _LOG_PRINT(Log::Error, __VA_ARGS__);    \
      exit(-1);                               \
   }
```

Definition at line 138 of file log.h.

7.224.1.10 LOG_PRINT_WARNING

```
#define LOG_PRINT_WARNING(
              ... )   _LOG_PRINT( Log::Warning, __VA_ARGS__);
```

Definition at line 117 of file log.h.

7.224.1.11 LOG_PRINT_WARNING_ONCE

```
#define LOG_PRINT_WARNING_ONCE(
               ...  )
```

Value:

```
{                                                \
   static bool already_printed = false;          \
   if (!already_printed)                         \
   {                                             \
      _LOG_PRINT(Log::Warning, __VA_ARGS__); \
      _LOG_PRINT(Log::Warning, "Future warnings of this type will be suppressed."); \
      already_printed = true;                    \
   }                                             \
}
```

Definition at line 120 of file log.h.

7.224.1.12 unlikely

```
#define unlikely(
              x ) __builtin_expect((x), 0)
```

Definition at line 94 of file log.h.

7.225 common/misc/logmem.cc File Reference

```
#include "logmem.h"
#include "simulator.h"
#include "lock.h"
#include "timer.h"
#include "hooks_manager.h"
#include <cstdio>
```

Functions

- void **logmem_enable** (bool enabled)
- void **logmem_write_allocations** ()

7.225.1 Function Documentation

7.225.1.1 logmem_enable()

```
void logmem_enable (
             bool enabled )
```

Definition at line 25 of file logmem.cc.

Referenced by MagicServer::setPerformance().

7.225.1.2 logmem_write_allocations()

```
void logmem_write_allocations ( )
```

Definition at line 26 of file logmem.cc.

Referenced by MagicServer::setPerformance().

7.226 common/misc/logmem.d File Reference

7.227 common/misc/logmem.h File Reference

Functions

- void **logmem_enable** (bool enabled)
- void **logmem_write_allocations** ()

7.227.1 Function Documentation

7.227.1.1 logmem_enable()

```
void logmem_enable (
            bool enabled )
```

Definition at line 25 of file logmem.cc.

Referenced by MagicServer::setPerformance().

7.227.1.2 logmem_write_allocations()

```
void logmem_write_allocations ( )
```

Definition at line 26 of file logmem.cc.

Referenced by MagicServer::setPerformance().

7.228 common/misc/memguard.cc File Reference

```
#include "memguard.h"
#include <sys/mman.h>
```

7.229 common/misc/memguard.d File Reference

7.230 common/misc/memguard.h File Reference

Classes

- class **MemGuard**

Macros

- #define **PAGE_SIZE** 4096

7.230.1 Macro Definition Documentation

7.230.1.1 PAGE_SIZE

```
#define PAGE_SIZE 4096
```

Definition at line 8 of file memguard.h.

7.231 common/misc/modulo_num.cc File Reference

```
#include <cassert>
#include "modulo_num.h"
```

7.232 common/misc/modulo_num.d File Reference

7.233 common/misc/modulo_num.h File Reference

```
#include "fixed_types.h"
```

Classes

- class **ModuloNum**

7.234 common/misc/moving_average.h File Reference

```
#include <vector>
#include <cmath>
#include "fixed_types.h"
#include "modulo_num.h"
#include "log.h"
```

Classes

- class **MovingAverage**< **T** >
- class **MovingArithmeticMean**< **T** >
- class **MovingGeometricMean**< **T** >
- class **MovingMedian**< **T** >

7.235 common/misc/mt_circular_queue.h File Reference

```
#include "circular_queue.h"
#include "lock.h"
#include "cond.h"
```

Classes

- class **MTCircularQueue**< **T** >
- class **MTCircularQueue**< **T** >**::iterator**

7.236 common/misc/os_compat.h File Reference

Macros

- #define **FUTEX_WAIT_BITSET** 9
- #define **FUTEX_WAKE_BITSET** 10
- #define **FUTEX_BITSET_MATCH_ANY** 0xffffffff
- #define **FUTEX_PRIVATE_FLAG** 0
- #define **FUTEX_CMD_MASK** 0x7f
- #define **CLOCK_MONOTONIC_RAW** 4
- #define **CLOCK_MONOTONIC_COARSE** 6
- #define **CPU_SET_S**(cpu, setsize, cpusetp) **__CPU_SET_S** (cpu, setsize, cpusetp)
- #define **CPU_CLR_S**(cpu, setsize, cpusetp) **__CPU_CLR_S** (cpu, setsize, cpusetp)
- #define **CPU_ISSET_S**(cpu, setsize, cpusetp)
- #define **CPU_ZERO_S**(setsize, cpusetp) **__CPU_ZERO_S** (setsize, cpusetp)
- #define **CPU_COUNT_S**(setsize, cpusetp) **__CPU_COUNT_S** (setsize, cpusetp)
- #define **__CPU_SET_S**(cpu, setsize, cpusetp)
- #define **__CPU_CLR_S**(cpu, setsize, cpusetp)
- #define **__CPU_ISSET_S**(cpu, setsize, cpusetp)
- #define **__CPU_ZERO_S**(setsize, cpusetp) do __builtin_memset (cpusetp, '\0', setsize); while (0)
- #define **__CPU_COUNT_S**(setsize, cpusetp) __sched_cpucount (setsize, cpusetp)

7.236.1 Macro Definition Documentation

7.236.1.1 __CPU_CLR_S

```
#define __CPU_CLR_S(
              cpu,
              setsize,
              cpusetp )
```

Value:

```
  (__extension__                                                              \
   ({ size_t __cpu = (cpu);                                                   \
      __cpu < 8 * (setsize)                                                   \
      ? (((__cpu_mask *) ((cpusetp)->__bits))[__CPUELT (__cpu)]               \
         &= ~__CPUMASK (__cpu))                                               \
      : 0; }))
```

Definition at line 46 of file os_compat.h.

7.236.1.2 __CPU_COUNT_S

```
#define __CPU_COUNT_S(
              setsize,
              cpusetp ) __sched_cpucount (setsize, cpusetp)
```

Definition at line 63 of file os_compat.h.

7.236.1.3 __CPU_ISSET_S

```
#define __CPU_ISSET_S(
              cpu,
              setsize,
              cpusetp )
```

Value:

```
  (__extension__                                                              \
   ({ size_t __cpu = (cpu);                                                   \
      __cpu < 8 * (setsize)                                                   \
      ? ((((__const __cpu_mask *) ((cpusetp)->__bits))[__CPUELT (__cpu)]      \
          & __CPUMASK (__cpu))) != 0                                          \
      : 0; }))
```

Definition at line 53 of file os_compat.h.

7.236.1.4 __CPU_SET_S

```
#define __CPU_SET_S(
              cpu,
              setsize,
              cpusetp )
```

Value:

```
  (__extension__                                                                \
   ({ size_t __cpu = (cpu);                                                     \
      __cpu < 8 * (setsize)                                                     \
      ? (((__cpu_mask *) ((cpusetp)->__bits))[__CPUELT (__cpu)]                 \
         |= __CPUMASK (__cpu))                                                  \
      : 0; }))
```

Definition at line 39 of file os_compat.h.

7.236.1.5 __CPU_ZERO_S

```
#define __CPU_ZERO_S(
              setsize,
              cpusetp ) do __builtin_memset (cpusetp, '\0', setsize); while (0)
```

Definition at line 61 of file os_compat.h.

7.236.1.6 CLOCK_MONOTONIC_COARSE

```
#define CLOCK_MONOTONIC_COARSE 6
```

Definition at line 28 of file os_compat.h.

7.236.1.7 CLOCK_MONOTONIC_RAW

```
#define CLOCK_MONOTONIC_RAW 4
```

Definition at line 24 of file os_compat.h.

7.236.1.8 CPU_CLR_S

```
#define CPU_CLR_S(
              cpu,
              setsize,
              cpusetp )  __CPU_CLR_S (cpu, setsize, cpusetp)
```

Definition at line 33 of file os_compat.h.

7.236.1.9 CPU_COUNT_S

```
#define CPU_COUNT_S(
              setsize,
              cpusetp )  __CPU_COUNT_S (setsize, cpusetp)
```

Definition at line 37 of file os_compat.h.

7.236.1.10 CPU_ISSET_S

```
#define CPU_ISSET_S(
              cpu,
              setsize,
              cpusetp )
```

Value:

```
__CPU_ISSET_S (cpu, setsize, \
cpusetp)
```

Definition at line 34 of file os_compat.h.

7.236.1.11 CPU_SET_S

```
#define CPU_SET_S(
              cpu,
              setsize,
              cpusetp )  __CPU_SET_S (cpu, setsize, cpusetp)
```

Definition at line 32 of file os_compat.h.

7.236.1.12 CPU_ZERO_S

```
#define CPU_ZERO_S(
              setsize,
              cpusetp )  __CPU_ZERO_S (setsize, cpusetp)
```

Definition at line 36 of file os_compat.h.

7.236.1.13 FUTEX_BITSET_MATCH_ANY

```
#define FUTEX_BITSET_MATCH_ANY 0xffffffff
```

Definition at line 10 of file os_compat.h.

7.236.1.14 FUTEX_CMD_MASK

```
#define FUTEX_CMD_MASK 0x7f
```

Definition at line 21 of file os_compat.h.

7.236.1.15 FUTEX_PRIVATE_FLAG

```
#define FUTEX_PRIVATE_FLAG 0
```

Definition at line 16 of file os_compat.h.

7.236.1.16 FUTEX_WAIT_BITSET

```
#define FUTEX_WAIT_BITSET 9
```

Definition at line 4 of file os_compat.h.

7.236.1.17 FUTEX_WAKE_BITSET

```
#define FUTEX_WAKE_BITSET 10
```

Definition at line 7 of file os_compat.h.

7.237 common/misc/packetize.cc File Reference

```
#include "packetize.h"
```

7.238 common/misc/packetize.d File Reference

7.239 common/misc/packetize.h File Reference

```
#include "log.h"
#include "fixed_types.h"
#include "subsecond_time.h"
#include <assert.h>
#include <iostream>
#include <utility>
```

Classes

- class **UnstructuredBuffer**

7.240 common/misc/progress.cc File Reference

```
#include "progress.h"
#include "simulator.h"
#include "config.hpp"
#include "core_manager.h"
#include "hooks_manager.h"
#include "trace_manager.h"
#include "magic_server.h"
```

7.241 common/misc/progress.d File Reference

7.242 common/misc/progress.h File Reference

```
#include "subsecond_time.h"
```

Classes

- class **Progress**

7.243 common/misc/pthread_lock.cc File Reference

```
#include "pthread_lock.h"
```

Functions

- **__attribute__** ((weak)) **LockImplementation** * **LockCreator_Default**

7.243.1 Function Documentation

7.243.1.1 __attribute__()

```
__attribute__ (
             (weak)  )
```

Definition at line 23 of file pthread_lock.cc.

7.244 common/misc/pthread_lock.d File Reference

7.245 common/misc/pthread_lock.h File Reference

```
#include "lock.h"
#include <pthread.h>
```

Classes

- class **PthreadLock**

7.246 common/misc/pthread_thread.cc File Reference

```
#include "pthread_thread.h"
#include "log.h"
```

Functions

- **__attribute__** ((weak)) **_Thread** * **_Thread**

7.246.1 Function Documentation

7.246.1.1 __attribute__()

```
__attribute__ (
            (weak)  )
```

Definition at line 34 of file pthread_thread.cc.

7.247 common/misc/pthread_thread.d File Reference

7.248 common/misc/pthread_thread.h File Reference

```
#include "_thread.h"
#include <pthread.h>
```

Classes

- class **PthreadThread**
- struct **PthreadThread::FuncData**

7.249 common/misc/pthread_tls.cc File Reference

```
#include "tls.h"
#include <pthread.h>
```

Classes

- class **PthreadTLS**

Functions

- **__attribute__** ((weak)) **TLS** ∗ **TLS**

7.249.1 Function Documentation

7.249.1.1 __attribute__()

```
__attribute__ (
             (weak)  )
```

Definition at line 38 of file pthread_tls.cc.

7.250 common/misc/pthread_tls.d File Reference

7.251 common/misc/random.h File Reference

Classes

- class **Random**

7.252 common/misc/rng.h File Reference

Macros

- #define **RNG_A** __UINT64_C(0x5DEECE66D)
- #define **RNG_C** __UINT64_C(0xB)
- #define **RNG_M** ((__UINT64_C(1) << 48) - 1)

Functions

- **UInt64 rng_next** (**UInt64** &state)
- **UInt64 rng_seed** (**UInt64** seed)

7.252.1 Macro Definition Documentation

7.252.1.1 RNG_A

```
#define RNG_A __UINT64_C(0x5DEECE66D)
```

Definition at line 5 of file rng.h.

7.252.1.2 RNG_C

```
#define RNG_C __UINT64_C(0xB)
```

Definition at line 6 of file rng.h.

7.252.1.3 RNG_M

```
#define RNG_M ((__UINT64_C(1) << 48) - 1)
```

Definition at line 7 of file rng.h.

7.252.2 Function Documentation

7.252.2.1 rng_next()

```
UInt64 rng_next (
            UInt64 & state )  [inline]
```

Definition at line 10 of file rng.h.

References RNG_A, RNG_C, and RNG_M.

Referenced by ATD::ATD(), BbvCount::count(), SchedulerBigSmall::pickBigThread(), FaultInjectorRandom::preRead(), BbvCount::sampleReset(), CacheBase::splitAddress(), and TraceThread::TraceThread().

7.252.2.2 rng_seed()

```
UInt64 rng_seed (
            UInt64 seed )  [inline]
```

Definition at line 15 of file rng.h.

Referenced by ATD::ATD(), BbvCount::count(), CacheBase::splitAddress(), and TraceThread::TraceThread().

7.253 common/misc/selock.cc File Reference

```
#include "selock.h"
#include <assert.h>
```

Macros

- #define **WAIT_WHILE**(condition)

7.253.1 Macro Definition Documentation

7.253.1.1 WAIT_WHILE

```
#define WAIT_WHILE(
              condition )
```

Value:

```
   /* First busy wait a little */                  \
   for(int i = 0; i < 10000 && (condition); ++i) ; \
   while(condition) {                              \
      /* Then reschedule */                        \
      sched_yield();                               \
   }
```

Definition at line 14 of file selock.cc.

7.254 common/misc/selock.d File Reference

7.255 common/misc/selock.h File Reference

```
#include "lock.h"
```

Classes

- class **SELock**

7.256 common/misc/semaphore.cc File Reference

```
#include "semaphore.h"
#include "os_compat.h"
#include <unistd.h>
#include <sys/syscall.h>
#include <linux/futex.h>
#include <limits.h>
```

7.257 common/misc/semaphore.d File Reference

7.258 common/misc/semaphore.h File Reference

```
#include "lock.h"
```

Classes

- class **Semaphore**

7.259 common/misc/setlock.cc File Reference

```
#include "setlock.h"
#include <assert.h>
```

7.260 common/misc/setlock.d File Reference

7.261 common/misc/setlock.h File Reference

```
#include "lock.h"
#include "selock.h"
#include <vector>
#include <pthread.h>
```

Classes

- class **_SetLock**
- class **_SetLock::PersetLock**
- class **_SELock**

Typedefs

- typedef **_SetLock SetLock**

Functions

- **PersetLock** ()
- void **acquire** ()
- void **release** ()

Variables

- pthread_mutex_t **_mutx**

7.261.1 Typedef Documentation

7.261.1.1 SetLock

```
typedef _SetLock SetLock
```

Definition at line 54 of file setlock.h.

7.261.2 Function Documentation

7.261.2.1 acquire()

```
void __attribute__::acquire ( )
```

Definition at line 24 of file setlock.h.

7.261.2.2 PersetLock()

```
__attribute__::PersetLock ( )
```

Definition at line 23 of file setlock.h.

7.261.2.3 release()

```
void __attribute__::release ( )
```

Definition at line 25 of file setlock.h.

7.261.3 Variable Documentation

7.261.3.1 _mutx

```
pthread_mutex_t _mutx  [private]
```

Definition at line 27 of file setlock.h.

7.262 common/misc/spin_loop_detector.cc File Reference

```
#include "spin_loop_detector.h"
#include "core.h"
#include "performance_model.h"
```

7.263 common/misc/spin_loop_detector.d File Reference

7.264 common/misc/spin_loop_detector.h File Reference

```
#include "fixed_types.h"
#include "thread.h"
#include <unordered_map>
#include <deque>
```

Classes

- class **SpinLoopDetector**
- struct **SpinLoopDetector::SdtEntry**

Variables

- class **SpinLoopDetector __attribute__**

7.264.1 Variable Documentation

7.264.1.1 __attribute__

```
__attribute__
```

Definition at line 28 of file pthread_lock.cc.

Referenced by Fxsupport::Fxsupport(), PerformanceModel::handleIdleInstruction(), PthreadEmu::init(), Cache↩ ::insertSingleLine(), ParametricDramDirectoryMSI::CacheCntlr::invalidateCacheBlock(), UnstructuredBuffer↩ ::operator>>(), printInsInfo(), ParametricDramDirectoryMSI::CacheCntlr::retrieveCacheBlock(), PinTLS::set(), SimThreadManager::spawnSimThreads(), SyscallMdl::SyscallMdl(), ParametricDramDirectoryMSI::CacheCntlr↩ ::updateCacheBlock(), and ParametricDramDirectoryMSI::CacheCntlr::writeCacheBlock().

7.265 common/misc/spinlock.h File Reference

Classes

- struct **raw_spinlock_t**

Macros

- #define **__RAW_SPIN_LOCK_UNLOCKED**
- #define **__raw_spin_is_locked**(x) (∗(volatile signed char ∗)(&(x)->slock) <= 0)
- #define **__raw_spin_lock_string**
- #define **__raw_spin_lock_string_flags**
- #define **__raw_spin_unlock_string**
- #define **__raw_spin_unlock_wait**(lock) do { while (**__raw_spin_is_locked**(lock)) cpu_relax(); } while (0)

Functions

- static void **__raw_spin_lock** (**raw_spinlock_t** ∗lock)
- static void **__raw_spin_lock_flags** (**raw_spinlock_t** ∗lock, unsigned long flags)
- static int **__raw_spin_trylock** (**raw_spinlock_t** ∗lock)
- static void **__raw_spin_unlock** (**raw_spinlock_t** ∗lock)

7.265.1 Macro Definition Documentation

7.265.1.1 __raw_spin_is_locked

```
#define __raw_spin_is_locked(
              x ) (*(volatile signed char *)(&(x)->slock) <= 0)
```

Definition at line 10 of file spinlock.h.

7.265.1.2 __raw_spin_lock_string

```
#define __raw_spin_lock_string
```

Value:

```
"\n1:\t" \
"lock ; decb %0\n\t" \
"jns 3f\n" \
"2:\t" \
"rep;nop\n\t" \
"cmpb $0,%0\n\t" \
"jle 2b\n\t" \
"jmp 1b\n" \
"3:\n\t"
```

Definition at line 13 of file spinlock.h.

7.265.1.3 __raw_spin_lock_string_flags

```
#define __raw_spin_lock_string_flags
```

Value:

```
"\n1:\t" \
"lock ; decb %0\n\t" \
"jns 5f\n" \
"2:\t" \
"testl $0x200, %1\n\t" \
"jz 4f\n\t" \
"sti\n" \
"3:\t" \
"rep;nop\n\t" \
"cmpb $0, %0\n\t" \
"jle 3b\n\t" \
"cli\n\t" \
"jmp 1b\n" \
"4:\t" \
"rep;nop\n\t" \
"cmpb $0, %0\n\t" \
"jg 1b\n\t" \
"jmp 4b\n" \
"5:\n\t"
```

Definition at line 29 of file spinlock.h.

7.265.1.4 __RAW_SPIN_LOCK_UNLOCKED

```
#define __RAW_SPIN_LOCK_UNLOCKED
```

Value:

```
{ 1 }/*
* Your basic SMP spinlocks, allowing only a single CPU anywhere
*
* Simple spin lock operations.  There are two variants, one clears IRQ's
* on the local processor, one does not.
*
* We make no fairness assumptions. They have a cost.
*
* (the type definitions are in asm/spinlock_types.h)
*/
```

Definition at line 8 of file spinlock.h.

7.265.1.5 __raw_spin_unlock_string

```
#define __raw_spin_unlock_string
```

Value:

```
        "movb $1,%0" \
                :"+m" (lock->slock) : : "memory"
```

Definition at line 86 of file spinlock.h.

7.265.1.6 __raw_spin_unlock_wait

```
#define __raw_spin_unlock_wait(
                lock ) do { while ( __raw_spin_is_locked(lock)) cpu_relax(); } while (0)
```

Definition at line 116 of file spinlock.h.

7.265.2 Function Documentation

7.265.2.1 __raw_spin_lock()

```
static void __raw_spin_lock (
                raw_spinlock_t * lock )  [inline], [static]
```

Definition at line 50 of file spinlock.h.

7.265.2.2 __raw_spin_lock_flags()

```
static void __raw_spin_lock_flags (
                raw_spinlock_t * lock,
               unsigned long flags )  [inline], [static]
```

Definition at line 61 of file spinlock.h.

7.265.2.3 __raw_spin_trylock()

```
static int __raw_spin_trylock (
                raw_spinlock_t * lock )  [inline], [static]
```

Definition at line 67 of file spinlock.h.

References __raw_spin_lock_string_flags, and raw_spinlock_t::slock.

7.265.2.4 __raw_spin_unlock()

```
static void __raw_spin_unlock (
              raw_spinlock_t * lock )  [inline], [static]
```

Definition at line 91 of file spinlock.h.

References __raw_spin_unlock_string.

7.266 common/misc/stable_iterator.h File Reference

```
#include <vector>
```

Classes

- class **StableIterator**< **T** >

7.267 common/misc/stats.cc File Reference

```
#include "stats.h"
#include "simulator.h"
#include "hooks_manager.h"
#include "utils.h"
#include "itostr.h"
#include <math.h>
#include <stdio.h>
#include <sstream>
#include <unordered_set>
#include <string>
#include <cstring>
#include <zlib.h>
#include <sys/time.h>
```

Functions

- template<> **UInt64** **makeStatsValue**< **UInt64** > (**UInt64** t)
- template<> **UInt64** **makeStatsValue**< **SubsecondTime** > (**SubsecondTime** t)
- template<> **UInt64** **makeStatsValue**< **ComponentTime** > (**ComponentTime** t)
- **UInt64** **getWallclockTimeCallback** (String objectName, **UInt32** index, String metricName, **UInt64** arg)

Variables

- const char * **db_create_stmts** [ ]
- const char **db_insert_stmt_name** [ ] = "INSERT INTO `names` (nameid, objectname, metricname) VALUES (?, ?, ?);"
- const char **db_insert_stmt_prefix** [ ] = "INSERT INTO `prefixes` (prefixid, prefixname) VALUES (?, ?);"
- const char **db_insert_stmt_value** [ ] = "INSERT INTO `values` (prefixid, nameid, core, value) VALUES (?, ?, ?, ?);"

7.267.1 Function Documentation

7.267.1.1 getWallclockTimeCallback()

```
UInt64 getWallclockTimeCallback (
            String objectName,
             UInt32 index,
            String metricName,
             UInt64 arg )
```

Definition at line 35 of file stats.cc.

Referenced by StatsManager::StatsManager().

7.267.1.2 makeStatsValue< ComponentTime >()

```
template<>
UInt64 makeStatsValue< ComponentTime > (
             ComponentTime t )
```

Definition at line 18 of file stats.cc.

References ComponentTime::getElapsedTime(), and SubsecondTime::getFS().

7.267.1.3 makeStatsValue< SubsecondTime >()

```
template<>
UInt64 makeStatsValue< SubsecondTime > (
             SubsecondTime t )
```

Definition at line 17 of file stats.cc.

References SubsecondTime::getFS().

7.267.1.4 makeStatsValue< UInt64 >()

```
template<>
UInt64 makeStatsValue< UInt64 > (
             UInt64 t )
```

Definition at line 16 of file stats.cc.

7.267.2 Variable Documentation

7.267.2.1 db_create_stmts

```
const char* db_create_stmts[]
```

Initial value:

```
= {

   "CREATE TABLE `names` (nameid INTEGER, objectname TEXT, metricname TEXT);",
   "CREATE TABLE `prefixes` (prefixid INTEGER, prefixname TEXT);",
   "CREATE TABLE `values` (prefixid INTEGER, nameid INTEGER, core INTEGER, value INTEGER);",
   "CREATE INDEX `idx_prefix_name` ON `prefixes`(`prefixname`);",
   "CREATE INDEX `idx_value_prefix` ON `values`(`prefixid`);",

   "CREATE TABLE `topology` (componentname TEXT, coreid INTEGER, masterid INTEGER);",
   "CREATE TABLE `event` (event INTEGER, time INTEGER, core INTEGER, thread INTEGER, value0 INTEGER, value1
       INTEGER, description TEXT);",
}
```

Definition at line 20 of file stats.cc.

Referenced by StatsManager::init().

7.267.2.2 db_insert_stmt_name

```
const char db_insert_stmt_name[] = "INSERT INTO `names` (nameid, objectname, metricname) VAL↩
UES (?, ?, ?);"
```

Definition at line 31 of file stats.cc.

Referenced by StatsManager::init().

7.267.2.3 db_insert_stmt_prefix

```
const char db_insert_stmt_prefix[] = "INSERT INTO `prefixes` (prefixid, prefixname) VALUES (?,
?);"
```

Definition at line 32 of file stats.cc.

Referenced by StatsManager::init().

7.267.2.4 db_insert_stmt_value

```
const char db_insert_stmt_value[] = "INSERT INTO `values` (prefixid, nameid, core, value) VA↩
LUES (?, ?, ?, ?);"
```

Definition at line 33 of file stats.cc.

Referenced by StatsManager::init().

7.268 common/misc/stats.d File Reference

7.269 common/misc/stats.h File Reference

```
#include "simulator.h"
#include "itostr.h"
#include <cstring>
#include <sqlite3.h>
```

Classes

- class **StatsMetricBase**
- class **StatsMetric**< **T** >
- class **StatsMetricCallback**
- class **StatsManager**
- class **StatHist**

Typedefs

- typedef **UInt64**(* **StatsCallback**) (String objectName, **UInt32** index, String metricName, **UInt64** arg)

Functions

- template<class T >
  UInt64 **makeStatsValue** (T t)
- template<class T >
  void **registerStatsMetric** (String objectName, **UInt32** index, String metricName, T *metric)

7.269.1 Typedef Documentation

7.269.1.1 StatsCallback

```
typedef UInt64(* StatsCallback) (String objectName, UInt32 index, String metricName, UInt64
arg)
```

Definition at line 42 of file stats.h.

7.269.2 Function Documentation

7.269.2.1 makeStatsValue()

```
template<class T >
UInt64 makeStatsValue (
            T t )
```

7.269.2.2 registerStatsMetric()

```
template<class T >
void registerStatsMetric (
            String objectName,
             UInt32 index,
            String metricName,
            T * metric )
```

Definition at line 104 of file stats.h.

Referenced by ATD::ATD(), BarrierSyncServer::BarrierSyncServer(), BbvCount::BbvCount(), BranchPredictor↩ ::BranchPredictor(), FastNehalem::Cache< assoc, size_kb >::Cache(), ParametricDramDirectoryMSI::Cache↩ Cntlr::CacheCntlr(), CacheSetInfoLRU::CacheSetInfoLRU(), CheetahManager::CheetahStats::CheetahStats(), ContentionModel::ContentionModel(), Core::Core(), Directory::Directory(), FastNehalem::Dram::Dram(), Dram↩ Cache::DramCache(), PrL1PrL2DramDirectoryMSI::DramCntlr::DramCntlr(), PrL1PrL2DramDirectoryMSI::↩ DramDirectoryCntlr::DramDirectoryCntlr(), DramPerfModelConstant::DramPerfModelConstant(), DramPerf↩ ModelNormal::DramPerfModelNormal(), DramPerfModelReadWrite::DramPerfModelReadWrite(), Fastforward↩ PerformanceModel::FastforwardPerformanceModel(), PthreadEmu::init(), RobSmtTimer::initializeThread(), InstructionTracerFPStats::InstructionTracerFPStats(), IntervalContentionBoomV1::IntervalContentionBoomV1(), IntervalContentionNehalem::IntervalContentionNehalem(), IntervalTimer::IntervalTimer(), MicroOpPerformance↩ Model::MicroOpPerformanceModel(), NetworkModel::NetworkModel(), NetworkModelBusGlobal::NetworkModel↩ BusGlobal(), NetworkModelEMeshHopByHop::NetworkModelEMeshHopByHop(), NucaCache::NucaCache(), OneIPCPerformanceModel::OneIPCPerformanceModel(), PerformanceModel::PerformanceModel(), Queue↩ ModelHistoryList::QueueModelHistoryList(), QueueModelWindowedMG1::QueueModelWindowedMG1(), Rob↩ SmtTimer::RobSmtTimer(), RobTimer::RobTimer(), SyscallMdl::SyscallMdl(), BottleGraphManager::threadStart(), ThreadStatsManager::ThreadStats::ThreadStats(), and ParametricDramDirectoryMSI::TLB::TLB().

7.270 common/misc/subsecond_time.cc File Reference

```
#include "core.h"
#include "simulator.h"
#include "core_manager.h"
#include "subsecond_time.h"
#include <iostream>
```

Functions

- std::ostream & **operator**<< (std::ostream &os, const **SubsecondTime** &time)

7.270.1 Function Documentation

7.270.1.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const  SubsecondTime & time )
```

Definition at line 8 of file subsecond_time.cc.

References SubsecondTime::m_time.

7.271 common/misc/subsecond_time.d File Reference

7.272 common/misc/subsecond_time.h File Reference

```
#include "fixed_types.h"
#include "lock.h"
#include "subsecond_time_c.h"
#include <cassert>
#include <iostream>
```

Classes

- struct **TimeConverter< T >**
- class **SubsecondTime**
- class **ComponentPeriod**
- class **SubsecondTimeCycleConverter**
- class **ComponentBandwidth**
- class **ComponentBandwidthPerCycle**
- class **ComponentLatency**
- class **ComponentTime**

Macros

- #define **SUBSECOND_TIME_SIMPLE_OSTREAM**

Functions

- template<class T >
  SubsecondTime operator* (T lhs, const **SubsecondTime** &rhs)
- **SubsecondTime operator+** (**SubsecondTime** lhs, const **SubsecondTime** &rhs)
- **SubsecondTime operator-** (**SubsecondTime** lhs, const **SubsecondTime** &rhs)
- **SubsecondTime operator<<** (**SubsecondTime** lhs, uint64_t rhs)
- template<class T >
  SubsecondTime operator* (**SubsecondTime** lhs, T rhs)
- bool **operator==** (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)
- bool **operator!=** (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)
- bool **operator<** (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)
- bool **operator<=** (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)

- bool **operator>=** (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)
- bool **operator>** (const **SubsecondTime** &lhs, const **SubsecondTime** &rhs)
- void **atomic_add_subsecondtime** (**SubsecondTime** &src_dest, const **SubsecondTime** & **src**)
- **ComponentPeriod operator*** (**ComponentPeriod** lhs, uint64_t rhs)
- **ComponentPeriod operator*** (uint64_t lhs, const **ComponentPeriod** &rhs)
- std::ostream & **operator<<** (std::ostream &os, const **ComponentPeriod** &period)
- std::ostream & **operator<<** (std::ostream &os, const **ComponentBandwidth** &bandwidth)
- std::ostream & **operator<<** (std::ostream &os, const **ComponentBandwidthPerCycle** &bandwidth)
- std::ostream & **operator<<** (std::ostream &os, const **ComponentLatency** &latency)
- std::ostream & **operator<<** (std::ostream &os, const **ComponentTime** &time)

7.272.1 Macro Definition Documentation

7.272.1.1 SUBSECOND_TIME_SIMPLE_OSTREAM

```
#define SUBSECOND_TIME_SIMPLE_OSTREAM
```

Definition at line 13 of file subsecond_time.h.

7.272.2 Function Documentation

7.272.2.1 atomic_add_subsecondtime()

```
void atomic_add_subsecondtime (
              SubsecondTime & src_dest,
             const  SubsecondTime & src )  [inline]
```

Definition at line 282 of file subsecond_time.h.

References SubsecondTime::m_time, and src.

Referenced by ShmemPerfModel::incrElapsedTime(), ParametricDramDirectoryMSI::CacheCntlr::incrementQB↩SLookupCost(), ShmemPerfModel::incrTotalMemoryAccessLatency(), ParametricDramDirectoryMSI::CacheCntlr↩::insertCacheBlock(), ParametricDramDirectoryMSI::CacheCntlr::Prefetch(), and ParametricDramDirectoryMSI::↩CacheCntlr::processShmemReqFromPrevCache().

7.272.2.2 operator"!=()

```
bool operator!= (
             const  SubsecondTime & lhs,
             const  SubsecondTime & rhs )  [inline]
```

Definition at line 261 of file subsecond_time.h.

References operator==().

7.272.2.3 operator*() [1/4]

```
ComponentPeriod operator* (
            ComponentPeriod lhs,
            uint64_t rhs )  [inline]
```

Definition at line 346 of file subsecond_time.h.

7.272.2.4 operator*() [2/4]

```
template<class T >
SubsecondTime operator* (
            SubsecondTime lhs,
            T rhs )  [inline]
```

Definition at line 244 of file subsecond_time.h.

7.272.2.5 operator*() [3/4]

```
template<class T >
SubsecondTime operator* (
            T lhs,
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 250 of file subsecond_time.h.

7.272.2.6 operator*() [4/4]

```
ComponentPeriod operator* (
            uint64_t lhs,
            const  ComponentPeriod & rhs )  [inline]
```

Definition at line 350 of file subsecond_time.h.

7.272.2.7 operator+()

```
SubsecondTime operator+ (
            SubsecondTime lhs,
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 222 of file subsecond_time.h.

7.272.2.8 operator-()

```
SubsecondTime operator- (
            SubsecondTime lhs,
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 229 of file subsecond_time.h.

7.272.2.9 operator<()

```
bool operator< (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 265 of file subsecond_time.h.

References SubsecondTime::m_time.

Referenced by operator>(), and operator>=().

7.272.2.10 operator<<() [1/6]

```
std::ostream& operator<< (
            std::ostream & os,
            const  ComponentBandwidth & bandwidth )  [inline]
```

Definition at line 415 of file subsecond_time.h.

References ComponentBandwidth::m_bw_in_bits_per_us.

7.272.2.11 operator<<() [2/6]

```
std::ostream& operator<< (
            std::ostream & os,
            const  ComponentBandwidthPerCycle & bandwidth )  [inline]
```

Definition at line 467 of file subsecond_time.h.

References ComponentBandwidthPerCycle::m_bw_in_bits_per_cycle, and ComponentBandwidthPerCycle::m_↩ period.

7.272.2.12 operator<<() [3/6]

```
std::ostream& operator<< (
            std::ostream & os,
            const  ComponentLatency & latency )  [inline]
```

Definition at line 513 of file subsecond_time.h.

References ComponentLatency::m_fixed_cycle_latency, and ComponentLatency::m_period.

7.272.2.13 operator<<() [4/6]

```
std::ostream& operator<< (
            std::ostream & os,
            const  ComponentPeriod & period )  [inline]
```

Definition at line 355 of file subsecond_time.h.

References ComponentPeriod::m_period.

7.272.2.14 operator<<() [5/6]

```
std::ostream& operator<< (
            std::ostream & os,
            const  ComponentTime & time )  [inline]
```

Definition at line 631 of file subsecond_time.h.

References ComponentPeriod::getPeriod(), ComponentTime::m_period, and ComponentTime::m_time.

7.272.2.15 operator<<() [6/6]

```
SubsecondTime operator<< (
             SubsecondTime lhs,
            uint64_t rhs )  [inline]
```

Definition at line 236 of file subsecond_time.h.

7.272.2.16 operator<=()

```
bool operator<= (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 269 of file subsecond_time.h.

References operator>().

7.272.2.17 operator==()

```
bool operator== (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 257 of file subsecond_time.h.

References SubsecondTime::m_time.

Referenced by operator!=().

7.272.2.18 operator>()

```
bool operator> (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 277 of file subsecond_time.h.

References operator<().

Referenced by operator<=().

7.272.2.19 operator>=()

```
bool operator>= (
            const  SubsecondTime & lhs,
            const  SubsecondTime & rhs )  [inline]
```

Definition at line 273 of file subsecond_time.h.

References operator<().

7.273 common/misc/subsecond_time_c.cc File Reference

```
#include "subsecond_time.h"
```

Functions

- std::ostream & **operator**<< (std::ostream &os, const **subsecond_time_t** &time)

7.273.1 Function Documentation

7.273.1.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & os,
            const  subsecond_time_t & time )
```

Definition at line 4 of file subsecond_time_c.cc.

7.274 common/misc/subsecond_time_c.d File Reference

7.275 common/misc/subsecond_time_c.h File Reference

```
#include <stdint.h>
```

Classes

- struct **subsecond_time_s**

Typedefs

- typedef struct **subsecond_time_s** **subsecond_time_t**

7.275.1 Typedef Documentation

7.275.1.1 subsecond_time_t

```
typedef struct  subsecond_time_s  subsecond_time_t
```

Definition at line 30 of file subsecond_time_c.h.

7.276 common/misc/syscall_strings.cc File Reference

```
#include "syscall_strings.h"
```

Functions

- const char * **syscall_string** (int syscall_number)

7.276.1 Function Documentation

7.276.1.1 syscall_string()

```
const char* syscall_string (
            int syscall_number )
```

Definition at line 5 of file syscall_strings.cc.

Referenced by SyscallMdl::formatSyscall().

7.277 common/misc/syscall_strings.d File Reference

7.278 common/misc/syscall_strings.h File Reference

Functions

- const char * **syscall_string** (int syscall_number)

7.278.1 Function Documentation

7.278.1.1 syscall_string()

```
const char* syscall_string (
            int syscall_number )
```

Definition at line 5 of file syscall_strings.cc.

Referenced by SyscallMdl::formatSyscall().

7.279 common/misc/tags.cc File Reference

```
#include "tags.h"
#include "simulator.h"
#include "config.hpp"
#include <iostream>
```

7.280 common/misc/tags.d File Reference

7.281 common/misc/tags.h File Reference

```
#include "fixed_types.h"
#include <string>
#include <map>
```

Classes

- struct **Tag**
- class **TagsManager**

Namespaces

- **config**

7.282 common/misc/timer.cc File Reference

```
#include "timer.h"
#include "simulator.h"
#include <execinfo.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

Functions

- **UInt64 rdtsc** (void)
- **UInt64 rdtsc_and_cpuid** (**UInt32** *id)
- static **UInt64 getHashByStacktrace** (void)

Variables

- static const int **MAX_TIMERS** = 1024
- static **TotalTimer** * **alltimers** [**MAX_TIMERS**]
- static int **numtimers** = 0
- static std::map< **UInt64**, **TotalTimer** * > * **totaltimershash** = NULL

7.282.1 Function Documentation

7.282.1.1 getHashByStacktrace()

```
static UInt64 getHashByStacktrace (
            void ) [static]
```

Definition at line 136 of file timer.cc.

References n.

Referenced by TotalTimer::getTimerByStacktrace().

7.282.1.2 rdtsc()

```
UInt64 rdtsc (
            void )
```

Definition at line 16 of file timer.cc.

Referenced by exceptionHandler(), Timer::getTime(), CircularLog::getTime(), TotalTimer::reports(), and Timer::start().

7.282.1.3 rdtsc_and_cpuid()

```
UInt64 rdtsc_and_cpuid (
            UInt32 * id ) [inline]
```

Definition at line 27 of file timer.cc.

Referenced by Timer::getTime(), Timer::start(), and Timer::Timer().

7.282.2 Variable Documentation

7.282.2.1 alltimers

```
TotalTimer* alltimers[ MAX_TIMERS]  [static]
```

Definition at line 98 of file timer.cc.

Referenced by TotalTimer::reports(), and TotalTimer::TotalTimer().

7.282.2.2 MAX_TIMERS

```
const int MAX_TIMERS = 1024  [static]
```

Definition at line 97 of file timer.cc.

Referenced by TotalTimer::TotalTimer().

7.282.2.3 numtimers

```
int numtimers = 0  [static]
```

Definition at line 99 of file timer.cc.

Referenced by TotalTimer::reports(), and TotalTimer::TotalTimer().

7.282.2.4 totaltimershash

```
std::map< UInt64,  TotalTimer*>* totaltimershash = NULL  [static]
```

Definition at line 148 of file timer.cc.

Referenced by TotalTimer::getTimerByStacktrace().

7.283 common/misc/timer.d File Reference

7.284 common/misc/timer.h File Reference

```
#include "fixed_point.h"
#include <sys/time.h>
#include <vector>
```

Classes

- class **Timer**
- class **TotalTimer**
- class **ScopedTimer**

Functions

- **UInt64 rdtsc** (void)

7.284.1 Function Documentation

7.284.1.1 rdtsc()

```
UInt64 rdtsc (
            void  )
```

Definition at line 16 of file timer.cc.

Referenced by exceptionHandler(), Timer::getTime(), CircularLog::getTime(), TotalTimer::reports(), and Timer↩ ::start().

7.285 common/misc/tls.cc File Reference

```
#include "tls.h"
```

7.286 common/misc/tls.d File Reference

7.287 common/misc/tls.h File Reference

```
#include "fixed_types.h"
```

Classes

- class **TLS**

7.288 common/misc/utils.cc File Reference

```
#include "utils.h"
```

Functions

- String **myDecStr** (**UInt64** v, **UInt32** w)
- bool **isPower2** (**UInt32 n**)
- **SInt32 floorLog2** (**UInt32 n**)
- **SInt32 ceilLog2** (**UInt32 n**)
- **UInt64 countBits** (**UInt64 n**)

7.288.1 Function Documentation

7.288.1.1 ceilLog2()

```
SInt32 ceilLog2 (
            UInt32 n )
```

Definition at line 38 of file utils.cc.

References floorLog2(), and n.

Referenced by Config::computeCoreIDLength().

7.288.1.2 countBits()

```
UInt64 countBits (
            UInt64 n )
```

Definition at line 44 of file utils.cc.

References n.

Referenced by RoutineTracerFunctionStats::RtnMaster::ce_notify_evict().

7.288.1.3 floorLog2()

```
SInt32 floorLog2 (
            UInt32 n )
```

Definition at line 22 of file utils.cc.

References n.

Referenced by FastNehalem::Cache< assoc, size_kb >::Cache(), CacheBase::CacheBase(), ceilLog2(), ParametricDramDirectoryMSI::CacheMasterCntlr::createSetLocks(), PrL1PrL2DramDirectoryMSI::DramDirectory↩Cache::DramDirectoryCache(), ParametricDramDirectoryMSI::MemoryManager::MemoryManager(), and Stat↩Hist::update().

7.288.1.4 isPower2()

```
bool isPower2 (
              UInt32 n )
```

Definition at line 18 of file utils.cc.

References n.

Referenced by IntervalTimer::IntervalTimer().

7.288.1.5 myDecStr()

```
String myDecStr (
              UInt64 v,
              UInt32 w )
```

Definition at line 8 of file utils.cc.

7.289 common/misc/utils.d File Reference

7.290 common/misc/utils.h File Reference

```
#include "fixed_types.h"
#include <assert.h>
#include <sstream>
#include <iostream>
```

Macros

- #define **safeFDiv**(x) (x ? (double) x : 1.0)

Functions

- String **myDecStr** (**UInt64** v, **UInt32** w)
- bool **isPower2** (**UInt32 n**)
- **SInt32 floorLog2** (**UInt32 n**)
- **SInt32 ceilLog2** (**UInt32 n**)
- template<class T >
  T **getMin** (T v1, T v2)
- template<class T >
  T **getMax** (T v1, T v2)
- **UInt64 countBits** (**UInt64 n**)

7.290.1 Macro Definition Documentation

7.290.1.1 safeFDiv

```
#define safeFDiv(
             x ) (x ?  (double) x :  1.0)
```

Definition at line 13 of file utils.h.

7.290.2 Function Documentation

7.290.2.1 ceilLog2()

```
SInt32 ceilLog2 (
             UInt32 n )
```

Definition at line 38 of file utils.cc.

References floorLog2(), and n.

Referenced by Config::computeCoreIDLength().

7.290.2.2 countBits()

```
UInt64 countBits (
             UInt64 n )
```

Definition at line 44 of file utils.cc.

References n.

Referenced by RoutineTracerFunctionStats::RtnMaster::ce_notify_evict().

7.290.2.3 floorLog2()

```
SInt32 floorLog2 (
             UInt32 n )
```

Definition at line 22 of file utils.cc.

References n.

Referenced by FastNehalem::Cache< assoc, size_kb >::Cache(), CacheBase::CacheBase(), ceilLog2(), ParametricDramDirectoryMSI::CacheMasterCntlr::createSetLocks(), PrL1PrL2DramDirectoryMSI::DramDirectory↩Cache::DramDirectoryCache(), ParametricDramDirectoryMSI::MemoryManager::MemoryManager(), and Stat↩Hist::update().

7.290.2.4 getMax()

```
template<class T >
T getMax (
            T v1,
            T v2 )
```

Definition at line 43 of file utils.h.

7.290.2.5 getMin()

```
template<class T >
T getMin (
            T v1,
            T v2 )
```

Definition at line 37 of file utils.h.

7.290.2.6 isPower2()

```
bool isPower2 (
            UInt32 n )
```

Definition at line 18 of file utils.cc.

References n.

Referenced by IntervalTimer::IntervalTimer().

7.290.2.7 myDecStr()

```
String myDecStr (
            UInt64 v,
            UInt32 w )
```

Definition at line 8 of file utils.cc.

7.291 common/network/network.cc File Reference

```
#include <string.h>
#include "transport.h"
#include "core.h"
#include "network.h"
#include "memory_manager_base.h"
#include "simulator.h"
#include "core_manager.h"
#include "log.h"
#include "subsecond_time.h"
#include "performance_model.h"
#include "instruction.h"
```

Classes

- class **NetRecvIterator**

7.292 common/network/network.d File Reference

7.293 common/network/network.h File Reference

```
#include "packet_type.h"
#include "fixed_types.h"
#include "cond.h"
#include "transport.h"
#include "network_model.h"
#include "subsecond_time.h"
#include <iostream>
#include <vector>
#include <list>
```

Classes

- class **NetPacket**
- class **NetMatch**
- class **Network**

Typedefs

- typedef std::list< **NetPacket** > **NetQueue**

7.293.1 Typedef Documentation

7.293.1.1 NetQueue

```
typedef std::list< NetPacket>  NetQueue
```

Definition at line 46 of file network.h.

7.294 common/network/network_model.cc File Reference

```
#include <cassert>
#include "network.h"
#include "network_types.h"
#include "network_model_magic.h"
#include "network_model_emesh_hop_counter.h"
#include "network_model_emesh_hop_by_hop.h"
#include "network_model_bus.h"
#include "stats.h"
#include "log.h"
#include "config.hpp"
```

7.295 common/network/network_model.d File Reference

7.296 common/network/network_model.h File Reference

```
#include "packet_type.h"
#include "fixed_types.h"
#include "subsecond_time.h"
#include <vector>
```

Classes

- class **NetworkModel**
- struct **NetworkModel::Hop**

7.297 common/network/network_model_bus.cc File Reference

```
#include "core_manager.h"
#include "simulator.h"
#include "network.h"
#include "network_model_bus.h"
#include "memory_manager_base.h"
#include "stats.h"
#include "log.h"
#include "dvfs_manager.h"
#include "config.hpp"
```

7.298 common/network/network_model_bus.d File Reference

7.299 common/network/network_model_bus.h File Reference

```
#include "network.h"
#include "core.h"
#include "lock.h"
#include "subsecond_time.h"
#include "queue_model.h"
#include "contention_model.h"
```

Classes

- class **NetworkModelBusGlobal**
- class **NetworkModelBus**

7.300 common/network/network_model_emesh_hop_by_hop.cc File Reference

```
#include "network_model_emesh_hop_by_hop.h"
#include "core.h"
#include "simulator.h"
#include "config.h"
#include "utils.h"
#include "packet_type.h"
#include "queue_model_history_list.h"
#include "memory_manager_base.h"
#include "dvfs_manager.h"
#include "stats.h"
#include "config.hpp"
#include <math.h>
#include <stdlib.h>
```

Functions

- const char * **OutputDirectionString** (**NetworkModelEMeshHopByHop::OutputDirection** direction)

Variables

- const char * **output_direction_names** [ ]

7.300.1 Function Documentation

7.300.1.1 OutputDirectionString()

```
const char* OutputDirectionString (
              NetworkModelEMeshHopByHop::OutputDirection direction )
```

Definition at line 22 of file network_model_emesh_hop_by_hop.cc.

References LOG_ASSERT_ERROR, NetworkModelEMeshHopByHop::MAX_OUTPUT_DIRECTIONS, and output_direction_names.

7.300.2 Variable Documentation

7.300.2.1 output_direction_names

```
const char* output_direction_names[]
```

Initial value:

```
= {
   "up", "down", "left", "right", "---", "self", "peer", "destination"
}
```

Definition at line 16 of file network_model_emesh_hop_by_hop.cc.

Referenced by OutputDirectionString().

7.301 common/network/network_model_emesh_hop_by_hop.d File Reference

7.302 common/network/network_model_emesh_hop_by_hop.h File Reference

```
#include "network.h"
#include "network_model.h"
#include "fixed_types.h"
#include "queue_model.h"
#include "lock.h"
#include "subsecond_time.h"
```

Classes

- class **NetworkModelEMeshHopByHop**

7.303 common/network/network_model_emesh_hop_counter.cc File Reference

```
#include <stdlib.h>
#include <math.h>
#include "simulator.h"
#include "config.h"
#include "network_model_emesh_hop_counter.h"
#include "core.h"
#include "memory_manager_base.h"
#include "dvfs_manager.h"
#include "config.hpp"
```

7.304 common/network/network_model_emesh_hop_counter.d File Reference

7.305 common/network/network_model_emesh_hop_counter.h File Reference

```
#include "network.h"
#include "network_model.h"
#include "lock.h"
```

Classes

- class **NetworkModelEMeshHopCounter**

7.306 common/network/network_model_magic.cc File Reference

```
#include "simulator.h"
#include "network.h"
#include "network_model_magic.h"
#include "memory_manager_base.h"
#include "log.h"
#include "dvfs_manager.h"
```

7.307 common/network/network_model_magic.d File Reference

7.308 common/network/network_model_magic.h File Reference

```
#include "network.h"
#include "core.h"
#include "lock.h"
#include "subsecond_time.h"
```

Classes

- class **NetworkModelMagic**

7.309 common/network/network_types.h File Reference

Enumerations

- enum **NetworkType** {
  NETWORK_MAGIC, **NETWORK_EMESH_HOP_COUNTER**, **NETWORK_EMESH_HOP_BY_HOP**, **N↩ ETWORK_BUS**,
  NUM_NETWORK_TYPES }

7.309.1 Enumeration Type Documentation

7.309.1.1 NetworkType

enum **NetworkType**

Enumerator

NETWORK_MAGIC	
NETWORK_EMESH_HOP_COUNTER	
NETWORK_EMESH_HOP_BY_HOP	
NETWORK_BUS	
NUM_NETWORK_TYPES	

Definition at line 4 of file network_types.h.

7.310 common/network/packet_type.cc File Reference

Variables

- const char * **EStaticNetworkStrings** [ ]

7.310.1 Variable Documentation

7.310.1.1 EStaticNetworkStrings

```
const char* EStaticNetworkStrings[]
```

Initial value:
```
= {
   "shmem-1",
   "system",
}
```

Definition at line 2 of file packet_type.cc.

Referenced by NetworkModel::NetworkModel(), NetworkModelBus::NetworkModelBus(), and NetworkModelE↩ MeshHopByHop::NetworkModelEMeshHopByHop().

7.311 common/network/packet_type.d File Reference

7.312 common/network/packet_type.h File Reference

Enumerations

- enum **PacketType** {
  INVALID_PACKET_TYPE, **SHARED_MEM_1**, **SIM_THREAD_TERMINATE_THREADS**, **CORE_THR↩ EAD_TERMINATE_THREADS**,
  NUM_PACKET_TYPES }
- enum **EStaticNetwork** { **STATIC_NETWORK_MEMORY_1**, **STATIC_NETWORK_SYSTEM**, **NUM_ST↩ ATIC_NETWORKS** }

Functions

- static **EStaticNetwork** g_type_to_static_network_map[ ] **__attribute__** ((unused))

Variables

- const char ∗ **EStaticNetworkStrings** [ ]

7.312.1 Enumeration Type Documentation

7.312.1.1 EStaticNetwork

```
enum EStaticNetwork
```

Enumerator

STATIC_NETWORK_MEMORY↩ _1	
STATIC_NETWORK_SYSTEM	
NUM_STATIC_NETWORKS	

Definition at line 14 of file packet_type.h.

7.312.1.2 PacketType

```
enum PacketType
```

Enumerator

INVALID_PACKET_TYPE	
SHARED_MEM_1	
SIM_THREAD_TERMINATE_THREADS	
CORE_THREAD_TERMINATE_THREADS	
NUM_PACKET_TYPES	

Definition at line 4 of file packet_type.h.

7.312.2 Function Documentation

7.312.2.1 __attribute__()

```
static EStaticNetwork g_type_to_static_network_map [] __attribute__ (
            (unused) ) [static]
```

Definition at line 127 of file simulator.h.

References Simulator::getSingleton().

7.312.3 Variable Documentation

7.312.3.1 EStaticNetworkStrings

```
const char* EStaticNetworkStrings[]
```

Definition at line 2 of file packet_type.cc.

Referenced by NetworkModel::NetworkModel(), NetworkModelBus::NetworkModelBus(), and NetworkModelE↩ MeshHopByHop::NetworkModelEMeshHopByHop().

7.313 common/performance_model/branch_predictor.cc File Reference

```
#include "simulator.h"
#include "branch_predictor.h"
#include "one_bit_branch_predictor.h"
#include "pentium_m_branch_predictor.h"
#include "config.hpp"
#include "stats.h"
```

7.314 common/performance_model/branch_predictor.d File Reference

7.315 common/performance_model/branch_predictor.h File Reference

```
#include <iostream>
#include "fixed_types.h"
```

Classes

- class **BranchPredictor**

7.316 common/performance_model/branch_predictors/branch_↩ predictor_return_value.cc File Reference

```
#include "branch_predictor_return_value.h"
```

7.317 common/performance_model/branch_predictors/branch_↩ predictor_return_value.d File Reference

7.318 common/performance_model/branch_predictors/branch_↩ predictor_return_value.h File Reference

```
#include "fixed_types.h"
#include <ostream>
#include <ios>
#include <boost/io/ios_state.hpp>
```

Classes

- class **BranchPredictorReturnValue**

Functions

- std::ostream & **operator**<< (std::ostream &stream, const **BranchPredictorReturnValue** &value)

7.318.1 Function Documentation

7.318.1.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & stream,
            const  BranchPredictorReturnValue & value )  [inline]
```

Definition at line 34 of file branch_predictor_return_value.h.

References BranchPredictorReturnValue::BranchTypeNames, BranchPredictorReturnValue::hit, BranchPredictor↩ ReturnValue::prediction, BranchPredictorReturnValue::target, and BranchPredictorReturnValue::type.

7.319 common/performance_model/branch_predictors/btb.h File Reference

```
#include "branch_predictor.h"
```

Classes

- class **BranchTargetBuffer**

7.320 common/performance_model/branch_predictors/global_↩ predictor.h File Reference

```
#include <vector>
#include <stdint.h>
#include "simulator.h"
#include "branch_predictor.h"
#include "branch_predictor_return_value.h"
#include "saturating_predictor.h"
```

Classes

- class **GlobalPredictor**
- class **GlobalPredictor::Way**

7.321 common/performance_model/branch_predictors/ibtb.h File Reference

```
#include <boost/scoped_array.hpp>
#include "simulator.h"
#include "branch_predictor.h"
```

Classes

- class **IndirectBranchTargetBuffer**

7.322 common/performance_model/branch_predictors/lpb.h File Reference

```
#include <vector>
#include <stdint.h>
#include "simulator.h"
#include "branch_predictor.h"
#include "branch_predictor_return_value.h"
#include "saturating_predictor.h"
```

Classes

- class **LoopBranchPredictor**
- class **LoopBranchPredictor::Way**

Macros

- #define **DEBUG** 0
- #define **debug_cout** if (0) std::cout

7.322.1 Macro Definition Documentation

7.322.1.1 DEBUG

```
#define DEBUG 0
```

Definition at line 12 of file lpb.h.

7.322.1.2 debug_cout

```
#define debug_cout if (0) std::cout
```

Definition at line 15 of file lpb.h.

7.323 common/performance_model/branch_predictors/one_bit_branch_predictor.cc File Reference

```
#include "simulator.h"
#include "one_bit_branch_predictor.h"
```

7.324 common/performance_model/branch_predictors/one_bit_branch_predictor.d File Reference

7.325 common/performance_model/branch_predictors/one_bit_branch_predictor.h File Reference

```
#include "branch_predictor.h"
#include <vector>
```

Classes

- class **OneBitBranchPredictor**

7.326 common/performance_model/branch_predictors/pentium_m_bimodal_table.h File Reference

```
#include "simple_bimodal_table.h"
```

Classes

- class **PentiumMBimodalTable**

7.327 common/performance_model/branch_predictors/pentium_m_branch_predictor.cc File Reference

```
#include "simulator.h"
#include "pentium_m_branch_predictor.h"
```

7.328 common/performance_model/branch_predictors/pentium_m_branch_predictor.d File Reference

7.329 common/performance_model/branch_predictors/pentium_m_branch_predictor.h File Reference

```
#include "branch_predictor.h"
#include "branch_predictor_return_value.h"
#include "pentium_m_global_predictor.h"
#include "pentium_m_branch_target_buffer.h"
#include "pentium_m_bimodal_table.h"
#include "pentium_m_loop_branch_predictor.h"
#include <vector>
```

Classes

- class **PentiumMBranchPredictor**

7.330 common/performance_model/branch_predictors/pentium_m_branch_target_buffer.h File Reference

```
#include <vector>
#include "branch_predictor.h"
```

Classes

- class **PentiumMBranchTargetBuffer**
- class **PentiumMBranchTargetBuffer::Way**

Macros

- #define **NUM_WAYS** 4
- #define **NUM_ENTRIES** 512
- #define **IP_TO_INDEX**(_ip) ((_ip >> 4) & 0x1ff)
- #define **TAG_OFFSET_MASK** 0x3fe00f
- #define **IP_TO_TAGOFF**(_ip) (_ip & **TAG_OFFSET_MASK**)

7.330.1 Macro Definition Documentation

7.330.1.1 IP_TO_INDEX

```
#define IP_TO_INDEX(
              _ip ) ((_ip >> 4) & 0x1ff)
```

Definition at line 13 of file pentium_m_branch_target_buffer.h.

7.330.1.2 IP_TO_TAGOFF

```
#define IP_TO_TAGOFF(
              _ip ) (_ip &  TAG_OFFSET_MASK)
```

Definition at line 16 of file pentium_m_branch_target_buffer.h.

7.330.1.3 NUM_ENTRIES

```
#define NUM_ENTRIES 512
```

Definition at line 9 of file pentium_m_branch_target_buffer.h.

7.330.1.4 NUM_WAYS

```
#define NUM_WAYS 4
```

Definition at line 8 of file pentium_m_branch_target_buffer.h.

7.330.1.5 TAG_OFFSET_MASK

```
#define TAG_OFFSET_MASK 0x3fe00f
```

Definition at line 15 of file pentium_m_branch_target_buffer.h.

7.331 common/performance_model/branch_predictors/pentium_m_global_predictor.h File Reference

```
#include "global_predictor.h"
```

Classes

- class **PentiumMGlobalPredictor**

7.332 common/performance_model/branch_predictors/pentium_m_indirect_branch_target_buffer.h File Reference

```
#include "ibtb.h"
```

Classes

- class **PentiumMIndirectBranchTargetBuffer**

7.333 common/performance_model/branch_predictors/pentium_m_loop_branch_predictor.h File Reference

```
#include "lpb.h"
```

Classes

- class **PentiumMLoopBranchPredictor**

7.334 common/performance_model/branch_predictors/saturating_↩ predictor.h File Reference

```
#include <stdint.h>
#include "fixed_types.h"
```

Classes

- class **Pow2**< **N** >
- class **Pow2**< **0** >
- class **SaturatingPredictor**< **n** >

Macros

- #define **SAT_PRED_DEBUG** 0

7.334.1 Macro Definition Documentation

7.334.1.1 SAT_PRED_DEBUG

```
#define SAT_PRED_DEBUG 0
```

Definition at line 20 of file saturating_predictor.h.

7.335 common/performance_model/branch_predictors/simple_bimodal↩ _table.h File Reference

```
#include <boost/scoped_array.hpp>
#include "simulator.h"
#include "branch_predictor.h"
#include "saturating_predictor.h"
```

Classes

- class **SimpleBimodalTable**

7.336 common/performance_model/cache_perf_model.cc File Reference

```
#include "cache_perf_model.h"
#include "cache_perf_model_parallel.h"
#include "cache_perf_model_sequential.h"
#include "log.h"
```

7.337 common/performance_model/cache_perf_model.d File Reference

7.338 common/performance_model/cache_perf_model.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
```

Classes

- class **CachePerfModel**

7.339 common/performance_model/cache_perf_model_parallel.h File Reference

```
#include "cache_perf_model.h"
```

Classes

- class **CachePerfModelParallel**

7.340 common/performance_model/cache_perf_model_sequential.h File Reference

```
#include "cache_perf_model.h"
```

Classes

- class **CachePerfModelSequential**

7.341 common/performance_model/contention_model.cc File Reference

```
#include "contention_model.h"
#include "stats.h"
#include "subsecond_time.h"
#include "dvfs_manager.h"
```

7.342 common/performance_model/contention_model.d File Reference

7.343 common/performance_model/contention_model.h File Reference

```
#include <vector>
#include "fixed_types.h"
#include "subsecond_time.h"
```

Classes

- class **ContentionModel**

7.344 common/performance_model/dram_directory_perf_model.h File Reference

```
#include "dram_directory_perf_model_base.h"
```

Classes

- class **DramDirectoryPerfModel**

7.345 common/performance_model/dram_directory_perf_model_base.cc File Reference

```
#include "dram_directory_perf_model_base.h"
#include "dram_directory_perf_model.h"
#include "log.h"
```

7.346 common/performance_model/dram_directory_perf_model_base.d File Reference

7.347 common/performance_model/dram_directory_perf_model_base.h File Reference

```
#include "log.h"
#include "config.hpp"
#include "simulator.h"
```

Classes

- class **DramDirectoryPerfModelBase**

7.348 common/performance_model/dram_perf_model.cc File Reference

```
#include "simulator.h"
#include "dram_perf_model.h"
#include "dram_perf_model_constant.h"
#include "dram_perf_model_readwrite.h"
#include "dram_perf_model_normal.h"
#include "config.hpp"
```

7.349 common/performance_model/dram_perf_model.d File Reference

7.350 common/performance_model/dram_perf_model.h File Reference

```
#include "queue_model.h"
#include "fixed_types.h"
#include "subsecond_time.h"
#include "dram_cntlr_interface.h"
```

Classes

- class **DramPerfModel**

7.351 common/performance_model/dram_perf_model_constant.cc File Reference

```
#include "dram_perf_model_constant.h"
#include "simulator.h"
#include "config.h"
#include "config.hpp"
#include "stats.h"
#include "shmem_perf.h"
```

7.352 common/performance_model/dram_perf_model_constant.d File Reference

7.353 common/performance_model/dram_perf_model_constant.h File Reference

```
#include "dram_perf_model.h"
#include "queue_model.h"
#include "fixed_types.h"
#include "subsecond_time.h"
#include "dram_cntlr_interface.h"
```

Classes

- class **DramPerfModelConstant**

7.354 common/performance_model/dram_perf_model_normal.cc File Reference

```
#include "dram_perf_model_normal.h"
#include "simulator.h"
#include "config.h"
#include "config.hpp"
#include "stats.h"
#include "shmem_perf.h"
```

7.355 common/performance_model/dram_perf_model_normal.d File Reference

7.356 common/performance_model/dram_perf_model_normal.h File Reference

```
#include "dram_perf_model.h"
#include "queue_model.h"
#include "fixed_types.h"
#include "subsecond_time.h"
#include "dram_cntlr_interface.h"
#include "distribution.h"
```

Classes

- class **DramPerfModelNormal**

7.357 common/performance_model/dram_perf_model_readwrite.cc File Reference

```
#include "dram_perf_model_readwrite.h"
#include "simulator.h"
#include "config.h"
#include "config.hpp"
#include "stats.h"
#include "shmem_perf.h"
```

7.358 common/performance_model/dram_perf_model_readwrite.d File Reference

7.359 common/performance_model/dram_perf_model_readwrite.h File Reference

```
#include "dram_perf_model.h"
#include "queue_model.h"
#include "fixed_types.h"
#include "subsecond_time.h"
#include "dram_cntlr_interface.h"
```

Classes

- class **DramPerfModelReadWrite**

7.360 common/performance_model/dynamic_instruction.cc File Reference

```
#include "dynamic_instruction.h"
#include "instruction.h"
#include "allocator.h"
#include "core.h"
#include "branch_predictor.h"
#include "performance_model.h"
```

7.361 common/performance_model/dynamic_instruction.d File Reference

7.362 common/performance_model/dynamic_instruction.h File Reference

```
#include "operand.h"
#include "subsecond_time.h"
#include "hit_where.h"
#include "allocator.h"
```

Classes

- class **DynamicInstruction**
- struct **DynamicInstruction::BranchInfo**
- struct **DynamicInstruction::MemoryInfo**

7.363 common/performance_model/fastforward_performance_model.cc File Reference

```
#include "fastforward_performance_model.h"
#include "fastforward_performance_manager.h"
#include "simulator.h"
#include "core.h"
#include "sampling_manager.h"
#include "stats.h"
#include "config.hpp"
#include "instruction.h"
```

7.364 common/performance_model/fastforward_performance_model.d File Reference

7.365 common/performance_model/fastforward_performance_model.h File Reference

```
#include "performance_model.h"
```

Classes

- class **FastforwardPerformanceModel**

7.366 common/performance_model/hit_where.cc File Reference

```
#include "hit_where.h"
```

Functions

- const char * **HitWhereString** (**HitWhere::where_t** where)
- bool **HitWhereIsValid** (**HitWhere::where_t** where)

7.366.1 Function Documentation

7.366.1.1 HitWhereIsValid()

```
bool HitWhereIsValid (
              HitWhere::where_t where )
```

Definition at line 29 of file hit_where.cc.

References HitWhereString().

Referenced by FastforwardPerformanceModel::FastforwardPerformanceModel(), RobSmtTimer::initializeThread(), IntervalTimer::IntervalTimer(), OneIPCPerformanceModel::OneIPCPerformanceModel(), RobTimer::RobTimer(), RoutineTracerFunctionStats::ThreadStatCpiMem::ThreadStatCpiMem(), and MemoryTracker::~MemoryTracker().

7.366.1.2 HitWhereString()

```
const char* HitWhereString (
              HitWhere::where_t where )
```

Definition at line 3 of file hit_where.cc.

References HitWhere::CACHE_REMOTE, HitWhere::DRAM, HitWhere::DRAM_CACHE, HitWhere::DRAM_LOC↩
AL, HitWhere::DRAM_REMOTE, HitWhere::L1_OWN, HitWhere::L1_SIBLING, HitWhere::L1I, HitWhere::L2_OWN, HitWhere::L2_SIBLING, HitWhere::L3_OWN, HitWhere::L3_SIBLING, HitWhere::L4_OWN, HitWhere::L4_SIBLI↩
NG, HitWhere::MISS, HitWhere::NUCA_CACHE, HitWhere::PREDICATE_FALSE, HitWhere::PREFETCH_NO_↩
MAPPING, and HitWhere::UNKNOWN.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), FastforwardPerformanceModel::↩
FastforwardPerformanceModel(), HitWhereIsValid(), RobSmtTimer::initializeThread(), IntervalTimer::Interval↩
Timer(), OneIPCPerformanceModel::OneIPCPerformanceModel(), ParametricDramDirectoryMSI::CacheCntlr↩
::processMemOpFromCore(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache(), RobTimer::RobTimer(), RoutineTracerFunctionStats::ThreadStatCpiMem::ThreadStatCpiMem(), Parametric↩
DramDirectoryMSI::CacheCntlr::∼CacheCntlr(), and MemoryTracker::∼MemoryTracker().

7.367 common/performance_model/hit_where.d File Reference

7.368 common/performance_model/hit_where.h File Reference

```
#include "mem_component.h"
#include <cstddef>
#include <functional>
```

Classes

- class **HitWhere**
- struct **std::hash**< **HitWhere::where_t** >

Namespaces

- **std**

Functions

- const char * **HitWhereString** (**HitWhere::where_t** where)
- bool **HitWhereIsValid** (**HitWhere::where_t** where)

7.368.1 Function Documentation

7.368.1.1 HitWhereIsValid()

```
bool HitWhereIsValid (
            HitWhere::where_t where )
```

Definition at line 29 of file hit_where.cc.

References HitWhereString().

Referenced by FastforwardPerformanceModel::FastforwardPerformanceModel(), RobSmtTimer::initializeThread(), IntervalTimer::IntervalTimer(), OneIPCPerformanceModel::OneIPCPerformanceModel(), RobTimer::RobTimer(), RoutineTracerFunctionStats::ThreadStatCpiMem::ThreadStatCpiMem(), and MemoryTracker::~MemoryTracker().

7.368.1.2 HitWhereString()

```
const char* HitWhereString (
            HitWhere::where_t where )
```

Definition at line 3 of file hit_where.cc.

References HitWhere::CACHE_REMOTE, HitWhere::DRAM, HitWhere::DRAM_CACHE, HitWhere::DRAM_LOC↩AL, HitWhere::DRAM_REMOTE, HitWhere::L1_OWN, HitWhere::L1_SIBLING, HitWhere::L1I, HitWhere::L2_OWN, HitWhere::L2_SIBLING, HitWhere::L3_OWN, HitWhere::L3_SIBLING, HitWhere::L4_OWN, HitWhere::L4_SIBLI↩NG, HitWhere::MISS, HitWhere::NUCA_CACHE, HitWhere::PREDICATE_FALSE, HitWhere::PREFETCH_NO_↩MAPPING, and HitWhere::UNKNOWN.

Referenced by ParametricDramDirectoryMSI::CacheCntlr::CacheCntlr(), FastforwardPerformanceModel::↩FastforwardPerformanceModel(), HitWhereIsValid(), RobSmtTimer::initializeThread(), IntervalTimer::Interval↩Timer(), OneIPCPerformanceModel::OneIPCPerformanceModel(), ParametricDramDirectoryMSI::CacheCntlr↩::processMemOpFromCore(), ParametricDramDirectoryMSI::CacheCntlr::processShmemReqFromPrevCache(), RobTimer::RobTimer(), RoutineTracerFunctionStats::ThreadStatCpiMem::ThreadStatCpiMem(), Parametric↩DramDirectoryMSI::CacheCntlr::~CacheCntlr(), and MemoryTracker::~MemoryTracker().

7.369 common/performance_model/instruction.cc File Reference

```
#include "instruction.h"
#include "simulator.h"
#include "core_manager.h"
#include "core.h"
#include "performance_model.h"
#include "branch_predictor.h"
#include "config.hpp"
```

7.370 common/performance_model/instruction.d File Reference

7.371 common/performance_model/instruction.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include "operand.h"
#include <vector>
#include <sstream>
```

Classes

- class **Instruction**
- class **GenericInstruction**
- class **ArithInstruction**
- class **JmpInstruction**
- class **PseudoInstruction**
- class **RecvInstruction**
- class **SyncInstruction**
- class **SpawnInstruction**
- class **BranchInstruction**
- class **TLBMissInstruction**
- class **MemAccessInstruction**
- class **DelayInstruction**
- class **UnknownInstruction**

Enumerations

- enum **InstructionType** {
  INST_GENERIC, **INST_ADD**, **INST_SUB**, **INST_MUL**,
  INST_DIV, **INST_FADD**, **INST_FSUB**, **INST_FMUL**,
  INST_FDIV, **INST_JMP**, **INST_BRANCH**, **INST_PSEUDO_MISC**,
  INST_RECV, **INST_SYNC**, **INST_SPAWN**, **INST_TLB_MISS**,
  INST_MEM_ACCESS, **INST_DELAY**, **INST_UNKNOWN**, **MAX_INSTRUCTION_COUNT** }

Functions

- **__attribute__** ((unused)) static const char ∗INSTRUCTION_NAMES[ ]

7.371.1 Enumeration Type Documentation

7.371.1.1 InstructionType

enum **InstructionType**

Enumerator

INST_GENERIC	
INST_ADD	
INST_SUB	
INST_MUL	
INST_DIV	
INST_FADD	
INST_FSUB	
INST_FMUL	
INST_FDIV	
INST_JMP	
INST_BRANCH	

Enumerator

INST_PSEUDO_MISC	
INST_RECV	
INST_SYNC	
INST_SPAWN	
INST_TLB_MISS	
INST_MEM_ACCESS	
INST_DELAY	
INST_UNKNOWN	
MAX_INSTRUCTION_COUNT	

Definition at line 13 of file instruction.h.

7.371.2 Function Documentation

7.371.2.1 __attribute__()

```
__attribute__ (
            (unused)  ) const
```

7.372 common/performance_model/instruction_tracers/instruction_↩ tracer.cc File Reference

```
#include "instruction_tracer.h"
#include "simulator.h"
#include "config.hpp"
#include "instruction_tracer_fpstats.h"
#include "instruction_tracer_print.h"
#include "loop_tracer.h"
#include "loop_profiler.h"
```

7.373 common/performance_model/instruction_tracers/instruction_↩ tracer.d File Reference

7.374 common/performance_model/instruction_tracers/instruction_↩ tracer.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
```

Classes

- class **InstructionTracer**
- struct **InstructionTracer::uop_times_t**

7.375 common/performance_model/instruction_tracers/instruction_tracer_fpstats.cc File Reference

```
#include "fixed_types.h"
#include "instruction_tracer_fpstats.h"
#include "stats.h"
#include "thread_stats_manager.h"
#include "instruction.h"
#include "micro_op.h"
#include "dynamic_micro_op.h"
```

Variables

- const char *const **fp_iclasses** [ ]

7.375.1 Variable Documentation

7.375.1.1 fp_iclasses

```
const char* const fp_iclasses[]
```

Initial value:

```
= {
   "ADDPD", "ADDSD", "ADDSS", "ADDPS",
   "SUBPD", "SUBSD", "SUBSS", "SUBPS",
   "MULPD", "MULSD", "MULSS", "MULPS",
   "DIVPD", "DIVSD", "DIVSS", "DIVPS",
}
```

Definition at line 9 of file instruction_tracer_fpstats.cc.

Referenced by InstructionTracerFPStats::init(), and InstructionTracerFPStats::InstructionTracerFPStats().

7.376 common/performance_model/instruction_tracers/instruction_tracer_fpstats.d File Reference

7.377 common/performance_model/instruction_tracers/instruction_tracer_fpstats.h File Reference

```
#include "instruction_tracer.h"
#include <xed-iclass-enum.h>
#include <unordered_map>
```

Classes

- class **InstructionTracerFPStats**

7.378 common/performance_model/instruction_tracers/instruction_↩ tracer_print.cc File Reference

```
#include "instruction_tracer_print.h"
#include "core.h"
#include "dynamic_micro_op.h"
```

7.379 common/performance_model/instruction_tracers/instruction_↩ tracer_print.d File Reference

7.380 common/performance_model/instruction_tracers/instruction_↩ tracer_print.h File Reference

```
#include "instruction_tracer.h"
```

Classes

- class **InstructionTracerPrint**

7.381 common/performance_model/instruction_tracers/loop_profiler.cc File Reference

```
#include "loop_profiler.h"
#include "dynamic_micro_op.h"
#include "instruction.h"
#include <list>
```

7.382 common/performance_model/instruction_tracers/loop_profiler.d File Reference

7.383 common/performance_model/instruction_tracers/loop_profiler.h File Reference

```
#include "instruction_tracer.h"
#include <vector>
#include <unordered_map>
```

Classes

- class **LoopProfiler**
- struct **LoopProfiler::Loop**

7.384 common/performance_model/instruction_tracers/loop_tracer.cc File Reference

```
#include "loop_tracer.h"
#include "simulator.h"
#include "config.hpp"
#include "dynamic_micro_op.h"
#include "instruction.h"
#include "core.h"
```

7.385 common/performance_model/instruction_tracers/loop_tracer.d File Reference

7.386 common/performance_model/instruction_tracers/loop_tracer.h File Reference

```
#include "instruction_tracer.h"
#include <map>
```

Classes

- class **LoopTracer**
- class **LoopTracer::Instr**

7.387 common/performance_model/mmu_perf_model.h File Reference

```
#include "mmu_perf_model_base.h"
#include "log.h"
```

Classes

- class **MMUPerfModel**

7.388 common/performance_model/mmu_perf_model_base.cc File Reference

```
#include "mmu_perf_model_base.h"
#include "mmu_perf_model.h"
#include "log.h"
```

7.389 common/performance_model/mmu_perf_model_base.d File Reference

7.390 common/performance_model/mmu_perf_model_base.h File Reference

```
#include "config.h"
#include "log.h"
```

Classes

- class **MMUPerfModelBase**

7.391 common/performance_model/operand.h File Reference

```
#include "fixed_types.h"
#include <sstream>
#include <iostream>
#include <vector>
```

Classes

- class **Operand**

Typedefs

- typedef std::vector< **Operand** > **OperandList**

7.391.1 Typedef Documentation

7.391.1.1 OperandList

```
typedef std::vector< Operand>  OperandList
```

Definition at line 69 of file operand.h.

7.392 common/performance_model/performance_model.cc File Reference

```
#include "core.h"
#include "performance_model.h"
#include "fastforward_performance_model.h"
#include "branch_predictor.h"
#include "simulator.h"
#include "oneipc_performance_model.h"
#include "interval_performance_model.h"
#include "rob_performance_model.h"
#include "rob_smt_performance_model.h"
#include "core_manager.h"
#include "config.hpp"
#include "stats.h"
#include "dvfs_manager.h"
#include "instruction_tracer.h"
#include "dynamic_instruction.h"
```

7.393 common/performance_model/performance_model.d File Reference

7.394 common/performance_model/performance_model.h File Reference

```
#include "fixed_types.h"
#include "mt_circular_queue.h"
#include "lock.h"
#include "subsecond_time.h"
#include "instruction_tracer.h"
#include "hit_where.h"
#include <queue>
#include <iostream>
```

Classes

- class **PerformanceModel**

7.395 common/performance_model/performance_models/core_model/core_model.cc File Reference

```
#include "core_model.h"
#include "core_model_nehalem.h"
#include "core_model_boom_v1.h"
#include "log.h"
```

7.396 common/performance_model/performance_models/core_model/core_model.d File Reference

7.397 common/performance_model/performance_models/core_model/core_model.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include "allocator.h"
#include "dynamic_micro_op.h"
#include <map>
```

Classes

- class **CoreModel**
- class **BaseCoreModel**< **T** >

7.398 common/performance_model/performance_models/core_model/core_model_boom_v1.cc File Reference

```
#include "core_model_boom_v1.h"
#include "interval_contention_boom_v1.h"
#include "rob_contention_boom_v1.h"
#include "dynamic_micro_op_boom_v1.h"
#include "log.h"
#include "config.hpp"
#include "simulator.h"
#include <decoder.h>
#include <riscv_meta.h>
```

Variables

- static unsigned int **instructionLatencies** [**rv_op_last**]
- static unsigned int **bypassLatencies** [**DynamicMicroOpBoomV1::UOP_BYPASS_SIZE**]

7.398.1 Variable Documentation

7.398.1.1 bypassLatencies

```
unsigned int bypassLatencies[ DynamicMicroOpBoomV1::UOP_BYPASS_SIZE]  [static]
```

Definition at line 24 of file core_model_boom_v1.cc.

Referenced by CoreModelBoomV1::getBypassLatency().

7.398.1.2 instructionLatencies

```
unsigned int instructionLatencies[ rv_op_last]  [static]
```

Definition at line 23 of file core_model_boom_v1.cc.

Referenced by CoreModelBoomV1::CoreModelBoomV1(), and CoreModelBoomV1::getInstructionLatency().

7.399 common/performance_model/performance_models/core_↩ model/core_model_boom_v1.d File Reference

7.400 common/performance_model/performance_models/core_↩ model/core_model_boom_v1.h File Reference

```
#include "core_model.h"
#include "dynamic_micro_op_boom_v1.h"
```

Classes

- class **CoreModelBoomV1**

7.401 common/performance_model/performance_models/core_↩ model/core_model_nehalem.cc File Reference

```
#include "core_model_nehalem.h"
#include "interval_contention_nehalem.h"
#include "rob_contention_nehalem.h"
#include "dynamic_micro_op_nehalem.h"
#include "log.h"
#include "config.hpp"
#include "simulator.h"
```

Variables

- static unsigned int **instructionLatencies** [XED_ICLASS_LAST]
- static unsigned int **bypassLatencies** [**DynamicMicroOpNehalem::UOP_BYPASS_SIZE**]

7.401.1 Variable Documentation

7.401.1.1 bypassLatencies

`unsigned int bypassLatencies[ DynamicMicroOpNehalem::UOP_BYPASS_SIZE] [static]`

Definition at line 10 of file core_model_nehalem.cc.

Referenced by CoreModelNehalem::CoreModelNehalem(), and CoreModelNehalem::getBypassLatency().

7.401.1.2 instructionLatencies

`unsigned int instructionLatencies[XED_ICLASS_LAST] [static]`

Definition at line 9 of file core_model_nehalem.cc.

Referenced by CoreModelNehalem::CoreModelNehalem(), and CoreModelNehalem::getInstructionLatency().

7.402 common/performance_model/performance_models/core_↩ model/core_model_nehalem.d File Reference

7.403 common/performance_model/performance_models/core_↩ model/core_model_nehalem.h File Reference

```
#include "core_model.h"
#include "dynamic_micro_op_nehalem.h"
```

Classes

- class **CoreModelNehalem**

7.404 common/performance_model/performance_models/core_↩ model/dynamic_micro_op_boom_v1.cc File Reference

```
#include "dynamic_micro_op_boom_v1.h"
#include "micro_op.h"
#include <riscv_meta.h>
```

7.405 common/performance_model/performance_models/core_↩ model/dynamic_micro_op_boom_v1.d File Reference

7.406 common/performance_model/performance_models/core_↩ model/dynamic_micro_op_boom_v1.h File Reference

```
#include "dynamic_micro_op.h"
```

Classes

- class **DynamicMicroOpBoomV1**

7.407 common/performance_model/performance_models/core_↩ model/dynamic_micro_op_nehalem.cc File Reference

```
#include "dynamic_micro_op_nehalem.h"
#include "micro_op.h"
```

7.408 common/performance_model/performance_models/core_↩ model/dynamic_micro_op_nehalem.d File Reference

7.409 common/performance_model/performance_models/core_↩ model/dynamic_micro_op_nehalem.h File Reference

```
#include "dynamic_micro_op.h"
#include <xed-iclass-enum.h>
```

Classes

- class **DynamicMicroOpNehalem**

7.410 common/performance_model/performance_models/core_↩ model/riscv_meta.h File Reference

Classes

- struct **riscvinstr**

Enumerations

- enum **rv_op** {
  rv_op_illegal = 0, **rv_op_lui** = 1, **rv_op_auipc** = 2, **rv_op_jal** = 3,
  rv_op_jalr = 4, **rv_op_beq** = 5, **rv_op_bne** = 6, **rv_op_blt** = 7,
  rv_op_bge = 8, **rv_op_bltu** = 9, **rv_op_bgeu** = 10, **rv_op_lb** = 11,
  rv_op_lh = 12, **rv_op_lw** = 13, **rv_op_lbu** = 14, **rv_op_lhu** = 15,
  rv_op_sb = 16, **rv_op_sh** = 17, **rv_op_sw** = 18, **rv_op_addi** = 19,
  rv_op_slti = 20, **rv_op_sltiu** = 21, **rv_op_xori** = 22, **rv_op_ori** = 23,
  rv_op_andi = 24, **rv_op_slli** = 25, **rv_op_srli** = 26, **rv_op_srai** = 27,
  rv_op_add = 28, **rv_op_sub** = 29, **rv_op_sll** = 30, **rv_op_slt** = 31,
  rv_op_sltu = 32, **rv_op_xor** = 33, **rv_op_srl** = 34, **rv_op_sra** = 35,
  rv_op_or = 36, **rv_op_and** = 37, **rv_op_fence** = 38, **rv_op_fence_i** = 39,
  rv_op_lwu = 40, **rv_op_ld** = 41, **rv_op_sd** = 42, **rv_op_addiw** = 43,
  rv_op_slliw = 44, **rv_op_srliw** = 45, **rv_op_sraiw** = 46, **rv_op_addw** = 47,
  rv_op_subw = 48, **rv_op_sllw** = 49, **rv_op_srlw** = 50, **rv_op_sraw** = 51,
  rv_op_ldu = 52, **rv_op_lq** = 53, **rv_op_sq** = 54, **rv_op_addid** = 55,
  rv_op_sllid = 56, **rv_op_srlid** = 57, **rv_op_sraid** = 58, **rv_op_addd** = 59,
  rv_op_subd = 60, **rv_op_slld** = 61, **rv_op_srld** = 62, **rv_op_srad** = 63,
  rv_op_mul = 64, **rv_op_mulh** = 65, **rv_op_mulhsu** = 66, **rv_op_mulhu** = 67,
  rv_op_div = 68, **rv_op_divu** = 69, **rv_op_rem** = 70, **rv_op_remu** = 71,
  rv_op_mulw = 72, **rv_op_divw** = 73, **rv_op_divuw** = 74, **rv_op_remw** = 75,
  rv_op_remuw = 76, **rv_op_muld** = 77, **rv_op_divd** = 78, **rv_op_divud** = 79,
  rv_op_remd = 80, **rv_op_remud** = 81, **rv_op_lr_w** = 82, **rv_op_sc_w** = 83,

rv_op_amoswap_w = 84, **rv_op_amoadd_w** = 85, **rv_op_amoxor_w** = 86, **rv_op_amoor_w** = 87,
rv_op_amoand_w = 88, **rv_op_amomin_w** = 89, **rv_op_amomax_w** = 90, **rv_op_amominu_w** = 91,
rv_op_amomaxu_w = 92, **rv_op_lr_d** = 93, **rv_op_sc_d** = 94, **rv_op_amoswap_d** = 95,
rv_op_amoadd_d = 96, **rv_op_amoxor_d** = 97, **rv_op_amoor_d** = 98, **rv_op_amoand_d** = 99,
rv_op_amomin_d = 100, **rv_op_amomax_d** = 101, **rv_op_amominu_d** = 102, **rv_op_amomaxu_d** = 103,
rv_op_lr_q = 104, **rv_op_sc_q** = 105, **rv_op_amoswap_q** = 106, **rv_op_amoadd_q** = 107,
rv_op_amoxor_q = 108, **rv_op_amoor_q** = 109, **rv_op_amoand_q** = 110, **rv_op_amomin_q** = 111,
rv_op_amomax_q = 112, **rv_op_amominu_q** = 113, **rv_op_amomaxu_q** = 114, **rv_op_ecall** = 115,
rv_op_ebreak = 116, **rv_op_uret** = 117, **rv_op_sret** = 118, **rv_op_hret** = 119,
rv_op_mret = 120, **rv_op_dret** = 121, **rv_op_sfence_vm** = 122, **rv_op_wfi** = 123,
rv_op_csrrw = 124, **rv_op_csrrs** = 125, **rv_op_csrrc** = 126, **rv_op_csrrwi** = 127,
rv_op_csrrsi = 128, **rv_op_csrrci** = 129, **rv_op_flw** = 130, **rv_op_fsw** = 131,
rv_op_fmadd_s = 132, **rv_op_fmsub_s** = 133, **rv_op_fnmsub_s** = 134, **rv_op_fnmadd_s** = 135,
rv_op_fadd_s = 136, **rv_op_fsub_s** = 137, **rv_op_fmul_s** = 138, **rv_op_fdiv_s** = 139,
rv_op_fsgnj_s = 140, **rv_op_fsgnjn_s** = 141, **rv_op_fsgnjx_s** = 142, **rv_op_fmin_s** = 143,
rv_op_fmax_s = 144, **rv_op_fsqrt_s** = 145, **rv_op_fle_s** = 146, **rv_op_flt_s** = 147,
rv_op_feq_s = 148, **rv_op_fcvt_w_s** = 149, **rv_op_fcvt_wu_s** = 150, **rv_op_fcvt_s_w** = 151,
rv_op_fcvt_s_wu = 152, **rv_op_fmv_x_s** = 153, **rv_op_fclass_s** = 154, **rv_op_fmv_s_x** = 155,
rv_op_fcvt_l_s = 156, **rv_op_fcvt_lu_s** = 157, **rv_op_fcvt_s_l** = 158, **rv_op_fcvt_s_lu** = 159,
rv_op_fld = 160, **rv_op_fsd** = 161, **rv_op_fmadd_d** = 162, **rv_op_fmsub_d** = 163,
rv_op_fnmsub_d = 164, **rv_op_fnmadd_d** = 165, **rv_op_fadd_d** = 166, **rv_op_fsub_d** = 167,
rv_op_fmul_d = 168, **rv_op_fdiv_d** = 169, **rv_op_fsgnj_d** = 170, **rv_op_fsgnjn_d** = 171,
rv_op_fsgnjx_d = 172, **rv_op_fmin_d** = 173, **rv_op_fmax_d** = 174, **rv_op_fcvt_s_d** = 175,
rv_op_fcvt_d_s = 176, **rv_op_fsqrt_d** = 177, **rv_op_fle_d** = 178, **rv_op_flt_d** = 179,
rv_op_feq_d = 180, **rv_op_fcvt_w_d** = 181, **rv_op_fcvt_wu_d** = 182, **rv_op_fcvt_d_w** = 183,
rv_op_fcvt_d_wu = 184, **rv_op_fclass_d** = 185, **rv_op_fcvt_l_d** = 186, **rv_op_fcvt_lu_d** = 187,
rv_op_fmv_x_d = 188, **rv_op_fcvt_d_l** = 189, **rv_op_fcvt_d_lu** = 190, **rv_op_fmv_d_x** = 191,
rv_op_flq = 192, **rv_op_fsq** = 193, **rv_op_fmadd_q** = 194, **rv_op_fmsub_q** = 195,
rv_op_fnmsub_q = 196, **rv_op_fnmadd_q** = 197, **rv_op_fadd_q** = 198, **rv_op_fsub_q** = 199,
rv_op_fmul_q = 200, **rv_op_fdiv_q** = 201, **rv_op_fsgnj_q** = 202, **rv_op_fsgnjn_q** = 203,
rv_op_fsgnjx_q = 204, **rv_op_fmin_q** = 205, **rv_op_fmax_q** = 206, **rv_op_fcvt_s_q** = 207,
rv_op_fcvt_q_s = 208, **rv_op_fcvt_d_q** = 209, **rv_op_fcvt_q_d** = 210, **rv_op_fsqrt_q** = 211,
rv_op_fle_q = 212, **rv_op_flt_q** = 213, **rv_op_feq_q** = 214, **rv_op_fcvt_w_q** = 215,
rv_op_fcvt_wu_q = 216, **rv_op_fcvt_q_w** = 217, **rv_op_fcvt_q_wu** = 218, **rv_op_fclass_q** = 219,
rv_op_fcvt_l_q = 220, **rv_op_fcvt_lu_q** = 221, **rv_op_fcvt_q_l** = 222, **rv_op_fcvt_q_lu** = 223,
rv_op_fmv_x_q = 224, **rv_op_fmv_q_x** = 225, **rv_op_c_addi4spn** = 226, **rv_op_c_fld** = 227,
rv_op_c_lw = 228, **rv_op_c_flw** = 229, **rv_op_c_fsd** = 230, **rv_op_c_sw** = 231,
rv_op_c_fsw = 232, **rv_op_c_nop** = 233, **rv_op_c_addi** = 234, **rv_op_c_jal** = 235,
rv_op_c_li = 236, **rv_op_c_addi16sp** = 237, **rv_op_c_lui** = 238, **rv_op_c_srli** = 239,
rv_op_c_srai = 240, **rv_op_c_andi** = 241, **rv_op_c_sub** = 242, **rv_op_c_xor** = 243,
rv_op_c_or = 244, **rv_op_c_and** = 245, **rv_op_c_subw** = 246, **rv_op_c_addw** = 247,
rv_op_c_j = 248, **rv_op_c_beqz** = 249, **rv_op_c_bnez** = 250, **rv_op_c_slli** = 251,
rv_op_c_fldsp = 252, **rv_op_c_lwsp** = 253, **rv_op_c_flwsp** = 254, **rv_op_c_jr** = 255,
rv_op_c_mv = 256, **rv_op_c_ebreak** = 257, **rv_op_c_jalr** = 258, **rv_op_c_add** = 259,
rv_op_c_fsdsp = 260, **rv_op_c_swsp** = 261, **rv_op_c_fswsp** = 262, **rv_op_c_ld** = 263,
rv_op_c_sd = 264, **rv_op_c_addiw** = 265, **rv_op_c_ldsp** = 266, **rv_op_c_sdsp** = 267,
rv_op_c_lq = 268, **rv_op_c_sq** = 269, **rv_op_c_lqsp** = 270, **rv_op_c_sqsp** = 271,
rv_op_nop = 272, **rv_op_mv** = 273, **rv_op_not** = 274, **rv_op_neg** = 275,
rv_op_negw = 276, **rv_op_sext_w** = 277, **rv_op_seqz** = 278, **rv_op_snez** = 279,
rv_op_sltz = 280, **rv_op_sgtz** = 281, **rv_op_fmv_s** = 282, **rv_op_fabs_s** = 283,
rv_op_fneg_s = 284, **rv_op_fmv_d** = 285, **rv_op_fabs_d** = 286, **rv_op_fneg_d** = 287,
rv_op_fmv_q = 288, **rv_op_fabs_q** = 289, **rv_op_fneg_q** = 290, **rv_op_beqz** = 291,
rv_op_bnez = 292, **rv_op_blez** = 293, **rv_op_bgez** = 294, **rv_op_bltz** = 295,
rv_op_bgtz = 296, **rv_op_ble** = 297, **rv_op_bleu** = 298, **rv_op_bgt** = 299,
rv_op_bgtu = 300, **rv_op_j** = 301, **rv_op_ret** = 302, **rv_op_jr** = 303,
rv_op_rdcycle = 304, **rv_op_rdtime** = 305, **rv_op_rdinstret** = 306, **rv_op_rdcycleh** = 307,
rv_op_rdtimeh = 308, **rv_op_rdinstreth** = 309, **rv_op_frcsr** = 310, **rv_op_frrm** = 311,

rv_op_frflags = 312, **rv_op_fscsr** = 313, **rv_op_fsrm** = 314, **rv_op_fsflags** = 315, **rv_op_fsrmi** = 316, **rv_op_fsflagsi** = 317, **rv_op_last** = 318 }

Variables

- const **riscvinstr** **instrlist** [ ]

7.410.1 Enumeration Type Documentation

7.410.1.1 rv_op

enum **rv_op**

Enumerator

rv_op_illegal	
rv_op_lui	
rv_op_auipc	
rv_op_jal	
rv_op_jalr	
rv_op_beq	
rv_op_bne	
rv_op_blt	
rv_op_bge	
rv_op_bltu	
rv_op_bgeu	
rv_op_lb	
rv_op_lh	
rv_op_lw	
rv_op_lbu	
rv_op_lhu	
rv_op_sb	
rv_op_sh	
rv_op_sw	
rv_op_addi	
rv_op_slti	
rv_op_sltiu	
rv_op_xori	
rv_op_ori	
rv_op_andi	
rv_op_slli	
rv_op_srli	
rv_op_srai	
rv_op_add	
rv_op_sub	
rv_op_sll	

Enumerator

rv_op_slt	
rv_op_sltu	
rv_op_xor	
rv_op_srl	
rv_op_sra	
rv_op_or	
rv_op_and	
rv_op_fence	
rv_op_fence_i	
rv_op_lwu	
rv_op_ld	
rv_op_sd	
rv_op_addiw	
rv_op_slliw	
rv_op_srliw	
rv_op_sraiw	
rv_op_addw	
rv_op_subw	
rv_op_sllw	
rv_op_srlw	
rv_op_sraw	
rv_op_ldu	
rv_op_lq	
rv_op_sq	
rv_op_addid	
rv_op_sllid	
rv_op_srlid	
rv_op_sraid	
rv_op_addd	
rv_op_subd	
rv_op_slld	
rv_op_srld	
rv_op_srad	
rv_op_mul	
rv_op_mulh	
rv_op_mulhsu	
rv_op_mulhu	
rv_op_div	
rv_op_divu	
rv_op_rem	
rv_op_remu	
rv_op_mulw	
rv_op_divw	
rv_op_divuw	
rv_op_remw	
rv_op_remuw	
rv_op_muld	
rv_op_divd	

Enumerator

rv_op_divud	
rv_op_remd	
rv_op_remud	
rv_op_lr_w	
rv_op_sc_w	
rv_op_amoswap_w	
rv_op_amoadd_w	
rv_op_amoxor_w	
rv_op_amoor_w	
rv_op_amoand_w	
rv_op_amomin_w	
rv_op_amomax_w	
rv_op_amominu_w	
rv_op_amomaxu_w	
rv_op_lr_d	
rv_op_sc_d	
rv_op_amoswap_d	
rv_op_amoadd_d	
rv_op_amoxor_d	
rv_op_amoor_d	
rv_op_amoand_d	
rv_op_amomin_d	
rv_op_amomax_d	
rv_op_amominu_d	
rv_op_amomaxu_d	
rv_op_lr_q	
rv_op_sc_q	
rv_op_amoswap_q	
rv_op_amoadd_q	
rv_op_amoxor_q	
rv_op_amoor_q	
rv_op_amoand_q	
rv_op_amomin_q	
rv_op_amomax_q	
rv_op_amominu_q	
rv_op_amomaxu_q	
rv_op_ecall	
rv_op_ebreak	
rv_op_uret	
rv_op_sret	
rv_op_hret	
rv_op_mret	
rv_op_dret	

Enumerator

rv_op_sfence_vm	
rv_op_wfi	
rv_op_csrrw	
rv_op_csrrs	
rv_op_csrrc	
rv_op_csrrwi	
rv_op_csrrsi	
rv_op_csrrci	
rv_op_flw	
rv_op_fsw	
rv_op_fmadd_s	
rv_op_fmsub_s	
rv_op_fnmsub_s	
rv_op_fnmadd_s	
rv_op_fadd_s	
rv_op_fsub_s	
rv_op_fmul_s	
rv_op_fdiv_s	
rv_op_fsgnj_s	
rv_op_fsgnjn_s	
rv_op_fsgnjx_s	
rv_op_fmin_s	
rv_op_fmax_s	
rv_op_fsqrt_s	
rv_op_fle_s	
rv_op_flt_s	
rv_op_feq_s	
rv_op_fcvt_w_s	
rv_op_fcvt_wu_s	
rv_op_fcvt_s_w	
rv_op_fcvt_s_wu	
rv_op_fmv_x_s	
rv_op_fclass_s	
rv_op_fmv_s_x	
rv_op_fcvt_l_s	
rv_op_fcvt_lu_s	
rv_op_fcvt_s_l	
rv_op_fcvt_s_lu	
rv_op_fld	
rv_op_fsd	
rv_op_fmadd_d	
rv_op_fmsub_d	
rv_op_fnmsub_d	
rv_op_fnmadd_d	
rv_op_fadd_d	
rv_op_fsub_d	
rv_op_fmul_d	

Enumerator

rv_op_fdiv_d	
rv_op_fsgnj_d	
rv_op_fsgnjn_d	
rv_op_fsgnjx_d	
rv_op_fmin_d	
rv_op_fmax_d	
rv_op_fcvt_s_d	
rv_op_fcvt_d_s	
rv_op_fsqrt_d	
rv_op_fle_d	
rv_op_flt_d	
rv_op_feq_d	
rv_op_fcvt_w_d	
rv_op_fcvt_wu_d	
rv_op_fcvt_d_w	
rv_op_fcvt_d_wu	
rv_op_fclass_d	
rv_op_fcvt_l_d	
rv_op_fcvt_lu_d	
rv_op_fmv_x_d	
rv_op_fcvt_d_l	
rv_op_fcvt_d_lu	
rv_op_fmv_d_x	
rv_op_flq	
rv_op_fsq	
rv_op_fmadd_q	
rv_op_fmsub_q	
rv_op_fnmsub_q	
rv_op_fnmadd_q	
rv_op_fadd_q	
rv_op_fsub_q	
rv_op_fmul_q	
rv_op_fdiv_q	
rv_op_fsgnj_q	
rv_op_fsgnjn_q	
rv_op_fsgnjx_q	
rv_op_fmin_q	
rv_op_fmax_q	
rv_op_fcvt_s_q	
rv_op_fcvt_q_s	
rv_op_fcvt_d_q	
rv_op_fcvt_q_d	
rv_op_fsqrt_q	
rv_op_fle_q	
rv_op_flt_q	
rv_op_feq_q	

Enumerator

rv_op_fcvt_w_q	
rv_op_fcvt_wu_q	
rv_op_fcvt_q_w	
rv_op_fcvt_q_wu	
rv_op_fclass_q	
rv_op_fcvt_l_q	
rv_op_fcvt_lu_q	
rv_op_fcvt_q_l	
rv_op_fcvt_q_lu	
rv_op_fmv_x_q	
rv_op_fmv_q_x	
rv_op_c_addi4spn	
rv_op_c_fld	
rv_op_c_lw	
rv_op_c_flw	
rv_op_c_fsd	
rv_op_c_sw	
rv_op_c_fsw	
rv_op_c_nop	
rv_op_c_addi	
rv_op_c_jal	
rv_op_c_li	
rv_op_c_addi16sp	
rv_op_c_lui	
rv_op_c_srli	
rv_op_c_srai	
rv_op_c_andi	
rv_op_c_sub	
rv_op_c_xor	
rv_op_c_or	
rv_op_c_and	
rv_op_c_subw	
rv_op_c_addw	
rv_op_c_j	
rv_op_c_beqz	
rv_op_c_bnez	
rv_op_c_slli	
rv_op_c_fldsp	
rv_op_c_lwsp	
rv_op_c_flwsp	
rv_op_c_jr	
rv_op_c_mv	
rv_op_c_ebreak	
rv_op_c_jalr	
rv_op_c_add	
rv_op_c_fsdsp	
rv_op_c_swsp	

Enumerator

rv_op_c_fswsp	
rv_op_c_ld	
rv_op_c_sd	
rv_op_c_addiw	
rv_op_c_ldsp	
rv_op_c_sdsp	
rv_op_c_lq	
rv_op_c_sq	
rv_op_c_lqsp	
rv_op_c_sqsp	
rv_op_nop	
rv_op_mv	
rv_op_not	
rv_op_neg	
rv_op_negw	
rv_op_sext_w	
rv_op_seqz	
rv_op_snez	
rv_op_sltz	
rv_op_sgtz	
rv_op_fmv_s	
rv_op_fabs_s	
rv_op_fneg_s	
rv_op_fmv_d	
rv_op_fabs_d	
rv_op_fneg_d	
rv_op_fmv_q	
rv_op_fabs_q	
rv_op_fneg_q	
rv_op_beqz	
rv_op_bnez	
rv_op_blez	
rv_op_bgez	
rv_op_bltz	
rv_op_bgtz	
rv_op_ble	
rv_op_bleu	
rv_op_bgt	
rv_op_bgtu	
rv_op_j	
rv_op_ret	
rv_op_jr	
rv_op_rdcycle	
rv_op_rdtime	
rv_op_rdinstret	
rv_op_rdcycleh	
rv_op_rdtimeh	

Enumerator

rv_op_rdinstreth	
rv_op_frcsr	
rv_op_frrm	
rv_op_frflags	
rv_op_fscsr	
rv_op_fsrm	
rv_op_fsflags	
rv_op_fsrmi	
rv_op_fsflagsi	
rv_op_last	

Definition at line 4 of file riscv_meta.h.

7.410.2 Variable Documentation

7.410.2.1 instrlist

```
const riscvinstr instrlist[]
```

Definition at line 338 of file riscv_meta.h.

Referenced by CoreModelBoomV1::CoreModelBoomV1(), and DynamicMicroOpBoomV1::getPort().

7.411 common/performance_model/performance_models/interval_↩ performance_model.cc File Reference

```
#include "interval_performance_model.h"
#include "config.hpp"
#include <cstdio>
```

7.412 common/performance_model/performance_models/interval_↩ performance_model.d File Reference

7.413 common/performance_model/performance_models/interval_↩ performance_model.h File Reference

```
#include "micro_op_performance_model.h"
```

Classes

- class **IntervalPerformanceModel**

7.414 common/performance_model/performance_models/interval_↩ performance_model/interval_contention.h File Reference

```
#include "fixed_types.h"
```

Classes

- class **IntervalContention**

7.415 common/performance_model/performance_models/interval_↩ performance_model/interval_contention_boom_v1.cc File Reference

```
#include "interval_contention_boom_v1.h"
#include "core.h"
#include "stats.h"
```

7.416 common/performance_model/performance_models/interval_↩ performance_model/interval_contention_boom_v1.d File Reference

7.417 common/performance_model/performance_models/interval_↩ performance_model/interval_contention_boom_v1.h File Reference

```
#include "interval_contention.h"
#include "dynamic_micro_op_boom_v1.h"
```

Classes

- class **IntervalContentionBoomV1**

7.418 common/performance_model/performance_models/interval_↩ performance_model/interval_contention_nehalem.cc File Reference

```
#include "interval_contention_nehalem.h"
#include "core.h"
#include "stats.h"
```

7.419 common/performance_model/performance_models/interval_↩ performance_model/interval_contention_nehalem.d File Reference

7.420 common/performance_model/performance_models/interval_↩ performance_model/interval_contention_nehalem.h File Reference

```
#include "interval_contention.h"
#include "dynamic_micro_op_nehalem.h"
```

Classes

- class **IntervalContentionNehalem**

7.421 common/performance_model/performance_models/interval_↩ performance_model/interval_timer.cc File Reference

```
#include "interval_timer.h"
#include "tools.h"
#include "core_model.h"
#include "stats.h"
#include "core_manager.h"
#include "itostr.h"
#include "performance_model.h"
#include "instruction.h"
#include "config.hpp"
#include "utils.h"
#include <algorithm>
#include <iostream>
#include <cstdio>
```

Functions

- String **StopDispatchReasonStringHelper** (**StopDispatchReason** r)
- String **StopDispatchReasonString** (**StopDispatchReason** r)

7.421.1 Function Documentation

7.421.1.1 StopDispatchReasonString()

```
String StopDispatchReasonString (
              StopDispatchReason r )
```

Definition at line 738 of file interval_timer.cc.

References STOP_DISPATCH_NO_REASON, STOP_DISPATCH_SIZE, and StopDispatchReasonStringHelper().

Referenced by IntervalTimer::IntervalTimer().

7.421.1.2 StopDispatchReasonStringHelper()

```
String StopDispatchReasonStringHelper (
              StopDispatchReason r )
```

Definition at line 717 of file interval_timer.cc.

References STOP_DISPATCH_BRANCH_MISPREDICT, STOP_DISPATCH_DISPATCH_RATE, STOP_DISPA↩TCH_DISPATCH_WIDTH, STOP_DISPATCH_ICACHE_MISS, STOP_DISPATCH_NO_REASON, and STOP_↩DISPATCH_WINDOW_EMPTY.

Referenced by StopDispatchReasonString().

7.422 common/performance_model/performance_models/interval_↩ performance_model/interval_timer.d File Reference

7.423 common/performance_model/performance_models/interval_↩ performance_model/interval_timer.h File Reference

```
#include <vector>
#include "fixed_point.h"
#include "windows.h"
#include "dynamic_micro_op.h"
#include "boost/tuple/tuple.hpp"
#include "core.h"
#include "contention_model.h"
```

Classes

- class **IntervalTimer**

Macros

- #define **DEBUG_IT_INSN_PRINT** 0
- #define **ADD_STOP_DISPATCH_REASON**(_new_reason, _original_reasons) ((**StopDispatch↩ Reason**)(((int)_new_reason) | (int)(_original_reasons)))

Enumerations

- enum **StopDispatchReason** {
  STOP_DISPATCH_NO_REASON = 0, **STOP_DISPATCH_WINDOW_EMPTY** = 1, **STOP_DISPATCH_↩ DISPATCH_WIDTH** = 2, **STOP_DISPATCH_DISPATCH_RATE** = 4,
  STOP_DISPATCH_ICACHE_MISS = 8, **STOP_DISPATCH_BRANCH_MISPREDICT** = 16, **STOP_DIS↩ PATCH_SIZE** = 32 }

Functions

- String **StopDispatchReasonStringHelper** (**StopDispatchReason** r)
- String **StopDispatchReasonString** (**StopDispatchReason** r)

7.423.1 Macro Definition Documentation

7.423.1.1 ADD_STOP_DISPATCH_REASON

```
#define ADD_STOP_DISPATCH_REASON(
              _new_reason,
              _original_reasons ) (( StopDispatchReason)(((int)_new_reason) | (int)(_original↩
_reasons)))
```

Definition at line 29 of file interval_timer.h.

7.423.1.2 DEBUG_IT_INSN_PRINT

```
#define DEBUG_IT_INSN_PRINT 0
```

Definition at line 17 of file interval_timer.h.

7.423.2 Enumeration Type Documentation

7.423.2.1 StopDispatchReason

```
enum StopDispatchReason
```

Enumerator

STOP_DISPATCH_NO_REASON	
STOP_DISPATCH_WINDOW_EMPTY	
STOP_DISPATCH_DISPATCH_WIDTH	
STOP_DISPATCH_DISPATCH_RATE	
STOP_DISPATCH_ICACHE_MISS	
STOP_DISPATCH_BRANCH_MISPREDICT	
STOP_DISPATCH_SIZE	

Definition at line 19 of file interval_timer.h.

7.423.3 Function Documentation

7.423.3.1 StopDispatchReasonString()

```
String StopDispatchReasonString (
              StopDispatchReason r )
```

Definition at line 738 of file interval_timer.cc.

References STOP_DISPATCH_NO_REASON, STOP_DISPATCH_SIZE, and StopDispatchReasonStringHelper().

Referenced by IntervalTimer::IntervalTimer().

7.423.3.2 StopDispatchReasonStringHelper()

```
String StopDispatchReasonStringHelper (
              StopDispatchReason r )
```

Definition at line 717 of file interval_timer.cc.

References STOP_DISPATCH_BRANCH_MISPREDICT, STOP_DISPATCH_DISPATCH_RATE, STOP_DISPA↩TCH_DISPATCH_WIDTH, STOP_DISPATCH_ICACHE_MISS, STOP_DISPATCH_NO_REASON, and STOP_↩DISPATCH_WINDOW_EMPTY.

Referenced by StopDispatchReasonString().

7.424 common/performance_model/performance_models/interval_↩ performance_model/tools.h File Reference

```
#include "micro_op.h"
#include <cstring>
```

Classes

- class **Tools**

7.425 common/performance_model/performance_models/interval_↩ performance_model/windows.cc File Reference

```
#include "windows.h"
#include "log.h"
#include "instruction.h"
#include "core_model.h"
#include "interval_contention.h"
#include <algorithm>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <assert.h>
```

Functions

- **CpContrType getCpContrType** (const **Windows::WindowEntry** &entry)
- String **CpContrTypeString** (**CpContrType** type)

7.425.1 Function Documentation

7.425.1.1 CpContrTypeString()

```
String CpContrTypeString (
            CpContrType type )
```

Definition at line 451 of file windows.cc.

References CPCONTR_TYPE_BRANCH, CPCONTR_TYPE_FP_ADDSUB, CPCONTR_TYPE_FP_MULDIV, C↩ PCONTR_TYPE_GENERIC, CPCONTR_TYPE_LOAD_L1, CPCONTR_TYPE_LOAD_L2, CPCONTR_TYPE_L↩ OAD_L3, CPCONTR_TYPE_LOAD_LX, CPCONTR_TYPE_STORE, and LOG_PRINT_ERROR.

Referenced by IntervalTimer::dispatchWindow(), and IntervalTimer::IntervalTimer().

7.425.1.2 getCpContrType()

```
CpContrType getCpContrType (
            const  Windows::WindowEntry & entry )
```

Definition at line 417 of file windows.cc.

References CPCONTR_TYPE_BRANCH, CPCONTR_TYPE_FP_ADDSUB, CPCONTR_TYPE_FP_MULDIV, CPCONTR_TYPE_GENERIC, CPCONTR_TYPE_LOAD_L1, CPCONTR_TYPE_LOAD_L2, CPCONTR_TYPE_LOAD_L3, CPCONTR_TYPE_LOAD_LX, CPCONTR_TYPE_STORE, DynamicMicroOp::getDCacheHitWhere(), Windows::WindowEntry::getDynMicroOp(), Windows::WindowEntry::getMicroOp(), MicroOp::getSubtype(), HitWhere::L1_OWN, HitWhere::L1_SIBLING, HitWhere::L2_OWN, HitWhere::L2_SIBLING, HitWhere::L3_OWN, HitWhere::L3_SIBLING, LOG_PRINT_ERROR, MicroOp::UOP_SUBTYPE_BRANCH, MicroOp::UOP_SUBTYPE_FP_ADDSUB, MicroOp::UOP_SUBTYPE_FP_MULDIV, MicroOp::UOP_SUBTYPE_GENERIC, MicroOp::UOP_SUBTYPE_LOAD, and MicroOp::UOP_SUBTYPE_STORE.

Referenced by Windows::addFunctionalUnitStats(), and Windows::removeFunctionalUnitStats().

7.426 common/performance_model/performance_models/interval_performance_model/windows.d File Reference

7.427 common/performance_model/performance_models/interval_performance_model/windows.h File Reference

```
#include "micro_op.h"
#include "register_dependencies.h"
#include "memory_dependencies.h"
```

Classes

- class **Windows**
- struct **Windows::WindowEntry**
- class **Windows::Iterator**

Enumerations

- enum **CpContrType** {
  CPCONTR_TYPE_FP_ADDSUB, **CPCONTR_TYPE_FP_MULDIV**, **CPCONTR_TYPE_LOAD_L1**, **CPCONTR_TYPE_LOAD_L2**,
  CPCONTR_TYPE_LOAD_L3, **CPCONTR_TYPE_LOAD_LX**, **CPCONTR_TYPE_STORE**, **CPCONTR_TYPE_GENERIC**,
  CPCONTR_TYPE_BRANCH, **CPCONTR_TYPE_SIZE** }

Functions

- **CpContrType getCpContrType** (const **Windows::WindowEntry** &uop)
- String **CpContrTypeString** (**CpContrType** r)

7.427.1 Enumeration Type Documentation

7.427.1.1 CpContrType

enum **CpContrType**

Enumerator

CPCONTR_TYPE_FP_ADDSUB	
CPCONTR_TYPE_FP_MULDIV	
CPCONTR_TYPE_LOAD_L1	
CPCONTR_TYPE_LOAD_L2	
CPCONTR_TYPE_LOAD_L3	
CPCONTR_TYPE_LOAD_LX	
CPCONTR_TYPE_STORE	
CPCONTR_TYPE_GENERIC	
CPCONTR_TYPE_BRANCH	
CPCONTR_TYPE_SIZE	

Definition at line 15 of file windows.h.

7.427.2 Function Documentation

7.427.2.1 CpContrTypeString()

```
String CpContrTypeString (
            CpContrType r )
```

Definition at line 451 of file windows.cc.

References CPCONTR_TYPE_BRANCH, CPCONTR_TYPE_FP_ADDSUB, CPCONTR_TYPE_FP_MULDIV, C↩
PCONTR_TYPE_GENERIC, CPCONTR_TYPE_LOAD_L1, CPCONTR_TYPE_LOAD_L2, CPCONTR_TYPE_L↩
OAD_L3, CPCONTR_TYPE_LOAD_LX, CPCONTR_TYPE_STORE, and LOG_PRINT_ERROR.

Referenced by IntervalTimer::dispatchWindow(), and IntervalTimer::IntervalTimer().

7.427.2.2 getCpContrType()

```
CpContrType getCpContrType (
            const  Windows::WindowEntry & uop )
```

Definition at line 417 of file windows.cc.

References CPCONTR_TYPE_BRANCH, CPCONTR_TYPE_FP_ADDSUB, CPCONTR_TYPE_FP_MULDIV, CPCONTR_TYPE_GENERIC, CPCONTR_TYPE_LOAD_L1, CPCONTR_TYPE_LOAD_L2, CPCONTR_TYPE_LOAD_L3, CPCONTR_TYPE_LOAD_LX, CPCONTR_TYPE_STORE, DynamicMicroOp::getDCacheHitWhere(), Windows::WindowEntry::getDynMicroOp(), Windows::WindowEntry::getMicroOp(), MicroOp::getSubtype(), HitWhere::L1_OWN, HitWhere::L1_SIBLING, HitWhere::L2_OWN, HitWhere::L2_SIBLING, HitWhere::L3_OWN, HitWhere::L3_SIBLING, LOG_PRINT_ERROR, MicroOp::UOP_SUBTYPE_BRANCH, MicroOp::UOP_SUBTYPE_FP_ADDSUB, MicroOp::UOP_SUBTYPE_FP_MULDIV, MicroOp::UOP_SUBTYPE_GENERIC, MicroOp::UOP_SUBTYPE_LOAD, and MicroOp::UOP_SUBTYPE_STORE.

Referenced by Windows::addFunctionalUnitStats(), and Windows::removeFunctionalUnitStats().

7.428 common/performance_model/performance_models/micro_op/dynamic_micro_op.cc File Reference

```
#include "dynamic_micro_op.h"
#include "log.h"
#include "core_model.h"
```

7.429 common/performance_model/performance_models/micro_op/dynamic_micro_op.d File Reference

7.430 common/performance_model/performance_models/micro_op/dynamic_micro_op.h File Reference

```
#include "fixed_types.h"
#include "allocator.h"
#include "subsecond_time.h"
#include "memory_access.h"
#include "micro_op.h"
#include "hit_where.h"
```

Classes

- class **DynamicMicroOp**

7.431 common/performance_model/performance_models/micro_↩ op/instruction_decoder.cc File Reference

```
#include "instruction_decoder.h"
#include "instruction.h"
#include "micro_op.h"
#include <xed-reg-class.h>
```

7.432 common/performance_model/performance_models/micro_↩ op/instruction_decoder.d File Reference

7.433 common/performance_model/performance_models/micro_↩ op/instruction_decoder.h File Reference

```
#include "fixed_types.h"
#include <xed-decoded-inst.h>
#include <vector>
#include <set>
```

Classes

- class **InstructionDecoder**

7.434 common/performance_model/performance_models/micro_↩ op/instruction_decoder_wlib.cc File Reference

```
#include "instruction_decoder_wlib.h"
#include "instruction.h"
#include "micro_op.h"
#include "simulator.h"
#include <x86_decoder.h>
```

7.435 common/performance_model/performance_models/micro_↩ op/instruction_decoder_wlib.d File Reference

7.436 common/performance_model/performance_models/micro_↩ op/instruction_decoder_wlib.h File Reference

```
#include "fixed_types.h"
#include <decoder.h>
#include <vector>
#include <set>
```

Classes

- class **InstructionDecoder**

7.437 common/performance_model/performance_models/micro_↩ op/memory_access.h File Reference

Classes

- struct **Memory::Access**

Namespaces

- **Memory**

Functions

- Access **Memory::make_access** (**UInt64** address)

7.438 common/performance_model/performance_models/micro_↩ op/memory_dependencies.cc File Reference

```
#include "memory_dependencies.h"
```

7.439 common/performance_model/performance_models/micro_↩ op/memory_dependencies.d File Reference

7.440 common/performance_model/performance_models/micro_↩ op/memory_dependencies.h File Reference

```
#include "fixed_types.h"
#include "circular_queue.h"
#include "dynamic_micro_op.h"
```

Classes

- class **MemoryDependencies**
- struct **MemoryDependencies::Producer**

7.441 common/performance_model/performance_models/micro_↩ op/micro_op.cc File Reference

```
#include "micro_op.h"
#include "instruction.h"
#include <assert.h>
#include <iostream>
#include <sstream>
#include <iomanip>
```

Macros

- #define **VERIFY_MICROOP**() do {} while (0)

7.441.1 Macro Definition Documentation

7.441.1.1 VERIFY_MICROOP

```
#define VERIFY_MICROOP( ) do {} while (0)
```

Definition at line 20 of file micro_op.cc.

7.442 common/performance_model/performance_models/micro_↩ op/micro_op.d File Reference

7.443 common/performance_model/performance_models/micro_↩ op/micro_op.h File Reference

```
#include "fixed_types.h"
#include "memory_access.h"
#include "tools.h"
#include "subsecond_time.h"
#include "log.h"
#include "simulator.h"
#include <decoder.h>
#include <vector>
```

Classes

- struct **MicroOp**

Macros

- #define **MAX_SRC_REGS** 3
- #define **MAX_MEM_SRC_REGS** 2
- #define **MAX_DEST_REGS** 3
- #define **MAX_MEM_DEST_REGS** 2
- #define **MAX_CF_REGS** 11
- #define **MAXIMUM_NUMBER_OF_SOURCE_REGISTERS** (**MAX_SRC_REGS** + **MAX_CF_REGS**)
- #define **MAXIMUM_NUMBER_OF_ADDRESS_REGISTERS** **MAX_MEM_SRC_REGS**
- #define **MAXIMUM_NUMBER_OF_DESTINATION_REGISTERS** (**MAX_DEST_REGS** + **MAX_CF_RE↩ GS**)
- #define **MAXIMUM_NUMBER_OF_DEPENDENCIES** **MAXIMUM_NUMBER_OF_SOURCE_REGISTERS**

Variables

- const uint64_t **INVALID_SEQNR** = UINT64_MAX

7.443.1 Macro Definition Documentation

7.443.1.1 MAX_CF_REGS

```
#define MAX_CF_REGS 11
```

Definition at line 26 of file micro_op.h.

7.443.1.2 MAX_DEST_REGS

```
#define MAX_DEST_REGS 3
```

Definition at line 24 of file micro_op.h.

7.443.1.3 MAX_MEM_DEST_REGS

```
#define MAX_MEM_DEST_REGS 2
```

Definition at line 25 of file micro_op.h.

7.443.1.4 MAX_MEM_SRC_REGS

```
#define MAX_MEM_SRC_REGS 2
```

Definition at line 23 of file micro_op.h.

7.443.1.5 MAX_SRC_REGS

```
#define MAX_SRC_REGS 3
```

Definition at line 22 of file micro_op.h.

7.443.1.6 MAXIMUM_NUMBER_OF_ADDRESS_REGISTERS

```
#define MAXIMUM_NUMBER_OF_ADDRESS_REGISTERS  MAX_MEM_SRC_REGS
```

Definition at line 29 of file micro_op.h.

7.443.1.7 MAXIMUM_NUMBER_OF_DEPENDENCIES

```
#define MAXIMUM_NUMBER_OF_DEPENDENCIES  MAXIMUM_NUMBER_OF_SOURCE_REGISTERS
```

Definition at line 31 of file micro_op.h.

7.443.1.8 MAXIMUM_NUMBER_OF_DESTINATION_REGISTERS

```
#define MAXIMUM_NUMBER_OF_DESTINATION_REGISTERS ( MAX_DEST_REGS +  MAX_CF_REGS)
```

Definition at line 30 of file micro_op.h.

7.443.1.9 MAXIMUM_NUMBER_OF_SOURCE_REGISTERS

```
#define MAXIMUM_NUMBER_OF_SOURCE_REGISTERS ( MAX_SRC_REGS +  MAX_CF_REGS)
```

Definition at line 28 of file micro_op.h.

7.443.2 Variable Documentation

7.443.2.1 INVALID_SEQNR

```
const uint64_t INVALID_SEQNR = UINT64_MAX
```

Definition at line 33 of file micro_op.h.

Referenced by RegisterDependencies::clear(), MemoryDependencies::clear(), DynamicMicroOp::Dynamic↩ MicroOp(), MemoryDependencies::find(), RegisterDependencies::peekProducer(), RegisterDependencies::set↩ Dependencies(), MemoryDependencies::setDependencies(), RobSmtTimer::setStoreAddressProducers(), and RobTimer::simulate().

7.444 common/performance_model/performance_models/micro_↩ op/register_dependencies.cc File Reference

```
#include "register_dependencies.h"
#include "dynamic_micro_op.h"
```

7.445 common/performance_model/performance_models/micro_op/register_dependencies.d File Reference

7.446 common/performance_model/performance_models/micro_op/register_dependencies.h File Reference

```
#include "fixed_types.h"
#include <decoder.h>
```

Classes

- class **RegisterDependencies**

7.447 common/performance_model/performance_models/micro_op_performance_model.cc File Reference

```
#include "core.h"
#include "log.h"
#include "core_model.h"
#include "micro_op_performance_model.h"
#include "branch_predictor.h"
#include "stats.h"
#include "dvfs_manager.h"
#include "subsecond_time.h"
#include "micro_op.h"
#include "allocator.h"
#include "config.hpp"
#include "dynamic_instruction.h"
#include <cstdio>
#include <algorithm>
```

7.448 common/performance_model/performance_models/micro_op_performance_model.d File Reference

7.449 common/performance_model/performance_models/micro_op_performance_model.h File Reference

```
#include "performance_model.h"
#include "instruction.h"
```

```
#include "interval_timer.h"
#include "stats.h"
#include "subsecond_time.h"
#include "dynamic_micro_op.h"
```

Classes

- class **MicroOpPerformanceModel**

Macros

- #define **DEBUG_INSN_LOG** 0
- #define **DEBUG_DYN_INSN_LOG** 0
- #define **DEBUG_CYCLE_COUNT_LOG** 0

7.449.1 Macro Definition Documentation

7.449.1.1 DEBUG_CYCLE_COUNT_LOG

```
#define DEBUG_CYCLE_COUNT_LOG 0
```

Definition at line 13 of file micro_op_performance_model.h.

7.449.1.2 DEBUG_DYN_INSN_LOG

```
#define DEBUG_DYN_INSN_LOG 0
```

Definition at line 12 of file micro_op_performance_model.h.

7.449.1.3 DEBUG_INSN_LOG

```
#define DEBUG_INSN_LOG 0
```

Definition at line 11 of file micro_op_performance_model.h.

7.450 common/performance_model/performance_models/oneipc_↩ performance_model.cc File Reference

```
#include "oneipc_performance_model.h"
#include "simulator.h"
#include "core.h"
#include "log.h"
#include "config.hpp"
#include "branch_predictor.h"
#include "stats.h"
#include "dvfs_manager.h"
#include "subsecond_time.h"
#include "instruction.h"
#include "dynamic_instruction.h"
```

7.451 common/performance_model/performance_models/oneipc_↩ performance_model.d File Reference

7.452 common/performance_model/performance_models/oneipc_↩ performance_model.h File Reference

```
#include "performance_model.h"
```

Classes

- class **OneIPCPerformanceModel**

7.453 common/performance_model/performance_models/rob_↩ performance_model.cc File Reference

```
#include "rob_performance_model.h"
#include "config.hpp"
```

7.454 common/performance_model/performance_models/rob_↩ performance_model.d File Reference

7.455 common/performance_model/performance_models/rob_↩ performance_model.h File Reference

```
#include "micro_op_performance_model.h"
#include "rob_timer.h"
```

Classes

- class **RobPerformanceModel**

7.456 common/performance_model/performance_models/rob_↩ performance_model/rob_contention.h File Reference

```
#include "micro_op.h"
```

Classes

- class **RobContention**

7.457 common/performance_model/performance_models/rob_↩ performance_model/rob_contention_boom_v1.cc File Reference

```
#include "rob_contention_boom_v1.h"
#include "core_model.h"
#include "dynamic_micro_op.h"
#include "core.h"
#include "config.hpp"
#include "simulator.h"
#include "memory_manager_base.h"
```

7.458 common/performance_model/performance_models/rob_performance_model/rob_contention_boom_v1.d File Reference

7.459 common/performance_model/performance_models/rob_performance_model/rob_contention_boom_v1.h File Reference

```
#include "rob_contention.h"
#include "contention_model.h"
#include "core_model_boom_v1.h"
#include "dynamic_micro_op_boom_v1.h"
#include <vector>
```

Classes

- class **RobContentionBoomV1**

7.460 common/performance_model/performance_models/rob_performance_model/rob_contention_nehalem.cc File Reference

```
#include "rob_contention_nehalem.h"
#include "core_model.h"
#include "dynamic_micro_op.h"
#include "core.h"
#include "config.hpp"
#include "simulator.h"
#include "memory_manager_base.h"
```

7.461 common/performance_model/performance_models/rob_performance_model/rob_contention_nehalem.d File Reference

7.462 common/performance_model/performance_models/rob_performance_model/rob_contention_nehalem.h File Reference

```
#include "rob_contention.h"
#include "contention_model.h"
#include "core_model_nehalem.h"
#include "dynamic_micro_op_nehalem.h"
#include <vector>
```

Classes

- class **RobContentionNehalem**

7.463 common/performance_model/performance_models/rob_performance_model/rob_smt_timer.cc File Reference

```
#include "rob_smt_timer.h"
#include "tools.h"
#include "stats.h"
#include "config.hpp"
#include "core_manager.h"
#include "thread.h"
#include "thread_manager.h"
#include "itostr.h"
#include "performance_model.h"
#include "core_model.h"
#include "rob_contention.h"
#include "instruction.h"
#include <iostream>
#include <sstream>
#include <iomanip>
```

7.464 common/performance_model/performance_models/rob_performance_model/rob_smt_timer.d File Reference

7.465 common/performance_model/performance_models/rob_performance_model/rob_smt_timer.h File Reference

```
#include "interval_timer.h"
#include "smt_timer.h"
#include "rob_contention.h"
#include <deque>
```

Classes

- class **RobSmtTimer**
- class **RobSmtTimer::RobEntry**
- class **RobSmtTimer::RobThread**

7.466 common/performance_model/performance_models/rob_performance_model/rob_timer.cc File Reference

```
#include "rob_timer.h"
#include "tools.h"
#include "stats.h"
#include "config.hpp"
#include "core_manager.h"
#include "itostr.h"
#include "performance_model.h"
#include "core_model.h"
#include "rob_contention.h"
#include "instruction.h"
#include <iostream>
#include <sstream>
#include <iomanip>
```

7.467 common/performance_model/performance_models/rob_performance_model/rob_timer.d File Reference

7.468 common/performance_model/performance_models/rob_performance_model/rob_timer.h File Reference

```
#include "interval_timer.h"
#include "rob_contention.h"
#include "stats.h"
#include <deque>
```

Classes

- class **RobTimer**
- class **RobTimer::RobEntry**

7.469 common/performance_model/performance_models/rob_performance_model/smt_timer.cc File Reference

```
#include "smt_timer.h"
#include "simulator.h"
#include "thread.h"
#include "circular_log.h"
```

7.470 common/performance_model/performance_models/rob_performance_model/smt_timer.d File Reference

7.471 common/performance_model/performance_models/rob_performance_model/smt_timer.h File Reference

```
#include "hooks_manager.h"
#include "cond.h"
#include "boost/tuple/tuple.hpp"
```

Classes

- class **SmtTimer**
- class **SmtTimer::SmtThread**

Macros

- #define **INVALID_SMTTHREAD_ID** ((smtthread_id_t) -1)

7.471.1 Macro Definition Documentation

7.471.1.1 INVALID_SMTTHREAD_ID

```
#define INVALID_SMTTHREAD_ID ((smtthread_id_t) -1)
```

Definition at line 35 of file smt_timer.h.

7.472 common/performance_model/performance_models/rob_smt_performance_model.cc File Reference

```
#include "rob_smt_performance_model.h"
#include "config.hpp"
```

7.473 common/performance_model/performance_models/rob_smt_↩ performance_model.d File Reference

7.474 common/performance_model/performance_models/rob_smt_↩ performance_model.h File Reference

```
#include "micro_op_performance_model.h"
#include "rob_smt_timer.h"
#include <unordered_map>
```

Classes

- class **RobSmtPerformanceModel**

7.475 common/performance_model/queue_model.cc File Reference

```
#include "queue_model.h"
#include "simulator.h"
#include "config.h"
#include "queue_model_basic.h"
#include "queue_model_history_list.h"
#include "queue_model_contention.h"
#include "queue_model_windowed_mg1.h"
#include "log.h"
#include "config.hpp"
```

7.476 common/performance_model/queue_model.d File Reference

7.477 common/performance_model/queue_model.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include <iostream>
```

Classes

- class **QueueModel**

7.478 common/performance_model/queue_model_basic.cc File Reference

```
#include "queue_model_basic.h"
#include "utils.h"
#include "log.h"
```

7.479 common/performance_model/queue_model_basic.d File Reference

7.480 common/performance_model/queue_model_basic.h File Reference

```
#include "queue_model.h"
#include "moving_average.h"
#include "fixed_types.h"
#include "subsecond_time.h"
```

Classes

- class **QueueModelBasic**

7.481 common/performance_model/queue_model_contention.cc File Reference

```
#include "queue_model_contention.h"
```

7.482 common/performance_model/queue_model_contention.d File Reference

7.483 common/performance_model/queue_model_contention.h File Reference

```
#include "queue_model.h"
#include "fixed_types.h"
#include "contention_model.h"
```

Classes

- class **QueueModelContention**

7.484 common/performance_model/queue_model_history_list.cc File Reference

```
#include "queue_model_history_list.h"
#include "simulator.h"
#include "core_manager.h"
#include "config.h"
#include "fxsupport.h"
#include "log.h"
#include "stats.h"
#include "config.hpp"
```

7.485 common/performance_model/queue_model_history_list.d File Reference

7.486 common/performance_model/queue_model_history_list.h File Reference

```
#include <list>
#include "queue_model.h"
#include "fixed_types.h"
#include "moving_average.h"
```

Classes

- class **QueueModelHistoryList**

7.487 common/performance_model/queue_model_windowed_mg1.cc File Reference

```
#include "queue_model_windowed_mg1.h"
#include "simulator.h"
#include "config.hpp"
#include "log.h"
#include "stats.h"
```

7.488 common/performance_model/queue_model_windowed_mg1.d File Reference

7.489 common/performance_model/queue_model_windowed_mg1.h File Reference

```
#include "queue_model.h"
#include "fixed_types.h"
#include "contention_model.h"
#include <map>
```

Classes

- class **QueueModelWindowedMG1**

7.490 common/performance_model/shmem_perf_model.cc File Reference

```
#include "fixed_types.h"
#include "shmem_perf_model.h"
#include "simulator.h"
#include "core_manager.h"
#include "fxsupport.h"
#include "subsecond_time.h"
```

7.491 common/performance_model/shmem_perf_model.d File Reference

7.492 common/performance_model/shmem_perf_model.h File Reference

```
#include <cassert>
#include <iostream>
#include "lock.h"
#include "subsecond_time.h"
```

Classes

- class **ShmemPerfModel**

7.493 common/sampling/instr_count_sampling.h File Reference

```
#include "sampling_provider.h"
```

Classes

- class **InstrCountSampling**

7.494 common/sampling/periodic_sampling.cc File Reference

```
#include "periodic_sampling.h"
#include "sampling_manager.h"
#include "simulator.h"
#include "core_manager.h"
#include "performance_model.h"
#include "fastforward_performance_model.h"
#include "config.hpp"
#include "average.h"
```

7.495 common/sampling/periodic_sampling.d File Reference

7.496 common/sampling/periodic_sampling.h File Reference

```
#include "fixed_types.h"
#include "sampling_algorithm.h"
#include "circular_queue.h"
#include "random.h"
#include <vector>
```

Classes

- class **PeriodicSampling**

7.497 common/sampling/sampling_algorithm.cc File Reference

```
#include "sampling_algorithm.h"
#include "simulator.h"
#include "config.hpp"
#include "log.h"
#include "periodic_sampling.h"
```

7.498 common/sampling/sampling_algorithm.d File Reference

7.499 common/sampling/sampling_algorithm.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
```

Classes

- class **SamplingAlgorithm**

7.500 common/sampling/sampling_manager.cc File Reference

```
#include "sampling_manager.h"
#include "hooks_manager.h"
#include "magic_server.h"
#include "simulator.h"
#include "core.h"
#include "core_manager.h"
#include "thread_manager.h"
#include "performance_model.h"
#include "fastforward_performance_model.h"
#include "config.hpp"
#include "magic_client.h"
#include "sampling_provider.h"
```

7.501 common/sampling/sampling_manager.d File Reference

7.502 common/sampling/sampling_manager.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include "sampling_algorithm.h"
#include "inst_mode.h"
#include <vector>
#include <boost/tuple/tuple.hpp>
```

Classes

- class **SamplingManager**

7.503 common/sampling/sampling_provider.cc File Reference

```
#include "sampling_provider.h"
#include "simulator.h"
#include "config.hpp"
#include "log.h"
#include "instr_count_sampling.h"
```

7.504 common/sampling/sampling_provider.d File Reference

7.505 common/sampling/sampling_provider.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
```

Classes

- class **SamplingProvider**

Namespaces

- **InstrumentLevel**

Enumerations

- enum **InstrumentLevel::Level** { **InstrumentLevel::INSTR_WITH_BBVS**, **InstrumentLevel::INSTR**, **InstrumentLevel::NONE** }

7.506 common/scheduler/scheduler.cc File Reference

```
#include "scheduler.h"
#include "scheduler_static.h"
#include "scheduler_pinned.h"
#include "scheduler_roaming.h"
#include "scheduler_big_small.h"
#include "scheduler_sequential.h"
#include "simulator.h"
#include "config.hpp"
#include "core_manager.h"
#include "thread_manager.h"
#include "thread.h"
```

7.507 common/scheduler/scheduler.d File Reference

7.508 common/scheduler/scheduler.h File Reference

```
#include "fixed_types.h"
#include "thread_manager.h"
```

Classes

- class **Scheduler**

7.509 common/scheduler/scheduler_big_small.cc File Reference

```
#include "scheduler_big_small.h"
#include "simulator.h"
#include "config.hpp"
#include "thread.h"
#include "performance_model.h"
#include "core_manager.h"
#include "misc/tags.h"
#include "rng.h"
```

7.510 common/scheduler/scheduler_big_small.d File Reference

7.511 common/scheduler/scheduler_big_small.h File Reference

```
#include "scheduler_pinned_base.h"
#include <unordered_map>
```

Classes

- class **SchedulerBigSmall**

7.512 common/scheduler/scheduler_dynamic.cc File Reference

```
#include "scheduler_dynamic.h"
#include "simulator.h"
#include "thread_stats_manager.h"
#include "hooks_manager.h"
```

7.513 common/scheduler/scheduler_dynamic.d File Reference

7.514 common/scheduler/scheduler_dynamic.h File Reference

```
#include "scheduler.h"
#include "hooks_manager.h"
#include <vector>
```

Classes

- class **SchedulerDynamic**

7.515 common/scheduler/scheduler_pinned.cc File Reference

```
#include "scheduler_pinned.h"
#include "config.hpp"
```

7.516 common/scheduler/scheduler_pinned.d File Reference

7.517 common/scheduler/scheduler_pinned.h File Reference

```
#include "scheduler_pinned_base.h"
```

Classes

- class **SchedulerPinned**

7.518 common/scheduler/scheduler_pinned_base.cc File Reference

```
#include "scheduler_pinned_base.h"
#include "simulator.h"
#include "core_manager.h"
#include "performance_model.h"
#include "os_compat.h"
#include <sstream>
```

7.519 common/scheduler/scheduler_pinned_base.d File Reference

7.520 common/scheduler/scheduler_pinned_base.h File Reference

```
#include "scheduler_dynamic.h"
#include "simulator.h"
```

Classes

- class **SchedulerPinnedBase**
- class **SchedulerPinnedBase::ThreadInfo**

7.521 common/scheduler/scheduler_roaming.cc File Reference

```
#include "scheduler_roaming.h"
#include "config.hpp"
```

7.522 common/scheduler/scheduler_roaming.d File Reference

7.523 common/scheduler/scheduler_roaming.h File Reference

```
#include "scheduler_pinned.h"
```

Classes

- class **SchedulerRoaming**

7.524 common/scheduler/scheduler_sequential.cc File Reference

```
#include "scheduler_sequential.h"
#include "config.hpp"
#include <thread.h>
#include <core_manager.h>
#include <dvfs_manager.h>
#include <trace_manager.h>
#include <unistd.h>
#include <iostream>
#include <fstream>
#include <boost/algorithm/string.hpp>
#include "performance_model.h"
#include "stats.h"
```

7.525 common/scheduler/scheduler_sequential.d File Reference

7.526 common/scheduler/scheduler_sequential.h File Reference

```
#include "scheduler_pinned_base.h"
#include "simulator.h"
#include "thread_stats_manager.h"
```

Classes

- class **SchedulerSequential**

Macros

- #define **MAIN_CORE** 0

7.526.1 Macro Definition Documentation

7.526.1.1 MAIN_CORE

```
#define MAIN_CORE 0
```

Definition at line 9 of file scheduler_sequential.h.

7.527 common/scheduler/scheduler_static.cc File Reference

```
#include "scheduler_static.h"
#include "core_manager.h"
#include "simulator.h"
#include "config.hpp"
#include "thread.h"
#include "log.h"
```

7.528 common/scheduler/scheduler_static.d File Reference

7.529 common/scheduler/scheduler_static.h File Reference

```
#include "scheduler.h"
#include <vector>
```

Classes

- class **SchedulerStatic**

7.530 common/scripting/hooks_py.cc File Reference

```
#include "hooks_py.h"
#include "simulator.h"
#include "core_manager.h"
#include "config.hpp"
#include "fxsupport.h"
```

7.531 common/scripting/hooks_py.d File Reference

7.532 common/scripting/hooks_py.h File Reference

```
#include "fixed_types.h"
#include <Python.h>
```

Classes

- class **HooksPy**
- class **HooksPy::PyConfig**
- class **HooksPy::PyStats**
- class **HooksPy::PyHooks**
- class **HooksPy::PyDvfs**
- class **HooksPy::PyControl**
- class **HooksPy::PyBbv**
- class **HooksPy::PyMem**
- class **HooksPy::PyThread**

7.533 common/scripting/py_bbv.cc File Reference

```
#include "hooks_py.h"
#include "simulator.h"
#include "core_manager.h"
#include "core.h"
```

Functions

- static PyObject * **enableBbv** (PyObject *self, PyObject *args)
- static PyObject * **disableBbv** (PyObject *self, PyObject *args)
- static PyObject * **getBbv** (PyObject *self, PyObject *args)

Variables

- static PyMethodDef **PyBbvMethods** [ ]

7.533.1 Function Documentation

7.533.1.1 disableBbv()

```
static PyObject* disableBbv (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 18 of file py_bbv.cc.

7.533.1.2 enableBbv()

```
static PyObject* enableBbv (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 11 of file py_bbv.cc.

7.533.1.3 getBbv()

```
static PyObject* getBbv (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 25 of file py_bbv.cc.

References BbvCount::getDimension(), BbvCount::getInstructionCount(), and BbvCount::NUM_BBV.

7.533.2 Variable Documentation

7.533.2.1 PyBbvMethods

```
PyMethodDef PyBbvMethods[]  [static]
```

Initial value:

```
= {
   {"enable", enableBbv, METH_VARARGS, "Enable BBV collection."},
   {"disable", disableBbv, METH_VARARGS, "Enable BBV collection."},
   {"get",  getBbv, METH_VARARGS, "Retrieve cummulative BBV for core."},
   {NULL, NULL, 0, NULL}
}
```

Definition at line 56 of file py_bbv.cc.

Referenced by HooksPy::PyBbv::setup().

7.534 common/scripting/py_bbv.d File Reference

7.535 common/scripting/py_config.cc File Reference

```
#include "hooks_py.h"
#include "simulator.h"
#include "config.hpp"
```

Functions

- static PyObject * **getConfigString** (PyObject *self, PyObject *args)
- static PyObject * **getConfigInt** (PyObject *self, PyObject *args)
- static PyObject * **getConfigFloat** (PyObject *self, PyObject *args)
- static PyObject * **getConfigBool** (PyObject *self, PyObject *args)

Variables

- static PyMethodDef **PyConfigMethods** [ ]

7.535.1 Function Documentation

7.535.1.1 getConfigBool()

```
static PyObject* getConfigBool (
             PyObject * self,
             PyObject * args )  [static]
```

Definition at line 60 of file py_config.cc.

7.535.1.2 getConfigFloat()

```
static PyObject* getConfigFloat (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 42 of file py_config.cc.

7.535.1.3 getConfigInt()

```
static PyObject* getConfigInt (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 24 of file py_config.cc.

7.535.1.4 getConfigString()

```
static PyObject* getConfigString (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 6 of file py_config.cc.

7.535.2 Variable Documentation

7.535.2.1 PyConfigMethods

```
PyMethodDef PyConfigMethods[]  [static]
```

Initial value:

```
= {
   {"get",  getConfigString, METH_VARARGS, "Get configuration variable."},
   {"get_int",  getConfigInt, METH_VARARGS, "Get configuration variable."},
   {"get_float",  getConfigFloat, METH_VARARGS, "Get configuration variable."},
   {"get_bool",  getConfigBool, METH_VARARGS, "Get configuration variable."},
   {NULL, NULL, 0, NULL}
}
```

Definition at line 78 of file py_config.cc.

Referenced by HooksPy::PyConfig::setup().

7.536 common/scripting/py_config.d File Reference

7.537 common/scripting/py_control.cc File Reference

```
#include "hooks_py.h"
#include "simulator.h"
#include "magic_server.h"
#include "sim_api.h"
```

Functions

- static PyObject * **setROI** (PyObject *self, PyObject *args)
- static PyObject * **setInstrumentationMode** (PyObject *self, PyObject *args)
- static PyObject * **setProgress** (PyObject *self, PyObject *args)
- static PyObject * **simulatorAbort** (PyObject *self, PyObject *args)

Variables

- static PyMethodDef **PyControlMethods** [ ]

7.537.1 Function Documentation

7.537.1.1 setInstrumentationMode()

```
static PyObject* setInstrumentationMode (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 20 of file py_control.cc.

References INVALID_CORE_ID, INVALID_THREAD_ID, and LOG_PRINT_ERROR.

7.537.1.2 setProgress()

```
static PyObject* setProgress (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 43 of file py_control.cc.

7.537.1.3 setROI()

```
static PyObject* setROI (
             PyObject * self,
             PyObject * args )  [static]
```

Definition at line 7 of file py_control.cc.

7.537.1.4 simulatorAbort()

```
static PyObject* simulatorAbort (
             PyObject * self,
             PyObject * args )  [static]
```

Definition at line 56 of file py_control.cc.

References LOG_PRINT, and Simulator::release().

7.537.2 Variable Documentation

7.537.2.1 PyControlMethods

```
PyMethodDef PyControlMethods[]  [static]
```

Initial value:

```
= {
   { "set_roi", setROI, METH_VARARGS, "Set whether or not we are in the ROI" },
   { "set_instrumentation_mode", setInstrumentationMode, METH_VARARGS, "Set instrumentation mode" },
   { "set_progress", setProgress, METH_VARARGS, "Set simulation progress indicator (0..1)" },
   { "abort", simulatorAbort, METH_VARARGS, "Stop simulation now" },
   { NULL, NULL, 0, NULL }
}
```

Definition at line 70 of file py_control.cc.

Referenced by HooksPy::PyControl::setup().

7.538 common/scripting/py_control.d File Reference

7.539 common/scripting/py_dvfs.cc File Reference

```
#include "hooks_py.h"
#include "simulator.h"
#include "dvfs_manager.h"
#include "magic_server.h"
```

Functions

- static const **ComponentPeriod** * **getDomain** (**SInt64** domain_id, bool allow_global)
- static PyObject * **getFrequency** (PyObject *self, PyObject *args)
- static PyObject * **setFrequency** (PyObject *self, PyObject *args)

Variables

- static PyMethodDef **PyDvfsMethods** [ ]

7.539.1 Function Documentation

7.539.1.1 getDomain()

```
static const  ComponentPeriod* getDomain (
              SInt64 domain_id,
             bool allow_global )  [static]
```

Definition at line 7 of file py_dvfs.cc.

Referenced by getFrequency(), and setFrequency().

7.539.1.2 getFrequency()

```
static PyObject* getFrequency (
             PyObject * self,
             PyObject * args )  [static]
```

Definition at line 33 of file py_dvfs.cc.

References getDomain(), SubsecondTime::getFS(), and ComponentPeriod::getPeriod().

7.539.1.3 setFrequency()

```
static PyObject* setFrequency (
             PyObject * self,
             PyObject * args )  [static]
```

Definition at line 50 of file py_dvfs.cc.

References getDomain().

7.539.2 Variable Documentation

7.539.2.1 PyDvfsMethods

```
PyMethodDef PyDvfsMethods[]  [static]
```

Initial value:

```
= {
   {"get_frequency",  getFrequency, METH_VARARGS, "Get core or global frequency, in MHz."},
   {"set_frequency",  setFrequency, METH_VARARGS, "Set core frequency, in MHz."},
   {NULL, NULL, 0, NULL}
}
```

Definition at line 69 of file py_dvfs.cc.

Referenced by HooksPy::PyDvfs::setup().

7.540 common/scripting/py_dvfs.d File Reference

7.541 common/scripting/py_hooks.cc File Reference

```
#include "hooks_py.h"
#include "subsecond_time.h"
#include "simulator.h"
#include "hooks_manager.h"
#include "magic_server.h"
#include "syscall_model.h"
#include "sim_api.h"
```

Functions

- static **SInt64** **hookCallbackResult** (PyObject *pResult)
- static **SInt64** **hookCallbackNone** (**UInt64** pFunc, **UInt64**)
- static **SInt64** **hookCallbackInt** (**UInt64** pFunc, **UInt64** argument)
- static **SInt64** **hookCallbackSubsecondTime** (**UInt64** pFunc, **UInt64** argument)
- static **SInt64** **hookCallbackString** (**UInt64** pFunc, **UInt64** _argument)
- static **SInt64** **hookCallbackMagicMarkerType** (**UInt64** pFunc, **UInt64** _argument)
- static **SInt64** **hookCallbackThreadCreateType** (**UInt64** pFunc, **UInt64** _argument)
- static **SInt64** **hookCallbackThreadTimeType** (**UInt64** pFunc, **UInt64** _argument)
- static **SInt64** **hookCallbackThreadStallType** (**UInt64** pFunc, **UInt64** _argument)
- static **SInt64** **hookCallbackThreadResumeType** (**UInt64** pFunc, **UInt64** _argument)
- static **SInt64** **hookCallbackThreadMigrateType** (**UInt64** pFunc, **UInt64** _argument)
- static **SInt64** **hookCallbackSyscallEnter** (**UInt64** pFunc, **UInt64** _argument)
- static **SInt64** **hookCallbackSyscallExit** (**UInt64** pFunc, **UInt64** _argument)
- static PyObject * **registerHook** (PyObject *self, PyObject *args)
- static PyObject * **triggerHookMagicUser** (PyObject *self, PyObject *args)

Variables

- static PyMethodDef **PyHooksMethods** [ ]

7.541.1 Function Documentation

7.541.1.1 hookCallbackInt()

```
static SInt64 hookCallbackInt (
              UInt64 pFunc,
              UInt64 argument ) [static]
```

Definition at line 28 of file py_hooks.cc.

References HooksPy::callPythonFunction(), and hookCallbackResult().

Referenced by registerHook().

7.541.1.2 hookCallbackMagicMarkerType()

```
static SInt64 hookCallbackMagicMarkerType (
              UInt64 pFunc,
              UInt64 _argument ) [static]
```

Definition at line 48 of file py_hooks.cc.

References MagicServer::MagicMarkerType::arg0, MagicServer::MagicMarkerType::arg1, HooksPy::callPython↩Function(), MagicServer::MagicMarkerType::core_id, hookCallbackResult(), MagicServer::MagicMarkerType::str, and MagicServer::MagicMarkerType::thread_id.

Referenced by registerHook().

7.541.1.3 hookCallbackNone()

```
static SInt64 hookCallbackNone (
              UInt64 pFunc,
              UInt64 ) [static]
```

Definition at line 22 of file py_hooks.cc.

References HooksPy::callPythonFunction(), and hookCallbackResult().

Referenced by registerHook().

7.541.1.4 hookCallbackResult()

```
static SInt64 hookCallbackResult (
            PyObject * pResult ) [static]
```

Definition at line 9 of file py_hooks.cc.

Referenced by hookCallbackInt(), hookCallbackMagicMarkerType(), hookCallbackNone(), hookCallbackString(), hookCallbackSubsecondTime(), hookCallbackSyscallEnter(), hookCallbackSyscallExit(), hookCallbackThreadCreateType(), hookCallbackThreadMigrateType(), hookCallbackThreadResumeType(), hookCallbackThreadStallType(), and hookCallbackThreadTimeType().

7.541.1.5 hookCallbackString()

```
static SInt64 hookCallbackString (
              UInt64 pFunc,
              UInt64 _argument ) [static]
```

Definition at line 41 of file py_hooks.cc.

References HooksPy::callPythonFunction(), and hookCallbackResult().

Referenced by registerHook().

7.541.1.6 hookCallbackSubsecondTime()

```
static SInt64 hookCallbackSubsecondTime (
              UInt64 pFunc,
              UInt64 argument ) [static]
```

Definition at line 34 of file py_hooks.cc.

References HooksPy::callPythonFunction(), SubsecondTime::getFS(), and hookCallbackResult().

Referenced by registerHook().

7.541.1.7 hookCallbackSyscallEnter()

```
static SInt64 hookCallbackSyscallEnter (
              UInt64 pFunc,
              UInt64 _argument ) [static]
```

Definition at line 94 of file py_hooks.cc.

References SyscallMdl::syscall_args_t::arg0, SyscallMdl::syscall_args_t::arg1, SyscallMdl::syscall_args_t::arg2, SyscallMdl::syscall_args_t::arg3, SyscallMdl::syscall_args_t::arg4, SyscallMdl::syscall_args_t::arg5, SyscallMdl::HookSyscallEnter::args, HooksPy::callPythonFunction(), SyscallMdl::HookSyscallEnter::core_id, SubsecondTime::getFS(), hookCallbackResult(), SyscallMdl::HookSyscallEnter::syscall_number, SyscallMdl::HookSyscallEnter::thread_id, and SyscallMdl::HookSyscallEnter::time.

Referenced by registerHook().

7.541.1.8 hookCallbackSyscallExit()

```
static SInt64 hookCallbackSyscallExit (
              UInt64 pFunc,
              UInt64 _argument ) [static]
```

Definition at line 103 of file py_hooks.cc.

References HooksPy::callPythonFunction(), SyscallMdl::HookSyscallExit::core_id, SyscallMdl::HookSyscallExit::emulated, SubsecondTime::getFS(), hookCallbackResult(), SyscallMdl::HookSyscallExit::ret_val, SyscallMdl::HookSyscallExit::thread_id, and SyscallMdl::HookSyscallExit::time.

Referenced by registerHook().

7.541.1.9 hookCallbackThreadCreateType()

```
static SInt64 hookCallbackThreadCreateType (
              UInt64 pFunc,
              UInt64 _argument ) [static]
```

Definition at line 55 of file py_hooks.cc.

References HooksPy::callPythonFunction(), HooksManager::ThreadCreate::creator_thread_id, hookCallbackResult(), and HooksManager::ThreadCreate::thread_id.

Referenced by registerHook().

7.541.1.10 hookCallbackThreadMigrateType()

```
static SInt64 hookCallbackThreadMigrateType (
              UInt64 pFunc,
              UInt64 _argument ) [static]
```

Definition at line 86 of file py_hooks.cc.

References HooksPy::callPythonFunction(), HooksManager::ThreadMigrate::core_id, SubsecondTime::getFS(), hookCallbackResult(), HooksManager::ThreadMigrate::thread_id, and HooksManager::ThreadMigrate::time.

Referenced by registerHook().

7.541.1.11 hookCallbackThreadResumeType()

```
static SInt64 hookCallbackThreadResumeType (
              UInt64 pFunc,
              UInt64 _argument ) [static]
```

Definition at line 78 of file py_hooks.cc.

References HooksPy::callPythonFunction(), SubsecondTime::getFS(), hookCallbackResult(), HooksManager::ThreadResume::thread_by, HooksManager::ThreadResume::thread_id, and HooksManager::ThreadResume::time.

Referenced by registerHook().

7.541.1.12 hookCallbackThreadStallType()

```
static  SInt64 hookCallbackThreadStallType (
              UInt64 pFunc,
              UInt64 _argument )  [static]
```

Definition at line 70 of file py_hooks.cc.

References HooksPy::callPythonFunction(), SubsecondTime::getFS(), hookCallbackResult(), HooksManager::ThreadStall::reason, ThreadManager::stall_type_names, HooksManager::ThreadStall::thread_id, and HooksManager::ThreadStall::time.

Referenced by registerHook().

7.541.1.13 hookCallbackThreadTimeType()

```
static  SInt64 hookCallbackThreadTimeType (
              UInt64 pFunc,
              UInt64 _argument )  [static]
```

Definition at line 62 of file py_hooks.cc.

References HooksPy::callPythonFunction(), SubsecondTime::getFS(), hookCallbackResult(), HooksManager::ThreadTime::thread_id, and HooksManager::ThreadTime::time.

Referenced by registerHook().

7.541.1.14 registerHook()

```
static PyObject* registerHook (
             PyObject * self,
             PyObject * args )  [static]
```

Definition at line 113 of file py_hooks.cc.

References HookType::HOOK_APPLICATION_EXIT, HookType::HOOK_APPLICATION_ROI_BEGIN, HookType::HOOK_APPLICATION_ROI_END, HookType::HOOK_APPLICATION_START, HookType::HOOK_CPUFREQ_CHANGE, HookType::HOOK_INSTR_COUNT, HookType::HOOK_INSTRUMENT_MODE, HookType::HOOK_MAGIC_MARKER, HookType::HOOK_MAGIC_USER, HookType::HOOK_PERIODIC, HookType::HOOK_PERIODIC_INS, HookType::HOOK_PRE_STAT_WRITE, HookType::HOOK_ROI_BEGIN, HookType::HOOK_ROI_END, HookType::HOOK_SIGUSR1, HookType::HOOK_SIM_END, HookType::HOOK_SIM_START, HookType::HOOK_SYSCALL_ENTER, HookType::HOOK_SYSCALL_EXIT, HookType::HOOK_THREAD_CREATE, HookType::HOOK_THREAD_EXIT, HookType::HOOK_THREAD_MIGRATE, HookType::HOOK_THREAD_RESUME, HookType::HOOK_THREAD_STALL, HookType::HOOK_THREAD_START, HookType::HOOK_TYPES_MAX, hookCallbackInt(), hookCallbackMagicMarkerType(), hookCallbackNone(), hookCallbackString(), hookCallbackSubsecondTime(), hookCallbackSyscallEnter(), hookCallbackSyscallExit(), hookCallbackThreadCreateType(), hookCallbackThreadMigrateType(), hookCallbackThreadResumeType(), hookCallbackThreadStallType(), and hookCallbackThreadTimeType().

7.541.1.15 triggerHookMagicUser()

```
static PyObject* triggerHookMagicUser (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 191 of file py_hooks.cc.

References INVALID_CORE_ID, and INVALID_THREAD_ID.

7.541.2 Variable Documentation

7.541.2.1 PyHooksMethods

```
PyMethodDef PyHooksMethods[]  [static]
```

Initial value:

```
= {
   {"register",  registerHook, METH_VARARGS, "Register callback function to a Sniper hook."},
   {"trigger_magic_user", triggerHookMagicUser, METH_VARARGS, "Trigger HOOK_MAGIC_USER hook."},
   {NULL, NULL, 0, NULL}
}
```

Definition at line 203 of file py_hooks.cc.

Referenced by HooksPy::PyHooks::setup().

7.542 common/scripting/py_hooks.d File Reference

7.543 common/scripting/py_mem.cc File Reference

```
#include "hooks_py.h"
#include "simulator.h"
#include "core_manager.h"
```

Functions

- static PyObject * **readMemory** (PyObject *self, PyObject *args)
- static PyObject * **readCstr** (PyObject *self, PyObject *args)

Variables

- static PyMethodDef **PyMemMethods** [ ]

7.543.1 Function Documentation

7.543.1.1 readCstr()

```
static PyObject* readCstr (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 27 of file py_mem.cc.

References Core::accessMemory(), LOG_ASSERT_ERROR, Core::MEM_MODELED_NONE, Core::NONE, and Core::READ.

7.543.1.2 readMemory()

```
static PyObject* readMemory (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 6 of file py_mem.cc.

References Core::accessMemory(), LOG_ASSERT_ERROR, Core::MEM_MODELED_NONE, Core::NONE, and Core::READ.

7.543.2 Variable Documentation

7.543.2.1 PyMemMethods

```
PyMethodDef PyMemMethods[]  [static]
```

Initial value:

```
= {
   { "read", readMemory, METH_VARARGS, "Read memory (core, address, size)" },
   { "read_cstr", readCstr, METH_VARARGS, "Read null-terminated string (core, address)" },
   { NULL, NULL, 0, NULL }
}
```

Definition at line 56 of file py_mem.cc.

Referenced by HooksPy::PyMem::setup().

7.544 common/scripting/py_mem.d File Reference

7.545 common/scripting/py_stats.cc File Reference

```
#include "hooks_py.h"
#include "simulator.h"
#include "clock_skew_minimization_object.h"
#include "stats.h"
#include "magic_server.h"
#include "thread_stats_manager.h"
```

Classes

- struct **statsGetterObject**

Functions

- static PyObject * **getStatsValue** (PyObject *self, PyObject *args)
- static PyObject * **statsGetterGet** (PyObject *self, PyObject *args, PyObject *kw)
- static PyObject * **getStatsGetter** (PyObject *self, PyObject *args)
- static PyObject * **writeStats** (PyObject *self, PyObject *args)
- static **UInt64** **statsCallback** (String objectName, **UInt32** index, String metricName, **UInt64** _pFunc)
- static PyObject * **registerStats** (PyObject *self, PyObject *args)
- static PyObject * **registerPerThread** (PyObject *self, PyObject *args)
- static PyObject * **writeMarker** (PyObject *self, PyObject *args)
- static PyObject * **getTime** (PyObject *self, PyObject *args)
- static PyObject * **getIcount** (PyObject *self, PyObject *args)

Variables

- static PyTypeObject **statsGetterType**
- static PyMethodDef **PyStatsMethods** []

7.545.1 Function Documentation

7.545.1.1 getIcount()

```
static PyObject* getIcount (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 235 of file py_stats.cc.

References MagicServer::getGlobalInstructionCount().

7.545.1.2 getStatsGetter()

```
static PyObject* getStatsGetter (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 102 of file py_stats.cc.

References statsGetterObject::metric, and statsGetterType.

7.545.1.3 getStatsValue()

```
static PyObject* getStatsValue (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 14 of file py_stats.cc.

References StatsMetricBase::recordMetric().

7.545.1.4 getTime()

```
static PyObject* getTime (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 223 of file py_stats.cc.

References SubsecondTime::getFS().

7.545.1.5 registerPerThread()

```
static PyObject* registerPerThread (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 186 of file py_stats.cc.

References ThreadStatNamedStat::registerStat().

7.545.1.6 registerStats()

```
static PyObject* registerStats (
             PyObject * self,
             PyObject * args )  [static]
```

Definition at line 165 of file py_stats.cc.

References statsCallback().

7.545.1.7 statsCallback()

```
static  UInt64 statsCallback (
             String objectName,
              UInt32 index,
             String metricName,
              UInt64 _pFunc )  [static]
```

Definition at line 146 of file py_stats.cc.

References HooksPy::callPythonFunction(), and LOG_PRINT_WARNING.

Referenced by registerStats().

7.545.1.8 statsGetterGet()

```
static PyObject* statsGetterGet (
             PyObject * self,
             PyObject * args,
             PyObject * kw )  [static]
```

Definition at line 43 of file py_stats.cc.

References statsGetterObject::metric, and StatsMetricBase::recordMetric().

7.545.1.9 writeMarker()

```
static PyObject* writeMarker (
             PyObject * self,
             PyObject * args )  [static]
```

Definition at line 204 of file py_stats.cc.

References INVALID_CORE_ID, and INVALID_THREAD_ID.

7.545.1.10 writeStats()

```
static PyObject* writeStats (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 129 of file py_stats.cc.

7.545.2 Variable Documentation

7.545.2.1 PyStatsMethods

```
PyMethodDef PyStatsMethods[]  [static]
```

Initial value:

```
= {
   {"get",  getStatsValue, METH_VARARGS, "Retrieve current value of statistic (objectName, index,
       metricName)."},
   {"getter", getStatsGetter, METH_VARARGS, "Return object to retrieve statistics value."},
   {"write", writeStats, METH_VARARGS, "Write statistics (<prefix>, [<filename>])."},
   {"register", registerStats, METH_VARARGS, "Register callback that defines statistics value for
       (objectName, index, metricName)."},
   {"register_per_thread", registerPerThread, METH_VARARGS, "Add a per-thread statistic (perthreadName)
       based on a named statistic (objectName, metricName)."},
   {"marker", writeMarker, METH_VARARGS, "Record a marker (coreid, threadid, arg0, arg1, [description])."},
   {"time", getTime, METH_VARARGS, "Retrieve the current global time in femtoseconds (approximate, last
       barrier)."},
   {"icount", getIcount, METH_VARARGS, "Retrieve current global instruction count."},
   {NULL, NULL, 0, NULL}
}
```

Definition at line 246 of file py_stats.cc.

Referenced by HooksPy::PyStats::setup().

7.545.2.2 statsGetterType

```
PyTypeObject statsGetterType  [static]
```

Definition at line 50 of file py_stats.cc.

Referenced by getStatsGetter(), and HooksPy::PyStats::setup().

7.546 common/scripting/py_stats.d File Reference

7.547 common/scripting/py_thread.cc File Reference

```
#include "hooks_py.h"
#include "simulator.h"
#include "thread_manager.h"
#include "thread.h"
#include "scheduler.h"
```

Functions

- static PyObject * **getNthreads** (PyObject *self, PyObject *args)
- static PyObject * **getThreadAppid** (PyObject *self, PyObject *args)
- static PyObject * **getThreadName** (PyObject *self, PyObject *args)
- static PyObject * **getThreadAffinity** (PyObject *self, PyObject *args)
- static PyObject * **setThreadAffinity** (PyObject *self, PyObject *args)

Variables

- static PyMethodDef **PyThreadMethods** [ ]

7.547.1 Function Documentation

7.547.1.1 getNthreads()

```
static PyObject* getNthreads (
             PyObject * self,
             PyObject * args )  [static]
```

Definition at line 8 of file py_thread.cc.

7.547.1.2 getThreadAffinity()

```
static PyObject* getThreadAffinity (
             PyObject * self,
             PyObject * args )  [static]
```

Definition at line 62 of file py_thread.cc.

References INVALID_THREAD_ID.

7.547.1.3 getThreadAppid()

```
static PyObject* getThreadAppid (
             PyObject * self,
             PyObject * args )  [static]
```

Definition at line 14 of file py_thread.cc.

References Thread::getAppId(), and INVALID_THREAD_ID.

7.547.1.4 getThreadName()

```
static PyObject* getThreadName (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 38 of file py_thread.cc.

References Thread::getName(), and INVALID_THREAD_ID.

7.547.1.5 setThreadAffinity()

```
static PyObject* setThreadAffinity (
            PyObject * self,
            PyObject * args )  [static]
```

Definition at line 95 of file py_thread.cc.

References INVALID_THREAD_ID.

7.547.2 Variable Documentation

7.547.2.1 PyThreadMethods

```
PyMethodDef PyThreadMethods[]  [static]
```

Initial value:

```
= {
   { "get_nthreads", getNthreads, METH_VARARGS, "Get number of threads" },
   { "get_thread_appid", getThreadAppid, METH_VARARGS, "Get application ID for a thread" },
   { "get_thread_name", getThreadName, METH_VARARGS, "Get thread name" },
   { "get_thread_affinity", getThreadAffinity, METH_VARARGS, "Get thread affinity" },
   { "set_thread_affinity", setThreadAffinity, METH_VARARGS, "Set thread affinity" },
   { NULL, NULL, 0, NULL }
}
```

Definition at line 139 of file py_thread.cc.

Referenced by HooksPy::PyThread::setup().

7.548 common/scripting/py_thread.d File Reference

7.549 common/system/barrier_sync_client.cc File Reference

```
#include "barrier_sync_client.h"
#include "simulator.h"
#include "config.h"
#include "core.h"
#include "performance_model.h"
#include "subsecond_time.h"
#include "dvfs_manager.h"
#include "config.hpp"
```

7.550 common/system/barrier_sync_client.d File Reference

7.551 common/system/barrier_sync_client.h File Reference

```
#include "fixed_types.h"
#include "clock_skew_minimization_object.h"
#include "packetize.h"
#include "subsecond_time.h"
```

Classes

- class **BarrierSyncClient**

7.552 common/system/barrier_sync_server.cc File Reference

```
#include "barrier_sync_client.h"
#include "barrier_sync_server.h"
#include "simulator.h"
#include "core_manager.h"
#include "core.h"
#include "thread.h"
#include "performance_model.h"
#include "hooks_manager.h"
#include "syscall_server.h"
#include "config.h"
#include "log.h"
#include "stats.h"
#include "config.hpp"
#include "circular_log.h"
#include <algorithm>
```

7.553 common/system/barrier_sync_server.d File Reference

7.554 common/system/barrier_sync_server.h File Reference

```
#include "fixed_types.h"
#include "cond.h"
#include "hooks_manager.h"
#include <vector>
```

Classes

- class **BarrierSyncServer**

7.555 common/system/cache_efficiency_tracker.h File Reference

```
#include "cache_block_info.h"
#include "hit_where.h"
#include "core.h"
```

Classes

- struct **CacheEfficiencyTracker::Callbacks**

Namespaces

- **CacheEfficiencyTracker**

Typedefs

- typedef **UInt64**(∗ **CacheEfficiencyTracker::CallbackGetOwner**) (**UInt64** user, **core_id_t** core_id, **UInt64** address)
- typedef void(∗ **CacheEfficiencyTracker::CallbackNotifyAccess**) (**UInt64** user, **UInt64** owner, **Core**↩ **::mem_op_t** mem_op_type, **HitWhere::where_t** hit_where)
- typedef void(∗ **CacheEfficiencyTracker::CallbackNotifyEvict**) (**UInt64** user, bool on_roi_end, **UInt64** owner, **UInt64** evictor, **CacheBlockInfo::BitsUsedType** bits_used, **UInt32** bits_total)

7.556 common/system/clock_skew_minimization_object.cc File Reference

```
#include "clock_skew_minimization_object.h"
#include "barrier_sync_client.h"
#include "barrier_sync_server.h"
#include "simulator.h"
#include "log.h"
#include "config.hpp"
```

7.557 common/system/clock_skew_minimization_object.d File Reference

7.558 common/system/clock_skew_minimization_object.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include "log.h"
```

Classes

- class **ClockSkewMinimizationObject**
- class **ClockSkewMinimizationClient**
- class **ClockSkewMinimizationManager**
- class **ClockSkewMinimizationServer**

7.559 common/system/core_manager.cc File Reference

```
#include <sched.h>
#include <linux/unistd.h>
#include <sys/types.h>
#include <sys/syscall.h>
#include <unistd.h>
#include <limits.h>
#include <algorithm>
#include <vector>
#include "core_manager.h"
#include "core.h"
#include "network.h"
#include "cache.h"
#include "config.h"
#include "log.h"
```

7.560 common/system/core_manager.d File Reference

7.561 common/system/core_manager.h File Reference

```
#include "fixed_types.h"
#include "tls.h"
#include "lock.h"
#include "log.h"
#include "core.h"
#include <iostream>
#include <fstream>
#include <map>
#include <vector>
```

Classes

- class **CoreManager**

7.562 common/system/core_thread.cc File Reference

```
#include "core_thread.h"
#include "core_manager.h"
#include "performance_model.h"
#include "log.h"
#include "simulator.h"
#include "core.h"
#include "sim_thread_manager.h"
#include "sim_api.h"
#include <unistd.h>
```

7.563 common/system/core_thread.d File Reference

7.564 common/system/core_thread.h File Reference

```
#include "_thread.h"
#include "fixed_types.h"
#include "network.h"
```

Classes

- class **CoreThread**

7.565 common/system/dvfs_manager.cc File Reference

```
#include <cassert>
#include "dvfs_manager.h"
#include "simulator.h"
#include "core_manager.h"
#include "core.h"
#include "performance_model.h"
#include "instruction.h"
#include "log.h"
#include "config.hpp"
```

7.566 common/system/dvfs_manager.d File Reference

7.567 common/system/dvfs_manager.h File Reference

```
#include "subsecond_time.h"
#include <vector>
```

Classes

- class **DvfsManager**

7.568 common/system/fastforward_performance_manager.cc File Reference

```
#include "fastforward_performance_manager.h"
#include "simulator.h"
#include "config.hpp"
#include "thread.h"
#include "hooks_manager.h"
#include "core_manager.h"
#include "performance_model.h"
#include "fastforward_performance_model.h"
```

7.569 common/system/fastforward_performance_manager.d File Reference

7.570 common/system/fastforward_performance_manager.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
```

Classes

- class **FastForwardPerformanceManager**

7.571 common/system/hooks_manager.cc File Reference

```
#include "hooks_manager.h"
#include "log.h"
```

7.572 common/system/hooks_manager.d File Reference

7.573 common/system/hooks_manager.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include "thread_manager.h"
#include <vector>
#include <unordered_map>
```

Classes

- class **HookType**
- struct **std::hash< HookType::hook_type_t >**
- class **HooksManager**
- struct **HooksManager::HookCallback**
- struct **HooksManager::ThreadCreate**
- struct **HooksManager::ThreadTime**
- struct **HooksManager::ThreadStall**
- struct **HooksManager::ThreadResume**
- struct **HooksManager::ThreadMigrate**

Namespaces

- **std**

7.574 common/system/hooks_manager_init.cc File Reference

```
#include "hooks_manager.h"
#include "hooks_py.h"
#include "subsecond_time.h"
#include "fixed_point.h"
#include "simulator.h"
#include "stats.h"
#include "dvfs_manager.h"
```

Functions

- void **hook_print_core0_ipc** (void *, **subsecond_time_t** _time)

7.574.1 Function Documentation

7.574.1.1 hook_print_core0_ipc()

```
void hook_print_core0_ipc (
            void * ,
             subsecond_time_t _time )
```

Definition at line 14 of file hooks_manager_init.cc.

References SubsecondTime::divideRounded(), TFixedPoint< __UINT64_C(0x4000)>::floor(), SubsecondTime::↩
FS(), SubsecondTime::getNS(), LOG_ASSERT_ERROR, and StatsMetricBase::recordMetric().

7.575 common/system/hooks_manager_init.d File Reference

7.576 common/system/inst_mode.cc File Reference

```
#include "log.h"
#include "inst_mode.h"
```

Functions

- **__attribute__** ((weak)) void **InstMode**

Variables

- const char * **inst_mode_names** [ ]

7.576.1 Function Documentation

7.576.1.1 __attribute__()

```
__attribute__ (
             (weak)  )
```

Definition at line 17 of file inst_mode.cc.

References LOG_PRINT_ERROR.

7.576.2 Variable Documentation

7.576.2.1 inst_mode_names

```
const char* inst_mode_names[]
```

Initial value:

```
= {
   "INVALID", "DETAILED", "CACHE_ONLY", "FAST_FORWARD"
}
```

Definition at line 4 of file inst_mode.cc.

Referenced by Simulator::printInstModeSummary(), and Simulator::setInstrumentationMode().

7.577 pin/inst_mode.cc File Reference

```
#include "inst_mode.h"
#include "inst_mode_macros.h"
#include "pin.H"
```

7.578 common/system/inst_mode.d File Reference

7.579 common/system/inst_mode.h File Reference

```
#include "fixed_types.h"
```

Classes

- class **InstMode**

Variables

- const char * **inst_mode_names** [ ]

7.579.1 Variable Documentation

7.579.1.1 inst_mode_names

```
const char* inst_mode_names[]
```

Definition at line 4 of file inst_mode.cc.

Referenced by Simulator::printInstModeSummary(), and Simulator::setInstrumentationMode().

7.580 common/system/magic_client.cc File Reference

```
#include "magic_client.h"
#include "magic_server.h"
#include "sim_api.h"
#include "simulator.h"
#include "core.h"
#include "core_manager.h"
#include "thread.h"
#include "thread_manager.h"
```

Functions

- static **UInt64** **handleMagic** (**thread_id_t** thread_id, **UInt64** cmd, **UInt64** arg0=0, **UInt64** arg1=0)
- void **setInstrumentationMode** (**UInt64** opt)
- **UInt64** **handleMagicInstruction** (**thread_id_t** thread_id, **UInt64** cmd, **UInt64** arg0, **UInt64** arg1)

7.580.1 Function Documentation

7.580.1.1 handleMagic()

```
static  UInt64 handleMagic (
              thread_id_t thread_id,
              UInt64 cmd,
              UInt64 arg0 = 0,
              UInt64 arg1 = 0 )  [static]
```

Definition at line 10 of file magic_client.cc.

References Thread::getCore(), Core::getId(), INVALID_CORE_ID, and INVALID_THREAD_ID.

Referenced by handleMagicInstruction(), and setInstrumentationMode().

7.580.1.2 handleMagicInstruction()

```
 UInt64 handleMagicInstruction (
              thread_id_t thread_id,
              UInt64 cmd,
              UInt64 arg0,
              UInt64 arg1 )
```

Definition at line 22 of file magic_client.cc.

References Core::getId(), handleMagic(), and LOG_PRINT_WARNING_ONCE.

Referenced by handleMagic(), and TraceThread::handleMagicFunc().

7.580.1.3 setInstrumentationMode()

```
void setInstrumentationMode (
              UInt64 opt )
```

Definition at line 17 of file magic_client.cc.

References handleMagic(), and INVALID_THREAD_ID.

7.581 common/system/magic_client.d File Reference

7.582 common/system/magic_client.h File Reference

```
#include "fixed_types.h"
```

Functions

- **UInt64 handleMagicInstruction** (**thread_id_t** thread_id, **UInt64** cmd, **UInt64** arg0, **UInt64** arg1)
- void **setInstrumentationMode** (**UInt64** opt)

7.582.1 Function Documentation

7.582.1.1 handleMagicInstruction()

```
UInt64 handleMagicInstruction (
              thread_id_t thread_id,
              UInt64 cmd,
              UInt64 arg0,
              UInt64 arg1 )
```

Definition at line 22 of file magic_client.cc.

References Core::getId(), handleMagic(), and LOG_PRINT_WARNING_ONCE.

Referenced by handleMagic(), and TraceThread::handleMagicFunc().

7.582.1.2 setInstrumentationMode()

```
void setInstrumentationMode (
              UInt64 opt )
```

Definition at line 17 of file magic_client.cc.

References handleMagic(), and INVALID_THREAD_ID.

7.583 common/system/magic_server.cc File Reference

```
#include "magic_server.h"
#include "sim_api.h"
#include "simulator.h"
#include "thread_manager.h"
#include "logmem.h"
#include "performance_model.h"
#include "fastforward_performance_model.h"
#include "core_manager.h"
#include "dvfs_manager.h"
#include "hooks_manager.h"
#include "stats.h"
#include "timer.h"
#include "thread.h"
```

Functions

- **__attribute__** ((weak)) void **PinDetach**(void)
- void **print_allocations** ()

Variables

- static **Timer t_start**
- **UInt64 ninstrs_start**

7.583.1 Function Documentation

7.583.1.1 __attribute__()

```
__attribute__ (
             (weak)  )
```

Definition at line 118 of file magic_server.cc.

7.583.1.2 print_allocations()

```
void print_allocations ( )
```

7.583.2 Variable Documentation

7.583.2.1 ninstrs_start

```
UInt64 ninstrs_start
```

Definition at line 117 of file magic_server.cc.

Referenced by MagicServer::disablePerformance(), and MagicServer::enablePerformance().

7.583.2.2 t_start

```
Timer t_start  [static]
```

Definition at line 116 of file magic_server.cc.

Referenced by NucaCache::accessDataArray(), DramCache::accessDataArray(), QueueModelContention::computeQueueDelay(), MagicServer::disablePerformance(), ParametricDramDirectoryMSI::CacheCntlr::doPrefetch(), MagicServer::enablePerformance(), ContentionModel::getBarrierCompletionTime(), ContentionModel::getCompletionTime(), ContentionModel::getNumUsed(), ContentionModel::getStartTime(), ContentionModel::hasFreeSlot(), ParametricDramDirectoryMSI::CacheCntlr::processMemOpFromCore(), MagicServer::setPerformance(), and NetworkModelBusGlobal::useBus().

7.584 common/system/magic_server.d File Reference

7.585 common/system/magic_server.h File Reference

```
#include "fixed_types.h"
#include "progress.h"
```

Classes

- class **MagicServer**
- struct **MagicServer::MagicMarkerType**

7.586 common/system/memory_tracker.cc File Reference

```
#include "memory_tracker.h"
#include "simulator.h"
#include "thread_manager.h"
#include "thread.h"
#include <unordered_set>
```

7.587 common/system/memory_tracker.d File Reference

7.588 common/system/memory_tracker.h File Reference

```
#include "fixed_types.h"
#include "lock.h"
#include "routine_tracer.h"
#include "cache_efficiency_tracker.h"
#include <vector>
#include <map>
#include <unordered_map>
```

Classes

- class **MemoryTracker**
- class **MemoryTracker::RoutineTracerThread**
- class **MemoryTracker::RoutineTracer**
- struct **MemoryTracker::AllocationSite**
- struct **MemoryTracker::Allocation**

7.589 common/system/pthread_emu.cc File Reference

```
#include "simulator.h"
#include "core_manager.h"
#include "pthread_emu.h"
#include "thread_manager.h"
#include "performance_model.h"
#include "sync_api.h"
#include "log.h"
#include "stats.h"
#include "logmem.h"
#include "config.hpp"
#include <stdlib.h>
#include <malloc.h>
#include <errno.h>
```

Classes

- struct **PthreadEmu::pthread_counters_t**

Namespaces

- **PthreadEmu**

Functions

- void **PthreadEmu::pthreadCount** (pthread_enum_t function, **Core** ∗core, **SubsecondTime** delay_sync, **SubsecondTime** delay_mem)
- **IntPtr PthreadEmu::futexHbAddress** (pthread_mutex_t ∗mux)
- void **PthreadEmu::updateState** (**Core** ∗core, state_t state, **SubsecondTime** delay)
- void **PthreadEmu::init** ()
- **IntPtr PthreadEmu::MutexInit** (pthread_mutex_t ∗mux, pthread_mutexattr_t ∗attributes)
- **IntPtr PthreadEmu::MutexLock** (pthread_mutex_t ∗mux)
- **IntPtr PthreadEmu::MutexTrylock** (pthread_mutex_t ∗mux)
- **IntPtr PthreadEmu::MutexUnlock** (pthread_mutex_t ∗mux)
- **IntPtr PthreadEmu::CondInit** (pthread_cond_t ∗cond, pthread_condattr_t ∗attributes)
- **IntPtr PthreadEmu::CondWait** (pthread_cond_t ∗cond, pthread_mutex_t ∗mutex)
- **IntPtr PthreadEmu::CondSignal** (pthread_cond_t ∗cond)
- **IntPtr PthreadEmu::CondBroadcast** (pthread_cond_t ∗cond)
- **IntPtr PthreadEmu::BarrierInit** (pthread_barrier_t ∗barrier, pthread_barrierattr_t ∗attributes, unsigned count)
- **IntPtr PthreadEmu::BarrierWait** (pthread_barrier_t ∗barrier)

Variables

- bool **PthreadEmu::pthread_stats_added** = false
- const char ∗ **PthreadEmu::pthread_names** [ ]
- struct **PthreadEmu::pthread_counters_t** ∗ **PthreadEmu::pthread_counters** = NULL
- static std::unordered_map< pthread_mutex_t ∗, **IntPtr** > **PthreadEmu::futex_map**
- static **Lock PthreadEmu::futex_map_lock**
- static **Lock PthreadEmu::trace_lock**
- static FILE ∗ **PthreadEmu::trace_fp** = NULL

7.590 common/system/pthread_emu.d File Reference

7.591 common/system/pthread_emu.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include <pthread.h>
```

Namespaces

- **PthreadEmu**

Enumerations

- enum **PthreadEmu::pthread_enum_t** {
  PthreadEmu::PTHREAD_MUTEX_LOCK = 0, **PthreadEmu::PTHREAD_MUTEX_TRYLOCK**, **Pthread↩ Emu::PTHREAD_MUTEX_UNLOCK**, **PthreadEmu::PTHREAD_COND_WAIT**,
  PthreadEmu::PTHREAD_COND_SIGNAL, **PthreadEmu::PTHREAD_COND_BROADCAST**, **Pthread↩ Emu::PTHREAD_BARRIER_WAIT**, **PthreadEmu::PTHREAD_ENUM_LAST** }
- enum **PthreadEmu::state_t** {
  PthreadEmu::STATE_STOPPED, **PthreadEmu::STATE_RUNNING**, **PthreadEmu::STATE_WAITING**, **PthreadEmu::STATE_INREGION**,
  PthreadEmu::STATE_SEPARATOR, **PthreadEmu::STATE_MAX**, **PthreadEmu::STATE_BY_RETURN** }

Functions

- void **PthreadEmu::init** ()
- void **PthreadEmu::pthreadCount** (pthread_enum_t function, **Core** *core, **SubsecondTime** delay_sync, **SubsecondTime** delay_mem)
- void **PthreadEmu::updateState** (**Core** *core, state_t state, **SubsecondTime** delay)
- **IntPtr PthreadEmu::MutexInit** (pthread_mutex_t *mux, pthread_mutexattr_t *attributes)
- **IntPtr PthreadEmu::MutexLock** (pthread_mutex_t *mux)
- **IntPtr PthreadEmu::MutexTrylock** (pthread_mutex_t *mux)
- **IntPtr PthreadEmu::MutexUnlock** (pthread_mutex_t *mux)
- **IntPtr PthreadEmu::CondInit** (pthread_cond_t *cond, pthread_condattr_t *attributes)
- **IntPtr PthreadEmu::CondWait** (pthread_cond_t *cond, pthread_mutex_t *mutex)
- **IntPtr PthreadEmu::CondSignal** (pthread_cond_t *cond)
- **IntPtr PthreadEmu::CondBroadcast** (pthread_cond_t *cond)
- **IntPtr PthreadEmu::BarrierInit** (pthread_barrier_t *barrier, pthread_barrierattr_t *attributes, unsigned count)
- **IntPtr PthreadEmu::BarrierWait** (pthread_barrier_t *barrier)

7.592 common/system/routine_tracer.cc File Reference

```
#include "routine_tracer.h"
#include "simulator.h"
#include "magic_server.h"
#include "config.hpp"
#include "routine_tracer_print.h"
#include "routine_tracer_ondemand.h"
#include "routine_tracer_funcstats.h"
#include "memory_tracker.h"
#include <cstring>
```

7.593 common/system/routine_tracer.d File Reference

7.594 common/system/routine_tracer.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include <deque>
#include <boost/functional/hash.hpp>
```

Classes

- struct **std::hash**< **std::deque**< **T** > >
- class **RoutineTracerThread**
- class **RoutineTracer**
- class **RoutineTracer::Routine**

Namespaces

- **std**

Typedefs

- typedef std::deque< **IntPtr** > **CallStack**

7.594.1 Typedef Documentation

7.594.1.1 CallStack

```
typedef std::deque< IntPtr>  CallStack
```

Definition at line 23 of file routine_tracer.h.

7.595 common/system/routine_tracer_funcstats.cc File Reference

```
#include "routine_tracer_funcstats.h"
#include "simulator.h"
#include "core_manager.h"
#include "thread.h"
#include "core.h"
#include "performance_model.h"
#include "log.h"
#include "stats.h"
#include "cache_efficiency_tracker.h"
#include "utils.h"
#include <sstream>
```

Classes

- class **ThreadStatAggregates**

7.596 common/system/routine_tracer_funcstats.d File Reference

7.597 common/system/routine_tracer_funcstats.h File Reference

```
#include "routine_tracer.h"
#include "thread_stats_manager.h"
#include "cache_efficiency_tracker.h"
#include <unordered_map>
```

Classes

- class **RoutineTracerFunctionStats**
- class **RoutineTracerFunctionStats::Routine**
- class **RoutineTracerFunctionStats::RtnMaster**
- class **RoutineTracerFunctionStats::RtnThread**
- class **RoutineTracerFunctionStats::ThreadStatAggregates**
- class **RoutineTracerFunctionStats::ThreadStatCpiMem**

7.598 common/system/routine_tracer_ondemand.cc File Reference

```
#include "routine_tracer_ondemand.h"
#include "simulator.h"
#include "thread_manager.h"
#include "thread.h"
#include "syscall_model.h"
#include "hooks_manager.h"
#include <signal.h>
```

7.599 common/system/routine_tracer_ondemand.d File Reference

7.600 common/system/routine_tracer_ondemand.h File Reference

```
#include "routine_tracer.h"
#include <unordered_map>
```

Classes

- class **RoutineTracerOndemand**
- class **RoutineTracerOndemand::RtnMaster**
- class **RoutineTracerOndemand::RtnThread**

7.601 common/system/routine_tracer_print.cc File Reference

```
#include "routine_tracer_print.h"
#include "thread.h"
```

7.602 common/system/routine_tracer_print.d File Reference

7.603 common/system/routine_tracer_print.h File Reference

```
#include "routine_tracer.h"
#include <unordered_map>
```

Classes

- class **RoutineTracerPrint**
- class **RoutineTracerPrint::RtnMaster**
- class **RoutineTracerPrint::RtnThread**

7.604 common/system/sim_thread.cc File Reference

```
#include "sim_thread.h"
#include "core_manager.h"
#include "log.h"
#include "simulator.h"
#include "core.h"
#include "sim_thread_manager.h"
#include "sim_api.h"
```

7.605 common/system/sim_thread.d File Reference

7.606 common/system/sim_thread.h File Reference

```
#include "_thread.h"
#include "fixed_types.h"
#include "network.h"
```

Classes

- class **SimThread**

7.607 common/system/sim_thread_manager.cc File Reference

```
#include "sim_thread_manager.h"
#include "lock.h"
#include "log.h"
#include "config.h"
#include "simulator.h"
```

7.608 common/system/sim_thread_manager.d File Reference

7.609 common/system/sim_thread_manager.h File Reference

```
#include "sim_thread.h"
#include "core_thread.h"
```

Classes

- class **SimThreadManager**

7.610 common/system/simulator.cc File Reference

```
#include "simulator.h"
#include "log.h"
#include "core.h"
#include "core_manager.h"
#include "thread_manager.h"
#include "sync_server.h"
#include "syscall_server.h"
#include "magic_server.h"
#include "sim_thread_manager.h"
#include "clock_skew_minimization_object.h"
#include "fastforward_performance_manager.h"
#include "fxsupport.h"
#include "timer.h"
#include "stats.h"
#include "thread_stats_manager.h"
#include "pthread_emu.h"
#include "trace_manager.h"
#include "dvfs_manager.h"
#include "hooks_manager.h"
#include "sampling_manager.h"
#include "fault_injection.h"
#include "routine_tracer.h"
#include "instruction.h"
#include "config.hpp"
#include "magic_client.h"
#include "tags.h"
#include "instruction_tracer.h"
#include "memory_tracker.h"
#include "circular_log.h"
#include <sstream>
```

7.611 common/system/simulator.d File Reference

7.612 common/system/simulator.h File Reference

```
#include "config.h"
#include "log.h"
#include "inst_mode.h"
#include <decoder.h>
```

Classes

- class **Simulator**

Namespaces

- **config**

Functions

- **__attribute__** ((unused)) static **Simulator** *Sim()

7.612.1 Function Documentation

7.612.1.1 __attribute__()

```
__attribute__ (
             (unused)  )
```

Definition at line 127 of file simulator.h.

References Simulator::getSingleton().

7.613 common/system/sync_client.cc File Reference

```
#include "sync_client.h"
#include "sync_server.h"
#include "simulator.h"
#include "thread.h"
#include "thread_manager.h"
#include "core.h"
#include "performance_model.h"
#include "instruction.h"
#include <iostream>
```

7.614 common/system/sync_client.d File Reference

7.615 common/system/sync_client.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include "sync_api.h"
```

Classes

- class **SyncClient**

7.616 common/system/sync_server.cc File Reference

```
#include "sync_server.h"
#include "sync_client.h"
#include "simulator.h"
#include "thread_manager.h"
#include "subsecond_time.h"
#include "config.hpp"
```

7.617 common/system/sync_server.d File Reference

7.618 common/system/sync_server.h File Reference

```
#include "fixed_types.h"
#include "sync_api.h"
#include "transport.h"
#include "network.h"
#include "packetize.h"
#include "stable_iterator.h"
#include <queue>
#include <vector>
#include <limits.h>
#include <string.h>
#include <unordered_map>
```

Classes

- class **SimMutex**
- class **SimCond**
- class **SimCond::CondWaiter**
- class **SimBarrier**
- class **SyncServer**

7.619 common/system/syscall_server.cc File Reference

```
#include "syscall_server.h"
#include "core.h"
#include "config.h"
#include "config.hpp"
#include "simulator.h"
#include "hooks_manager.h"
#include "thread_manager.h"
#include "thread.h"
#include "core_manager.h"
#include "log.h"
#include "circular_log.h"
#include <sys/syscall.h>
#include "os_compat.h"
```

7.620 common/system/syscall_server.d File Reference

7.621 common/system/syscall_server.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include <iostream>
#include <unordered_map>
#include <list>
#include <linux/futex.h>
#include <sys/time.h>
#include <errno.h>
```

Classes

- class **SimFutex**
- struct **SimFutex::Waiter**
- class **SyscallServer**
- struct **SyscallServer::futex_args_t**

7.622 common/system/thread_manager.cc File Reference

```
#include "thread_manager.h"
#include "core_manager.h"
#include "performance_model.h"
#include "instruction.h"
#include "hooks_manager.h"
#include "config.h"
#include "log.h"
#include "stats.h"
#include "transport.h"
#include "simulator.h"
#include "clock_skew_minimization_object.h"
#include "core.h"
#include "thread.h"
#include "scheduler.h"
#include "syscall_server.h"
#include "circular_log.h"
#include <sys/syscall.h>
#include "os_compat.h"
```

7.623 common/system/thread_manager.d File Reference

7.624 common/system/thread_manager.h File Reference

```
#include "fixed_types.h"
#include "semaphore.h"
```

```
#include "core.h"
#include "lock.h"
#include "subsecond_time.h"
#include <vector>
#include <queue>
```

Classes

- class **ThreadManager**
- struct **ThreadManager::ThreadSpawnRequest**
- struct **ThreadManager::ThreadState**

7.625 common/system/thread_stats_manager.cc File Reference

```
#include "thread_stats_manager.h"
#include "simulator.h"
#include "core_manager.h"
#include "performance_model.h"
#include "thread.h"
#include "stats.h"
#include <cstring>
```

7.626 common/system/thread_stats_manager.d File Reference

7.627 common/system/thread_stats_manager.h File Reference

```
#include "fixed_types.h"
#include "subsecond_time.h"
#include "hooks_manager.h"
#include "bottlegraph.h"
```

Classes

- class **ThreadStatsManager**
- class **ThreadStatsManager::ThreadStats**
- struct **ThreadStatsManager::StatCallback**
- class **ThreadStatNamedStat**

7.628 common/trace_frontend/trace_manager.cc File Reference

```
#include "trace_manager.h"
#include "trace_thread.h"
#include "simulator.h"
#include "thread_manager.h"
#include "hooks_manager.h"
#include "config.hpp"
#include "sim_api.h"
#include "stats.h"
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
```

7.629 common/trace_frontend/trace_manager.d File Reference

7.630 common/trace_frontend/trace_manager.h File Reference

```
#include "fixed_types.h"
#include "semaphore.h"
#include "core.h"
#include "_thread.h"
#include <vector>
```

Classes

- class **TraceManager**
- class **TraceManager::Monitor**
- struct **TraceManager::app_info_t**

7.631 common/trace_frontend/trace_thread.cc File Reference

```
#include "trace_thread.h"
#include "trace_manager.h"
#include "simulator.h"
#include "core_manager.h"
#include "thread_manager.h"
#include "thread.h"
#include "dvfs_manager.h"
#include "instruction.h"
#include "dynamic_instruction.h"
#include "performance_model.h"
#include "instruction_decoder_wlib.h"
#include "config.hpp"
#include "syscall_model.h"
#include "core.h"
#include "magic_client.h"
```

```
#include "branch_predictor.h"
#include "rng.h"
#include "routine_tracer.h"
#include "sim_api.h"
#include "stats.h"
#include <unistd.h>
#include <sys/syscall.h>
#include <x86_decoder.h>
```

7.632 common/trace_frontend/trace_thread.d File Reference

7.633 common/trace_frontend/trace_thread.h File Reference

```
#include "fixed_types.h"
#include "_thread.h"
#include "thread.h"
#include "core.h"
#include "sift_reader.h"
#include "operand.h"
#include "semaphore.h"
#include <decoder.h>
#include <unordered_map>
```

Classes

- class **TraceThread**

Macros

- #define **NUM_PAPI_COUNTERS** 6
- #define **PAPI_TOT_INS** 0
- #define **PAPI_TOT_CYC** 1
- #define **PAPI_L1_DCM** 2
- #define **PAPI_L2_DCM** 3
- #define **PAPI_L3_TCM** 4
- #define **PAPI_BR_MSP** 5

7.633.1 Macro Definition Documentation

7.633.1.1 NUM_PAPI_COUNTERS

```
#define NUM_PAPI_COUNTERS 6
```

Definition at line 20 of file trace_thread.h.

7.633.1.2 PAPI_BR_MSP

```
#define PAPI_BR_MSP 5
```

Definition at line 27 of file trace_thread.h.

7.633.1.3 PAPI_L1_DCM

```
#define PAPI_L1_DCM 2
```

Definition at line 24 of file trace_thread.h.

7.633.1.4 PAPI_L2_DCM

```
#define PAPI_L2_DCM 3
```

Definition at line 25 of file trace_thread.h.

7.633.1.5 PAPI_L3_TCM

```
#define PAPI_L3_TCM 4
```

Definition at line 26 of file trace_thread.h.

7.633.1.6 PAPI_TOT_CYC

```
#define PAPI_TOT_CYC 1
```

Definition at line 23 of file trace_thread.h.

7.633.1.7 PAPI_TOT_INS

```
#define PAPI_TOT_INS 0
```

Definition at line 22 of file trace_thread.h.

7.634 common/transport/smtransport.cc File Reference

```
#include <string.h>
#include "smtransport.h"
#include "config.h"
#include "log.h"
```

7.635 common/transport/smtransport.d File Reference

7.636 common/transport/smtransport.h File Reference

```
#include <queue>
#include "transport.h"
#include "cond.h"
```

Classes

- class **SmTransport**
- class **SmTransport::SmNode**

7.637 common/transport/transport.cc File Reference

```
#include <assert.h>
#include "transport.h"
#include "smtransport.h"
#include "config.h"
#include "log.h"
```

Macros

- #define **TRANSPORT_CC**

7.637.1 Macro Definition Documentation

7.637.1.1 TRANSPORT_CC

```
#define TRANSPORT_CC
```

Definition at line 1 of file transport.cc.

7.638 common/transport/transport.d File Reference

7.639 common/transport/transport.h File Reference

```
#include "fixed_types.h"
#include <map>
```

Classes

- class **Transport**
- class **Transport::Node**

7.640 common/user/sync_api.cc File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
#include "simulator.h"
#include "thread.h"
#include "sync_client.h"
#include "config_file.hpp"
#include "subsecond_time.h"
#include "thread_manager.h"
```

Functions

- void **CarbonMutexInit** (**carbon_mutex_t** ∗mux)
- **SubsecondTime CarbonMutexLock** (**carbon_mutex_t** ∗mux, **SubsecondTime** delay)
- **SubsecondTime CarbonMutexTrylock** (**carbon_mutex_t** ∗mux)
- **SubsecondTime CarbonMutexUnlock** (**carbon_mutex_t** ∗mux, **SubsecondTime** delay)
- void **CarbonCondInit** (**carbon_cond_t** ∗cond)
- **SubsecondTime CarbonCondWait** (**carbon_cond_t** ∗cond, **carbon_mutex_t** ∗mux)
- **SubsecondTime CarbonCondSignal** (**carbon_cond_t** ∗cond)
- **SubsecondTime CarbonCondBroadcast** (**carbon_cond_t** ∗cond)
- void **CarbonBarrierInit** (**carbon_barrier_t** ∗barrier, **UInt32** count)
- **SubsecondTime CarbonBarrierWait** (**carbon_barrier_t** ∗barrier)

7.640.1 Function Documentation

7.640.1.1 CarbonBarrierInit()

```
void CarbonBarrierInit (
              carbon_barrier_t * barrier,
              UInt32 count )
```

Definition at line 66 of file sync_api.cc.

References SyncClient::barrierInit(), and Thread::getSyncClient().

Referenced by PthreadEmu::BarrierInit(), and lite::routineCallback().

7.640.1.2 CarbonBarrierWait()

```
SubsecondTime CarbonBarrierWait (
              carbon_barrier_t * barrier )
```

Definition at line 72 of file sync_api.cc.

References SyncClient::barrierWait(), and Thread::getSyncClient().

Referenced by PthreadEmu::BarrierWait(), and lite::routineCallback().

7.640.1.3 CarbonCondBroadcast()

```
SubsecondTime CarbonCondBroadcast (
              carbon_cond_t * cond )
```

Definition at line 60 of file sync_api.cc.

References SyncClient::condBroadcast(), and Thread::getSyncClient().

Referenced by PthreadEmu::CondBroadcast(), and lite::routineCallback().

7.640.1.4 CarbonCondInit()

```
void CarbonCondInit (
              carbon_cond_t * cond )
```

Definition at line 42 of file sync_api.cc.

References SyncClient::condInit(), and Thread::getSyncClient().

Referenced by PthreadEmu::CondInit(), and lite::routineCallback().

7.640.1.5 CarbonCondSignal()

```
SubsecondTime CarbonCondSignal (
              carbon_cond_t * cond )
```

Definition at line 54 of file sync_api.cc.

References SyncClient::condSignal(), and Thread::getSyncClient().

Referenced by PthreadEmu::CondSignal(), and lite::routineCallback().

7.640.1.6 CarbonCondWait()

```
SubsecondTime CarbonCondWait (
              carbon_cond_t * cond,
              carbon_mutex_t * mux )
```

Definition at line 48 of file sync_api.cc.

References SyncClient::condWait(), and Thread::getSyncClient().

Referenced by PthreadEmu::CondWait(), and lite::routineCallback().

7.640.1.7 CarbonMutexInit()

```
void CarbonMutexInit (
              carbon_mutex_t * mux )
```

Definition at line 11 of file sync_api.cc.

References Thread::getSyncClient(), and SyncClient::mutexInit().

Referenced by PthreadEmu::MutexInit(), and lite::routineCallback().

7.640.1.8 CarbonMutexLock()

```
SubsecondTime CarbonMutexLock (
              carbon_mutex_t * mux,
              SubsecondTime delay )
```

Definition at line 17 of file sync_api.cc.

References Thread::getSyncClient(), and SyncClient::mutexLock().

Referenced by PthreadEmu::MutexLock(), and lite::routineCallback().

7.640.1.9 CarbonMutexTrylock()

```
SubsecondTime CarbonMutexTrylock (
              carbon_mutex_t * mux )
```

Definition at line 26 of file sync_api.cc.

References Thread::getSyncClient(), SubsecondTime::MaxTime(), and SyncClient::mutexTrylock().

Referenced by PthreadEmu::MutexTrylock().

7.640.1.10 CarbonMutexUnlock()

```
SubsecondTime CarbonMutexUnlock (
              carbon_mutex_t * mux,
              SubsecondTime delay )
```

Definition at line 36 of file sync_api.cc.

References Thread::getSyncClient(), and SyncClient::mutexUnlock().

Referenced by PthreadEmu::MutexUnlock(), and lite::routineCallback().

7.641 common/user/sync_api.d File Reference

7.642 common/user/sync_api.h File Reference

```
#include <stdbool.h>
#include "subsecond_time.h"
#include "fixed_types.h"
```

Typedefs

- typedef **SInt32 carbon_mutex_t**
- typedef **SInt32 carbon_cond_t**
- typedef **SInt32 carbon_barrier_t**

Functions

- void **CarbonMutexInit** (**carbon_mutex_t** *mux)
- **SubsecondTime CarbonMutexLock** (**carbon_mutex_t** *mux, **SubsecondTime** delay= **SubsecondTime::Zero**())
- **SubsecondTime CarbonMutexTrylock** (**carbon_mutex_t** *mux)
- **SubsecondTime CarbonMutexUnlock** (**carbon_mutex_t** *mux, **SubsecondTime** delay= **SubsecondTime::Zero**())
- bool **CarbonIsMutexValid** (**carbon_mutex_t** *mux)
- void **CarbonCondInit** (**carbon_cond_t** *cond)
- **SubsecondTime CarbonCondWait** (**carbon_cond_t** *cond, **carbon_mutex_t** *mux)
- **SubsecondTime CarbonCondSignal** (**carbon_cond_t** *cond)
- **SubsecondTime CarbonCondBroadcast** (**carbon_cond_t** *cond)
- bool **CarbonIsCondValid** (**carbon_cond_t** *cond)
- void **CarbonBarrierInit** (**carbon_barrier_t** *barrier, unsigned int count)
- **SubsecondTime CarbonBarrierWait** (**carbon_barrier_t** *barrier)
- bool **CarbonIsBarrierValid** (**carbon_barrier_t** *barrier)

7.642.1 Typedef Documentation

7.642.1.1 carbon_barrier_t

```
typedef SInt32 carbon_barrier_t
```

Definition at line 14 of file sync_api.h.

7.642.1.2 carbon_cond_t

```
typedef SInt32 carbon_cond_t
```

Definition at line 13 of file sync_api.h.

7.642.1.3 carbon_mutex_t

```
typedef SInt32 carbon_mutex_t
```

Definition at line 12 of file sync_api.h.

7.642.2 Function Documentation

7.642.2.1 CarbonBarrierInit()

```
void CarbonBarrierInit (
              carbon_barrier_t * barrier,
             unsigned int count )
```

7.642.2.2 CarbonBarrierWait()

```
 SubsecondTime CarbonBarrierWait (
              carbon_barrier_t * barrier )
```

Definition at line 72 of file sync_api.cc.

References SyncClient::barrierWait(), and Thread::getSyncClient().

Referenced by PthreadEmu::BarrierWait(), and lite::routineCallback().

7.642.2.3 CarbonCondBroadcast()

```
SubsecondTime CarbonCondBroadcast (
              carbon_cond_t * cond )
```

Definition at line 60 of file sync_api.cc.

References SyncClient::condBroadcast(), and Thread::getSyncClient().

Referenced by PthreadEmu::CondBroadcast(), and lite::routineCallback().

7.642.2.4 CarbonCondInit()

```
void CarbonCondInit (
              carbon_cond_t * cond )
```

Definition at line 42 of file sync_api.cc.

References SyncClient::condInit(), and Thread::getSyncClient().

Referenced by PthreadEmu::CondInit(), and lite::routineCallback().

7.642.2.5 CarbonCondSignal()

```
SubsecondTime CarbonCondSignal (
              carbon_cond_t * cond )
```

Definition at line 54 of file sync_api.cc.

References SyncClient::condSignal(), and Thread::getSyncClient().

Referenced by PthreadEmu::CondSignal(), and lite::routineCallback().

7.642.2.6 CarbonCondWait()

```
SubsecondTime CarbonCondWait (
              carbon_cond_t * cond,
              carbon_mutex_t * mux )
```

Definition at line 48 of file sync_api.cc.

References SyncClient::condWait(), and Thread::getSyncClient().

Referenced by PthreadEmu::CondWait(), and lite::routineCallback().

7.642.2.7 CarbonIsBarrierValid()

```
bool CarbonIsBarrierValid (
              carbon_barrier_t * barrier )
```

7.642.2.8 CarbonIsCondValid()

```
bool CarbonIsCondValid (
              carbon_cond_t * cond )
```

7.642.2.9 CarbonIsMutexValid()

```
bool CarbonIsMutexValid (
              carbon_mutex_t * mux )
```

7.642.2.10 CarbonMutexInit()

```
void CarbonMutexInit (
              carbon_mutex_t * mux )
```

Definition at line 11 of file sync_api.cc.

References Thread::getSyncClient(), and SyncClient::mutexInit().

Referenced by PthreadEmu::MutexInit(), and lite::routineCallback().

7.642.2.11 CarbonMutexLock()

```
SubsecondTime CarbonMutexLock (
              carbon_mutex_t * mux,
              SubsecondTime delay =  SubsecondTime::Zero() )
```

Definition at line 17 of file sync_api.cc.

References Thread::getSyncClient(), and SyncClient::mutexLock().

Referenced by PthreadEmu::MutexLock(), and lite::routineCallback().

7.642.2.12 CarbonMutexTrylock()

```
SubsecondTime CarbonMutexTrylock (
            carbon_mutex_t * mux )
```

Definition at line 26 of file sync_api.cc.

References Thread::getSyncClient(), SubsecondTime::MaxTime(), and SyncClient::mutexTrylock().

Referenced by PthreadEmu::MutexTrylock().

7.642.2.13 CarbonMutexUnlock()

```
SubsecondTime CarbonMutexUnlock (
            carbon_mutex_t * mux,
            SubsecondTime delay =  SubsecondTime::Zero() )
```

Definition at line 36 of file sync_api.cc.

References Thread::getSyncClient(), and SyncClient::mutexUnlock().

Referenced by PthreadEmu::MutexUnlock(), and lite::routineCallback().

7.643 pin/codecache_trace.cc File Reference

```
#include "fixed_types.h"
#include "codecache_trace.h"
#include "timer.h"
#include "hooks_manager.h"
#include "subsecond_time.h"
#include "simulator.h"
#include "pin.H"
#include <inttypes.h>
#include <stdio.h>
#include <iostream>
```

Macros

- #define **__STDC_FORMAT_MACROS**
- #define **atomic_inc_int64**(a) __sync_fetch_and_add(&(a), 1)

Functions

- VOID **cacheInit** ()
- VOID **newCacheBlock** (USIZE new_block_size)
- VOID **fullCache** (USIZE trace_size, USIZE stub_size)
- VOID **cacheFlushed** ()
- VOID **codeCacheEntered** (ADDRINT cache_pc)
- VOID **codeCacheExited** (ADDRINT cache_pc)
- VOID **traceLinked** (ADDRINT branch_pc, ADDRINT target_pc)
- VOID **traceUnlinked** (ADDRINT branch_pc, ADDRINT stub_pc)
- VOID **traceInvalidated** (ADDRINT orig_pc, ADDRINT cache_pc, BOOL success)
- VOID **traceInserted** (TRACE trace, VOID ∗v)
- void **printCodeCacheTrace** (const **SubsecondTime** &time)
- void **printCodeCacheStats** (const **SubsecondTime** &time)
- void **codeCachePeriodicCallback** (void ∗self, **subsecond_time_t** _time)
- void **initCodeCacheTracing** ()

Variables

- uint64_t **cc_counters** [8]
- static FILE ∗ **cctrace**
- static FILE ∗ **ccstats**
- **SubsecondTime next_callback**

7.643.1 Macro Definition Documentation

7.643.1.1 __STDC_FORMAT_MACROS

```
#define __STDC_FORMAT_MACROS
```

Definition at line 10 of file codecache_trace.cc.

7.643.1.2 atomic_inc_int64

```
#define atomic_inc_int64(
              a ) __sync_fetch_and_add(&(a), 1)
```

Definition at line 20 of file codecache_trace.cc.

7.643.2 Function Documentation

7.643.2.1 cacheFlushed()

```
VOID cacheFlushed ( )
```

Definition at line 36 of file codecache_trace.cc.

References atomic_inc_int64, and cc_counters.

Referenced by initCodeCacheTracing().

7.643.2.2 cacheInit()

```
VOID cacheInit ( )
```

Definition at line 22 of file codecache_trace.cc.

7.643.2.3 codeCacheEntered()

```
VOID codeCacheEntered (
             ADDRINT cache_pc )
```

Definition at line 41 of file codecache_trace.cc.

References atomic_inc_int64, and cc_counters.

Referenced by initCodeCacheTracing().

7.643.2.4 codeCacheExited()

```
VOID codeCacheExited (
             ADDRINT cache_pc )
```

Definition at line 46 of file codecache_trace.cc.

7.643.2.5 codeCachePeriodicCallback()

```
void codeCachePeriodicCallback (
             void * self,
              subsecond_time_t _time )
```

Definition at line 95 of file codecache_trace.cc.

References next_callback, SubsecondTime::NS(), printCodeCacheStats(), and printCodeCacheTrace().

Referenced by initCodeCacheTracing().

7.643.2.6 fullCache()

```
VOID fullCache (
            USIZE trace_size,
            USIZE stub_size )
```

Definition at line 31 of file codecache_trace.cc.

References atomic_inc_int64, and cc_counters.

Referenced by initCodeCacheTracing().

7.643.2.7 initCodeCacheTracing()

```
void initCodeCacheTracing ( )
```

Definition at line 107 of file codecache_trace.cc.

References cacheFlushed(), ccstats, cctrace, codeCacheEntered(), codeCachePeriodicCallback(), fullCache(), HookType::HOOK_PERIODIC, LOG_ASSERT_ERROR, newCacheBlock(), next_callback, traceInserted(), traceInvalidated(), traceLinked(), traceUnlinked(), and SubsecondTime::Zero().

Referenced by main().

7.643.2.8 newCacheBlock()

```
VOID newCacheBlock (
            USIZE new_block_size )
```

Definition at line 26 of file codecache_trace.cc.

References atomic_inc_int64, and cc_counters.

Referenced by initCodeCacheTracing().

7.643.2.9 printCodeCacheStats()

```
void printCodeCacheStats (
            const  SubsecondTime & time )
```

Definition at line 80 of file codecache_trace.cc.

References ccstats, and SubsecondTime::getNS().

Referenced by codeCachePeriodicCallback().

7.643.2.10 printCodeCacheTrace()

```
void printCodeCacheTrace (
             const  SubsecondTime & time )
```

Definition at line 70 of file codecache_trace.cc.

References cc_counters, cctrace, and SubsecondTime::getNS().

Referenced by codeCachePeriodicCallback().

7.643.2.11 traceInserted()

```
VOID traceInserted (
             TRACE trace,
             VOID * v )
```

Definition at line 65 of file codecache_trace.cc.

References atomic_inc_int64, and cc_counters.

Referenced by initCodeCacheTracing().

7.643.2.12 traceInvalidated()

```
VOID traceInvalidated (
             ADDRINT orig_pc,
             ADDRINT cache_pc,
             BOOL success )
```

Definition at line 60 of file codecache_trace.cc.

References atomic_inc_int64, and cc_counters.

Referenced by initCodeCacheTracing().

7.643.2.13 traceLinked()

```
VOID traceLinked (
             ADDRINT branch_pc,
             ADDRINT target_pc )
```

Definition at line 50 of file codecache_trace.cc.

References atomic_inc_int64, and cc_counters.

Referenced by initCodeCacheTracing().

7.643.2.14 traceUnlinked()

```
VOID traceUnlinked (
            ADDRINT branch_pc,
            ADDRINT stub_pc )
```

Definition at line 55 of file codecache_trace.cc.

References atomic_inc_int64, and cc_counters.

Referenced by initCodeCacheTracing().

7.643.3 Variable Documentation

7.643.3.1 cc_counters

```
uint64_t cc_counters[8]
```

Definition at line 15 of file codecache_trace.cc.

Referenced by cacheFlushed(), codeCacheEntered(), fullCache(), newCacheBlock(), printCodeCacheTrace(), traceInserted(), traceInvalidated(), traceLinked(), and traceUnlinked().

7.643.3.2 ccstats

```
FILE* ccstats  [static]
```

Definition at line 17 of file codecache_trace.cc.

Referenced by initCodeCacheTracing(), and printCodeCacheStats().

7.643.3.3 cctrace

```
FILE* cctrace  [static]
```

Definition at line 16 of file codecache_trace.cc.

Referenced by initCodeCacheTracing(), and printCodeCacheTrace().

7.643.3.4 next_callback

```
SubsecondTime next_callback
```

Definition at line 18 of file codecache_trace.cc.

Referenced by codeCachePeriodicCallback(), and initCodeCacheTracing().

7.644 pin/codecache_trace.h File Reference

Functions

- void **initCodeCacheTracing** ()

7.644.1 Function Documentation

7.644.1.1 initCodeCacheTracing()

```
void initCodeCacheTracing ( )
```

Definition at line 107 of file codecache_trace.cc.

References cacheFlushed(), ccstats, cctrace, codeCacheEntered(), codeCachePeriodicCallback(), fullCache(), HookType::HOOK_PERIODIC, LOG_ASSERT_ERROR, newCacheBlock(), next_callback, traceInserted(), traceInvalidated(), traceLinked(), traceUnlinked(), and SubsecondTime::Zero().

Referenced by main().

7.645 pin/follow_execv/follow_execv.cc File Reference

```
#include "pin.H"
#include <cstring>
#include <cassert>
```

Functions

- static BOOL **followChild** (CHILD_PROCESS childProcess, VOID *val)
- int **main** (int argc, char *argv[])

Variables

- int **orig_argc**
- const char *const * **orig_argv**
- bool **have_followed** = false

7.645.1 Function Documentation

7.645.1.1 followChild()

```
static BOOL followChild (
            CHILD_PROCESS childProcess,
            VOID * val )  [static]
```

Definition at line 10 of file follow_execv.cc.

References have_followed, orig_argc, and orig_argv.

Referenced by main().

7.645.1.2 main()

```
int main (
            int argc,
            char * argv[] )
```

Definition at line 57 of file follow_execv.cc.

References followChild(), orig_argc, and orig_argv.

7.645.2 Variable Documentation

7.645.2.1 have_followed

```
bool have_followed = false
```

Definition at line 8 of file follow_execv.cc.

Referenced by followChild().

7.645.2.2 orig_argc

```
int orig_argc
```

Definition at line 6 of file follow_execv.cc.

Referenced by followChild(), and main().

7.645.2.3 orig_argv

```
const char* const * orig_argv
```

Definition at line 7 of file follow_execv.cc.

Referenced by followChild(), and main().

7.646 pin/inst_mode_macros.h File Reference

```
#include "pin.H"
```

Macros

- #define **INSTR_IF_DETAILED**(__inst_mode) ((__inst_mode) == **InstMode::DETAILED**)
- #define **INSTR_IF_NOT_DETAILED**(__inst_mode) ((__inst_mode) != **InstMode::DETAILED**)
- #define **INSTR_IF_CACHEONLY**(__inst_mode) ((__inst_mode) == **InstMode::CACHE_ONLY**)
- #define **INSTR_IF_NOT_CACHEONLY**(__inst_mode) ((__inst_mode) != **InstMode::CACHE_ONLY**)
- #define **INSTR_IF_FASTFORWARD**(__inst_mode) ((__inst_mode) == **InstMode::FAST_FORWARD**)
- #define **INSTR_IF_NOT_FASTFORWARD**(__inst_mode) ((__inst_mode) != **InstMode::FAST_FORWA↩RD**)
- #define **__INSTRUMENT**(predicated, condition, trace, ins, point, func, ...)
- #define **INSTR_GET_MODE**(__trace) ((**InstMode::inst_mode_t**)TRACE_Version(__trace))
- #define **INSTRUMENT**(...) **__INSTRUMENT**(, __VA_ARGS__)
- #define **INSTRUMENT_IF**(...) **__INSTRUMENT**(If, __VA_ARGS__)
- #define **INSTRUMENT_THEN**(...) **__INSTRUMENT**(Then, __VA_ARGS__)
- #define **INSTRUMENT_PREDICATED**(...) **__INSTRUMENT**(Predicated, __VA_ARGS__)
- #define **INSTRUMENT_IF_PREDICATED**(...) **__INSTRUMENT**(IfPredicated, __VA_ARGS__)
- #define **INSTRUMENT_THEN_PREDICATED**(...) **__INSTRUMENT**(ThenPredicated, __VA_ARGS__)

7.646.1 Macro Definition Documentation

7.646.1.1 __INSTRUMENT

```
#define __INSTRUMENT(
              predicated,
              condition,
              trace,
              ins,
              point,
              func,
              ...  )
```

Value:

```
   if (condition)                                                    \
      INS_Insert##predicated##Call(ins, point, func, __VA_ARGS__);   \
```

Definition at line 13 of file inst_mode_macros.h.

7.646.1.2 INSTR_GET_MODE

```
#define INSTR_GET_MODE(
              __trace ) (( InstMode::inst_mode_t)TRACE_Version(__trace))
```

Definition at line 17 of file inst_mode_macros.h.

7.646.1.3 INSTR_IF_CACHEONLY

```
#define INSTR_IF_CACHEONLY(
              __inst_mode ) ((__inst_mode) ==  InstMode::CACHE_ONLY)
```

Definition at line 8 of file inst_mode_macros.h.

7.646.1.4 INSTR_IF_DETAILED

```
#define INSTR_IF_DETAILED(
              __inst_mode ) ((__inst_mode) ==  InstMode::DETAILED)
```

Definition at line 6 of file inst_mode_macros.h.

7.646.1.5 INSTR_IF_FASTFORWARD

```
#define INSTR_IF_FASTFORWARD(
              __inst_mode ) ((__inst_mode) ==  InstMode::FAST_FORWARD)
```

Definition at line 10 of file inst_mode_macros.h.

7.646.1.6 INSTR_IF_NOT_CACHEONLY

```
#define INSTR_IF_NOT_CACHEONLY(
              __inst_mode ) ((__inst_mode) !=  InstMode::CACHE_ONLY)
```

Definition at line 9 of file inst_mode_macros.h.

7.646.1.7 INSTR_IF_NOT_DETAILED

```
#define INSTR_IF_NOT_DETAILED(
              __inst_mode ) ((__inst_mode) !=  InstMode::DETAILED)
```

Definition at line 7 of file inst_mode_macros.h.

7.646.1.8 INSTR_IF_NOT_FASTFORWARD

```
#define INSTR_IF_NOT_FASTFORWARD(
              __inst_mode ) ((__inst_mode) !=  InstMode::FAST_FORWARD)
```

Definition at line 11 of file inst_mode_macros.h.

7.646.1.9 INSTRUMENT

```
#define INSTRUMENT(
              ... )   __INSTRUMENT(, __VA_ARGS__)
```

Definition at line 19 of file inst_mode_macros.h.

7.646.1.10 INSTRUMENT_IF

```
#define INSTRUMENT_IF(
              ... )   __INSTRUMENT(If, __VA_ARGS__)
```

Definition at line 20 of file inst_mode_macros.h.

7.646.1.11 INSTRUMENT_IF_PREDICATED

```
#define INSTRUMENT_IF_PREDICATED(
              ... )   __INSTRUMENT(IfPredicated, __VA_ARGS__)
```

Definition at line 23 of file inst_mode_macros.h.

7.646.1.12 INSTRUMENT_PREDICATED

```
#define INSTRUMENT_PREDICATED(
              ... )   __INSTRUMENT(Predicated, __VA_ARGS__)
```

Definition at line 22 of file inst_mode_macros.h.

7.646.1.13 INSTRUMENT_THEN

```
#define INSTRUMENT_THEN(
              ... )   __INSTRUMENT(Then, __VA_ARGS__)
```

Definition at line 21 of file inst_mode_macros.h.

7.646.1.14 INSTRUMENT_THEN_PREDICATED

```
#define INSTRUMENT_THEN_PREDICATED(
             ... )   __INSTRUMENT(ThenPredicated, __VA_ARGS__)
```

Definition at line 24 of file inst_mode_macros.h.

7.647 pin/instruction_modeling.cc File Reference

```
#include "instruction_modeling.h"
#include "inst_mode_macros.h"
#include "local_storage.h"
#include "spin_loop_detection.h"
#include "lite/memory_modeling.h"
#include "lite/handle_syscalls.h"
#include "simulator.h"
#include "performance_model.h"
#include "core_manager.h"
#include "core.h"
#include "thread.h"
#include "timer.h"
#include "instruction_decoder.h"
#include "instruction.h"
#include "dynamic_instruction.h"
#include "micro_op.h"
#include "magic_client.h"
#include "inst_mode.h"
#include "dvfs_manager.h"
#include "hooks_manager.h"
#include "branch_predictor.h"
#include <unordered_map>
```

Functions

- static void **handleBranch** (THREADID thread_id, ADDRINT eip, BOOL taken, ADDRINT target)
- static void **handleBranchWarming** (THREADID thread_id, ADDRINT eip, BOOL taken, ADDRINT target)
- static ADDRINT **handleMagic** (THREADID threadIndex, ADDRINT a, ADDRINT b, ADDRINT c)
- static void **handleRdtsc** (THREADID thread_id, PIN_REGISTER *gax, PIN_REGISTER *gdx)
- static void **handleCpuid** (THREADID thread_id, PIN_REGISTER *gax, PIN_REGISTER *gbx, PIN_REG↩ ISTER *gcx, PIN_REGISTER *gdx)
- static void **handlePause** ()
- static void **fillOperandListMemOps** (**OperandList** *list, INS ins)
- static void **fillOperandList** (**OperandList** *list, INS ins)

Variables

- std::unordered_map< ADDRINT, const std::vector< const **MicroOp** * > * > **instruction_cache**

7.647.1 Function Documentation

7.647.1.1 fillOperandList()

```
static void fillOperandList (
              OperandList * list,
             INS ins )  [static]
```

Definition at line 137 of file instruction_modeling.cc.

References fillOperandListMemOps(), Operand::IMMEDIATE, Operand::READ, Operand::REG, and Operand::WRITE.

Referenced by InstructionModeling::decodeInstruction().

7.647.1.2 fillOperandListMemOps()

```
static void fillOperandListMemOps (
              OperandList * list,
             INS ins )  [static]
```

Definition at line 118 of file instruction_modeling.cc.

References Operand::MEMORY, Operand::READ, and Operand::WRITE.

Referenced by fillOperandList().

7.647.1.3 handleBranch()

```
static void handleBranch (
             THREADID thread_id,
             ADDRINT eip,
             BOOL taken,
             ADDRINT target )  [static]
```

Definition at line 62 of file instruction_modeling.cc.

References localStore.

Referenced by InstructionModeling::addInstructionModeling().

7.647.1.4 handleBranchWarming()

```
static void handleBranchWarming (
             THREADID thread_id,
             ADDRINT eip,
             BOOL taken,
             ADDRINT target )  [static]
```

Definition at line 68 of file instruction_modeling.cc.

References Core::accessBranchPredictor(), Core::getPerformanceModel(), PerformanceModel::handleBranchMispredict(), and localStore.

Referenced by InstructionModeling::addInstructionModeling().

7.647.1.5 handleCpuid()

```
static void handleCpuid (
             THREADID thread_id,
             PIN_REGISTER * gax,
             PIN_REGISTER * gbx,
             PIN_REGISTER * gcx,
             PIN_REGISTER * gdx )  [static]
```

Definition at line 97 of file instruction_modeling.cc.

References cpuid_result_t::eax, cpuid_result_t::ebx, cpuid_result_t::ecx, cpuid_result_t::edx, Core::emulate↩ Cpuid(), and localStore.

Referenced by InstructionModeling::addInstructionModeling().

7.647.1.6 handleMagic()

```
static ADDRINT handleMagic (
             THREADID threadIndex,
             ADDRINT a,
             ADDRINT b,
             ADDRINT c )  [static]
```

Definition at line 79 of file instruction_modeling.cc.

References handleMagicInstruction(), and localStore.

Referenced by InstructionModeling::addInstructionModeling().

7.647.1.7 handlePause()

```
static void handlePause ( )  [static]
```

Definition at line 111 of file instruction_modeling.cc.

Referenced by InstructionModeling::addInstructionModeling().

7.647.1.8 handleRdtsc()

```
static void handleRdtsc (
             THREADID thread_id,
             PIN_REGISTER * gax,
             PIN_REGISTER * gdx )  [static]
```

Definition at line 84 of file instruction_modeling.cc.

References SubsecondTime::divideRounded(), PerformanceModel::getElapsedTime(), Core::getPerformance↩ Model(), and localStore.

Referenced by InstructionModeling::addInstructionModeling().

7.647.2 Variable Documentation

7.647.2.1 instruction_cache

```
std::unordered_map<ADDRINT, const std::vector<const  MicroOp *> *> instruction_cache
```

Definition at line 170 of file instruction_modeling.cc.

7.648 pin/instruction_modeling.h File Reference

```
#include "fixed_types.h"
#include "inst_mode.h"
#include <pin.H>
```

Classes

- class **InstructionModeling**

7.649 pin/lite/handle_syscalls.cc File Reference

```
#include "lite/handle_syscalls.h"
#include "simulator.h"
#include "thread_manager.h"
#include "thread.h"
#include "syscall_model.h"
#include "performance_model.h"
#include "log.h"
#include "syscall_strings.h"
#include "local_storage.h"
#include <syscall.h>
#include <stdlib.h>
```

Namespaces

- **lite**

Functions

- void **lite::handleSyscall** (THREADID threadIndex, CONTEXT *ctx)
- void **lite::syscallEnterRunModel** (THREADID threadIndex, CONTEXT *ctx, SYSCALL_STANDARD syscall_standard, void *v)
- void **lite::syscallExitRunModel** (THREADID threadIndex, CONTEXT *ctx, SYSCALL_STANDARD syscall_standard, void *v)
- bool **lite::interceptSignal** (THREADID threadIndex, INT32 signal, CONTEXT *ctx, BOOL hasHandler, const EXCEPTION_INFO *pExceptInfo, void *v)

7.650 pin/lite/handle_syscalls.h File Reference

```
#include "fixed_types.h"
#include "pin.H"
```

Namespaces

- **lite**

Functions

- void **lite::handleSyscall** (THREADID threadIndex, CONTEXT ∗ctx)
- void **lite::syscallEnterRunModel** (THREADID threadIndex, CONTEXT ∗ctx, SYSCALL_STANDARD syscall_standard, void ∗v)
- void **lite::syscallExitRunModel** (THREADID threadIndex, CONTEXT ∗ctx, SYSCALL_STANDARD syscall_standard, void ∗v)
- bool **lite::interceptSignal** (THREADID threadIndex, INT32 signal, CONTEXT ∗ctx, BOOL hasHandler, const EXCEPTION_INFO ∗pExceptInfo, void ∗v)

7.651 pin/lite/memory_modeling.cc File Reference

```
#include "lite/memory_modeling.h"
#include "simulator.h"
#include "performance_model.h"
#include "core_manager.h"
#include "core.h"
#include "thread.h"
#include "inst_mode.h"
#include "instruction_modeling.h"
#include "dynamic_instruction.h"
#include "fault_injection.h"
#include "inst_mode_macros.h"
#include "local_storage.h"
#include "toolreg.h"
```

Namespaces

- **lite**

Functions

- void **lite::addMemoryModeling** (TRACE trace, INS ins, **InstMode::inst_mode_t** inst_mode)
- void **lite::handleMemoryRead** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_↩ update, **IntPtr** read_address, **UInt32** read_data_size)
- void **lite::handleMemoryReadDetailed** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_↩ atomic_update, **IntPtr** read_address, **UInt32** read_data_size)
- void **lite::handleMemoryReadDetailedIssue** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_update, **IntPtr** read_address, **UInt32** read_data_size)
- ADDRINT **lite::handleMemoryReadFaultinjectionNondetailed** (bool is_atomic_update, ADDRINT read↩ _address, ADDRINT ∗save_ea)
- ADDRINT **lite::handleMemoryReadFaultinjection** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_update, ADDRINT read_address, **UInt32** read_data_size, **UInt32** op_num, ADDRI↩ NT ∗save_ea)
- void **lite::completeMemoryWrite** (bool is_atomic_update, ADDRINT write_address, ADDRINT scratch, U↩ INT32 write_size)
- void **lite::handleMemoryWrite** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_↩ update, **IntPtr** write_address, **UInt32** write_data_size)
- void **lite::handleMemoryWriteDetailed** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_↩ atomic_update, **IntPtr** write_address, **UInt32** write_data_size)
- void **lite::handleMemoryWriteDetailedIssue** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_update, **IntPtr** write_address, **UInt32** write_data_size)
- void **lite::handleMemoryWriteFaultinjection** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_update, **IntPtr** write_address, **UInt32** write_data_size)

Variables

- **Lock lite::g_atomic_lock**
- char **lite::g_zeros** [1024] = { 0 }

7.652 pin/lite/memory_modeling.h File Reference

```
#include "fixed_types.h"
#include "inst_mode.h"
#include "pin.H"
```

Namespaces

- **lite**

Functions

- void **lite::addMemoryModeling** (TRACE trace, INS ins, **InstMode::inst_mode_t** inst_mode)
- void **lite::handleMemoryRead** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_↩ update, **IntPtr** read_address, **UInt32** read_data_size)
- void **lite::handleMemoryReadDetailed** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_↩ atomic_update, **IntPtr** read_address, **UInt32** read_data_size)
- void **lite::handleMemoryReadDetailedIssue** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_update, **IntPtr** read_address, **UInt32** read_data_size)

- ADDRINT **lite::handleMemoryReadFaultinjectionNondetailed** (bool is_atomic_update, ADDRINT read↩ _address, ADDRINT ∗save_ea)
- ADDRINT **lite::handleMemoryReadFaultinjection** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_update, ADDRINT read_address, **UInt32** read_data_size, **UInt32** op_num, ADDRI↩ NT ∗save_ea)
- void **lite::completeMemoryWrite** (bool is_atomic_update, ADDRINT write_address, ADDRINT scratch, U↩ INT32 write_size)
- void **lite::handleMemoryWrite** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_↩ update, **IntPtr** write_address, **UInt32** write_data_size)
- void **lite::handleMemoryWriteDetailed** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_↩ atomic_update, **IntPtr** write_address, **UInt32** write_data_size)
- void **lite::handleMemoryWriteDetailedIssue** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_update, **IntPtr** write_address, **UInt32** write_data_size)
- void **lite::handleMemoryWriteFaultinjection** (THREADID thread_id, BOOL executing, ADDRINT eip, bool is_atomic_update, **IntPtr** write_address, **UInt32** write_data_size)

7.653 pin/lite/routine_replace.cc File Reference

```
#include "lite/routine_replace.h"
#include "instruction_modeling.h"
#include "pthread_emu.h"
#include "simulator.h"
#include "sync_api.h"
#include "performance_model.h"
#include "thread_manager.h"
#include "core_manager.h"
#include "core.h"
#include "thread.h"
#include "log.h"
#include "network.h"
#include "packet_type.h"
#include "magic_client.h"
#include "local_storage.h"
#include "trace_rtn.h"
#include "memory_tracker.h"
#include <map>
#include <cerrno>
```

Classes

- struct **lite::pthread_functions_t**

Namespaces

- **lite**

Functions

- void **lite::printStackTrace** (THREADID threadid, char *function, BOOL enter)
- void **lite::routineStartCallback** (RTN rtn, INS ins)
- void **lite::routineCallback** (RTN rtn, void *v)
- AFUNPTR **lite::getFunptr** (CONTEXT *context, string func_name)
- **IntPtr** **lite::nullFunction** ()
- void **lite::pthreadBefore** (THREADID thread_id)
- void **lite::pthreadAfter** (THREADID thread_id, ADDRINT type_id, ADDRINT retval)
- void **lite::mallocBefore** (THREADID thread_id, ADDRINT eip, ADDRINT size)
- void **lite::mallocAfter** (THREADID thread_id, ADDRINT address)
- void **lite::freeBefore** (THREADID thread_id, ADDRINT eip, ADDRINT address)
- **IntPtr** **lite::emuGetNprocs** ()
- **IntPtr** **lite::emuGetCPU** (THREADID thread_id)
- **IntPtr** **lite::emuClockGettime** (THREADID thread_id, clockid_t clk_id, struct timespec *tp)
- **IntPtr** **lite::emuGettimeofday** (THREADID thread_id, struct timeval *tv, struct timezone *tz)
- void **lite::emuKmpReapMonitor** (THREADID threadIndex, CONTEXT *ctxt)

Variables

- std::unordered_map< **core_id_t**, **SubsecondTime** > **lite::pthread_t_start**
- AFUNPTR **lite::ptr_exit** = NULL
- struct **lite::pthread_functions_t** **lite::pthread_functions** [ ]

7.654 pin/lite/routine_replace.h File Reference

```
#include "fixed_types.h"
#include "pin.H"
#include <pthread.h>
#include <sys/time.h>
```

Namespaces

- **lite**

Functions

- void **lite::routineCallback** (RTN rtn, void *v)
- void **lite::routineStartCallback** (RTN rtn, INS ins)
- **IntPtr** **lite::nullFunction** ()
- void **lite::pthreadBefore** (THREADID thread_id)
- void **lite::pthreadAfter** (THREADID thread_id, ADDRINT type_id, ADDRINT retval)
- void **lite::mallocBefore** (THREADID thread_id, ADDRINT eip, ADDRINT size)
- void **lite::mallocAfter** (THREADID thread_id, ADDRINT address)
- void **lite::freeBefore** (THREADID thread_id, ADDRINT eip, ADDRINT address)
- **IntPtr** **lite::emuGetNprocs** ()
- **IntPtr** **lite::emuGetCPU** (THREADID thread_id)
- **IntPtr** **lite::emuClockGettime** (THREADID thread_id, clockid_t clk_id, struct timespec *tp)
- **IntPtr** **lite::emuGettimeofday** (THREADID thread_id, struct timeval *tv, struct timezone *tz)
- void **lite::emuKmpReapMonitor** (THREADID threadIndex, CONTEXT *ctxt)
- AFUNPTR **lite::getFunptr** (CONTEXT *context, string func_name)

7.655 pin/local_storage.cc File Reference

```
#include "local_storage.h"
```

Functions

- std::vector< **ThreadLocalStorage** > **localStore** (**MAX_PIN_THREADS**)

7.655.1 Function Documentation

7.655.1.1 localStore()

```
std::vector< ThreadLocalStorage> localStore (
             MAX_PIN_THREADS  )
```

7.656 pin/local_storage.h File Reference

```
#include "inst_mode.h"
#include "spin_loop_detection.h"
#include <vector>
```

Classes

- struct **ThreadLocalStorage**

Macros

- #define **MAX_PIN_THREADS** 2048

Variables

- std::vector< **ThreadLocalStorage** > **localStore**

7.656.1 Macro Definition Documentation

7.656.1.1 MAX_PIN_THREADS

```
#define MAX_PIN_THREADS 2048
```

Definition at line 4 of file local_storage.h.

7.656.2 Variable Documentation

7.656.2.1 localStore

```
std::vector< ThreadLocalStorage> localStore
```

Referenced by InstructionModeling::accessInstructionCacheWarmup(), InstructionModeling::countInstructions(), lite::emuClockGettime(), lite::emuGetCPU(), lite::emuGettimeofday(), lite::freeBefore(), InstructionModeling::handleBasicBlock(), handleBranch(), handleBranchWarming(), handleCheckScheduled(), handleCpuid(), InstructionModeling::handleInstruction(), handleMagic(), lite::handleMemoryRead(), lite::handleMemoryReadDetailed(), lite::handleMemoryReadDetailedIssue(), lite::handleMemoryReadFaultinjection(), lite::handleMemoryWrite(), lite::handleMemoryWriteDetailed(), lite::handleMemoryWriteDetailedIssue(), lite::handleMemoryWriteFaultinjection(), handleRdtsc(), lite::handleSyscall(), lite::mallocAfter(), lite::mallocBefore(), lite::pthreadAfter(), lite::pthreadBefore(), routineAssert(), routineCallSite(), routineEnter(), routineExit(), spinloopHandleBranch(), spinloopHandleRegWriteAfter(), spinloopHandleRegWriteBefore(), spinloopHandleWriteAfter(), spinloopHandleWriteBefore(), spinloopIfInSpin(), lite::syscallEnterRunModel(), lite::syscallExitRunModel(), threadFiniCallback(), and threadStartCallback().

7.657 pin/pin_exceptions.cc File Reference

```
#include "pin_exceptions.h"
#include "simulator.h"
#include "config.h"
#include "timer.h"
#include "callstack.h"
```

Functions

- EXCEPT_HANDLING_RESULT **exceptionHandler** (THREADID tid, EXCEPTION_INFO *pExceptInfo, PHYSICAL_CONTEXT *pPhysCtxt, VOID *v)

7.657.1 Function Documentation

7.657.1.1 exceptionHandler()

```
EXCEPT_HANDLING_RESULT exceptionHandler (
            THREADID tid,
            EXCEPTION_INFO * pExceptInfo,
            PHYSICAL_CONTEXT * pPhysCtxt,
            VOID * v )
```

Definition at line 7 of file pin_exceptions.cc.

References get_call_stack_from(), and rdtsc().

Referenced by main().

7.658 pin/pin_exceptions.h File Reference

```
#include "fixed_types.h"
#include "pin.H"
```

Functions

- EXCEPT_HANDLING_RESULT **exceptionHandler** (THREADID tid, EXCEPTION_INFO ∗pExceptInfo, P↩ HYSICAL_CONTEXT ∗pPhysCtxt, VOID ∗v)

7.658.1 Function Documentation

7.658.1.1 exceptionHandler()

```
EXCEPT_HANDLING_RESULT exceptionHandler (
            THREADID tid,
            EXCEPTION_INFO * pExceptInfo,
            PHYSICAL_CONTEXT * pPhysCtxt,
            VOID * v )
```

Definition at line 7 of file pin_exceptions.cc.

References get_call_stack_from(), and rdtsc().

Referenced by main().

7.659 pin/pin_lock.cc File Reference

```
#include "pin_lock.h"
```

7.660 pin/pin_lock.h File Reference

```
#include "lock.h"
#include "pin.H"
```

Classes

- class **PinLock**

7.661 pin/pin_sim.cc File Reference

```
#include "simulator.h"
#include "core_manager.h"
#include "config.h"
#include "core.h"
#include "thread.h"
#include "syscall_model.h"
#include "thread_manager.h"
#include "hooks_manager.h"
#include "trace_manager.h"
#include "config_file.hpp"
#include "handle_args.h"
#include "log.h"
#include "instruction_modeling.h"
#include "instruction.h"
#include "magic_client.h"
#include "sampling_manager.h"
#include "sampling_provider.h"
#include "performance_model.h"
#include "timer.h"
#include "logmem.h"
#include "circular_log.h"
#include "codecache_trace.h"
#include "local_storage.h"
#include "toolreg.h"
#include "pin_exceptions.h"
#include "trace_rtn.h"
#include "lite/routine_replace.h"
#include "lite/handle_syscalls.h"
#include "sim_api.h"
#include "pin.H"
#include "inst_mode_macros.h"
#include <iostream>
#include <assert.h>
#include <set>
#include <sys/syscall.h>
#include <unistd.h>
#include <signal.h>
#include <string.h>
#include <typeinfo>
```

Macros

- #define **SWITCH_VERSION**(v) if (inst_mode != (v)) INS_InsertVersionCase(ins_head, **g_toolregs**[**TOOLREG_TEMP**], v, v, IARG_CALL_ORDER, call_order, IARG_END);

Functions

- void **applicationMemCopy** (void *dest, const void * **src**, size_t **n**)
- void **printRtn** (ADDRINT rtn_addr, bool enter)
- VOID **printInsInfo** (CONTEXT *ctxt)
- void **showInstructionInfo** (INS ins)
- VOID **instructionCallback** (TRACE trace, INS ins, **InstMode::inst_mode_t** inst_mode)
- void **handleCheckScheduled** (THREADID threadIndex)
- VOID **addCheckScheduled** (TRACE trace, INS ins_head)
- static int **getInstMode** ()
- VOID **traceCallback** (TRACE trace, void *v)
- VOID **traceInvalidate** (ADDRINT orig_pc, ADDRINT cache_pc, BOOL success)
- void **ApplicationStart** ()
- void **ApplicationExit** (int, void *)
- void **ApplicationDetach** (void *)
- void **PinDetach** (void)
- VOID **threadStartCallback** (THREADID threadIndex, CONTEXT *ctxt, INT32 flags, VOID *v)
- VOID **threadFiniCallback** (THREADID threadIndex, const CONTEXT *ctxt, INT32 flags, VOID *v)
- bool **handleSigUsr1** (THREADID threadIndex, INT32 signal, CONTEXT *ctx, BOOL hasHandler, const EXCEPTION_INFO *pExceptInfo, void *v)
- VOID **forkAfterChild** (THREADID threadid, const CONTEXT *ctxt, VOID *v)
- int **main** (int argc, char *argv[])

Variables

- bool **done_app_initialization** = false
- **config::ConfigFile** * **cfg**
- int * **parent_tidptr**
- int * **child_tidptr**
- bool **forkedInChild** = false
- map< ADDRINT, string > **rtn_map**
- PIN_LOCK **rtn_map_lock**

7.661.1 Macro Definition Documentation

7.661.1.1 SWITCH_VERSION

```
#define SWITCH_VERSION(
              v ) if (inst_mode != (v)) INS_InsertVersionCase(ins_head,  g_toolregs[ TOOLREG_TEMP], v, v, IARG_CALL_ORDER, call_order, IARG_END);
```

7.661.2 Function Documentation

7.661.2.1 addCheckScheduled()

```
VOID addCheckScheduled (
             TRACE trace,
             INS ins_head )
```

Definition at line 178 of file pin_sim.cc.

References handleCheckScheduled().

Referenced by traceCallback().

7.661.2.2 ApplicationDetach()

```
void ApplicationDetach (
             void *  )
```

Definition at line 320 of file pin_sim.cc.

References ApplicationExit().

Referenced by main().

7.661.2.3 ApplicationExit()

```
void ApplicationExit (
             int ,
             void *  )
```

Definition at line 309 of file pin_sim.cc.

References cfg, forkedInChild, LOG_PRINT, and Simulator::release().

Referenced by ApplicationDetach(), and main().

7.661.2.4 applicationMemCopy()

```
void applicationMemCopy (
             void * dest,
             const void * src,
             size_t n )
```

Definition at line 96 of file pin_sim.cc.

References n, and src.

Referenced by Core::nativeMemOp().

7.661.2.5 ApplicationStart()

```
void ApplicationStart ( )
```

Definition at line 305 of file pin_sim.cc.

7.661.2.6 forkAfterChild()

```
VOID forkAfterChild (
             THREADID threadid,
             const CONTEXT * ctxt,
             VOID * v )
```

Definition at line 388 of file pin_sim.cc.

References forkedInChild.

Referenced by main().

7.661.2.7 getInstMode()

```
static int getInstMode ( )  [static]
```

Definition at line 183 of file pin_sim.cc.

Referenced by traceCallback().

7.661.2.8 handleCheckScheduled()

```
void handleCheckScheduled (
            THREADID threadIndex )
```

Definition at line 157 of file pin_sim.cc.

References Thread::getCore(), PerformanceModel::getElapsedTime(), Core::getPerformanceModel(), localStore, PerformanceModel::queuePseudoInstruction(), Thread::reschedule(), and SubsecondTime::Zero().

Referenced by addCheckScheduled().

7.661.2.9 handleSigUsr1()

```
bool handleSigUsr1 (
            THREADID threadIndex,
            INT32 signal,
            CONTEXT * ctx,
            BOOL hasHandler,
            const EXCEPTION_INFO * pExceptInfo,
            void * v )
```

Definition at line 382 of file pin_sim.cc.

References HookType::HOOK_SIGUSR1.

Referenced by main().

7.661.2.10 instructionCallback()

```
VOID instructionCallback (
            TRACE trace,
            INS ins,
             InstMode::inst_mode_t inst_mode )
```

Definition at line 140 of file pin_sim.cc.

References Log::getSingleton(), and printInsInfo().

7.661.2.11 main()

```
int main (
            int argc,
            char * argv[] )
```

Definition at line 395 of file pin_sim.cc.

References Simulator::allocate(), ApplicationDetach(), ApplicationExit(), cfg, exceptionHandler(), forkAfterChild(), config::Config::getBool(), config::Config::getString(), handle_args(), handleSigUsr1(), initCodeCacheTracing(), initToolregs(), lite::interceptSignal(), config::Config::load(), LOG_ASSERT_ERROR, LOG_PRINT, LOG_PRINT_ERROR, parse_args(), Config::PINTOOL, lite::routineCallback(), rtn_map_lock, Simulator::setConfig(), lite::syscallEnterRunModel(), lite::syscallExitRunModel(), threadFiniCallback(), threadStartCallback(), traceCallback(), and traceInvalidate().

7.661.2.12 PinDetach()

```
void PinDetach (
             void  )
```

Definition at line 325 of file pin_sim.cc.

Referenced by MagicServer::disablePerformance().

7.661.2.13 printInsInfo()

```
VOID printInsInfo (
             CONTEXT * ctxt )
```

Definition at line 119 of file pin_sim.cc.

References __attribute__, and LOG_PRINT.

Referenced by instructionCallback().

7.661.2.14 printRtn()

```
void printRtn (
             ADDRINT rtn_addr,
             bool enter )
```

Definition at line 101 of file pin_sim.cc.

References LOG_PRINT, rtn_map, and rtn_map_lock.

7.661.2.15 showInstructionInfo()

```
void showInstructionInfo (
             INS ins )
```

Definition at line 127 of file pin_sim.cc.

7.661.2.16 threadFiniCallback()

```
VOID threadFiniCallback (
             THREADID threadIndex,
             const CONTEXT * ctxt,
             INT32 flags,
             VOID * v )
```

Definition at line 370 of file pin_sim.cc.

References forkedInChild, and localStore.

Referenced by main().

7.661.2.17 threadStartCallback()

```
VOID threadStartCallback (
             THREADID threadIndex,
             CONTEXT * ctxt,
             INT32 flags,
             VOID * v )
```

Definition at line 330 of file pin_sim.cc.

References done_app_initialization, INVALID_THREAD_ID, itostr(), localStore, LOG_ASSERT_ERROR, Thread↩ LocalStorage::NUM_SCRATCHPADS, and ThreadLocalStorage::SCRATCHPAD_SIZE.

Referenced by main().

7.661.2.18 traceCallback()

```
VOID traceCallback (
             TRACE trace,
             void * v )
```

Definition at line 188 of file pin_sim.cc.

References InstructionModeling::accessInstructionCacheWarmup(), addCheckScheduled(), InstructionModeling↩ ::addInstructionModeling(), addRtnTracer(), InstMode::CACHE_ONLY, InstructionModeling::countInstructions(), InstMode::DETAILED, InstMode::FAST_FORWARD, g_toolregs, getInstMode(), InstructionModeling::handle↩ BasicBlock(), InstrumentLevel::INSTR, INSTR_GET_MODE, INSTR_IF_CACHEONLY, INSTR_IF_DETAILED, InstrumentLevel::INSTR_WITH_BBVS, INSTRUMENT, LOG_PRINT_ERROR, InstrumentLevel::NONE, lite↩ ::routineStartCallback(), SWITCH_VERSION, and TOOLREG_TEMP.

Referenced by main().

7.661.2.19 traceInvalidate()

```
VOID traceInvalidate (
            ADDRINT orig_pc,
            ADDRINT cache_pc,
            BOOL success )
```

Definition at line 291 of file pin_sim.cc.

References LOG_PRINT_WARNING_ONCE.

Referenced by main().

7.661.3 Variable Documentation

7.661.3.1 cfg

```
config::ConfigFile* cfg
```

Definition at line 76 of file pin_sim.cc.

Referenced by ApplicationExit(), BranchPredictor::create(), MemoryManagerBase::getCoreListWithMemory↩Controllers(), Config::getNearestAcceptableCoreCount(), Config::getNetworkModels(), handle_args(), handle_↩generic_arg(), config::ConfigFile::loadConfigFromString(), main(), and Simulator::setConfig().

7.661.3.2 child_tidptr

```
int* child_tidptr
```

7.661.3.3 done_app_initialization

```
bool done_app_initialization = false
```

Definition at line 75 of file pin_sim.cc.

Referenced by threadStartCallback().

7.661.3.4 forkedInChild

```
bool forkedInChild = false
```

Definition at line 85 of file pin_sim.cc.

Referenced by ApplicationExit(), forkAfterChild(), and threadFiniCallback().

7.661.3.5 parent_tidptr

```
int* parent_tidptr
```

7.661.3.6 rtn_map

```
map<ADDRINT, string> rtn_map
```

Definition at line 89 of file pin_sim.cc.

Referenced by printRtn().

7.661.3.7 rtn_map_lock

```
PIN_LOCK rtn_map_lock
```

Definition at line 90 of file pin_sim.cc.

Referenced by main(), and printRtn().

7.662 pin/pin_thread.cc File Reference

```
#include "pin_thread.h"
#include <assert.h>
```

7.663 pin/pin_thread.h File Reference

```
#include "_thread.h"
#include "pin.H"
```

Classes

- class **PinThread**

7.664 pin/pin_tls.cc File Reference

```
#include "tls.h"
#include "log.h"
#include <pin.H>
```

Classes

- class **PinTLS**

7.665 pin/spin_loop_detection.cc File Reference

```
#include "spin_loop_detection.h"
#include "inst_mode_macros.h"
#include "local_storage.h"
#include "simulator.h"
#include "thread.h"
#include <algorithm>
```

Functions

- static ADDRINT **spinloopIfInSpin** (THREADID thread_id)
- static void **spinloopHandleBranch** (THREADID thread_id, ADDRINT eip, BOOL taken, ADDRINT target)
- static void **spinloopHandleWriteBefore** (THREADID thread_id, ADDRINT addr)
- static void **spinloopHandleWriteAfter** (THREADID thread_id)
- static void **spinloopHandleRegWriteBefore** (THREADID thread_id, UINT32 reg, ADDRINT value)
- static void **spinloopHandleRegWriteAfter** (THREADID thread_id, UINT32 _reg, ADDRINT value)
- void **addSpinLoopDetection** (TRACE trace, INS ins, **InstMode::inst_mode_t** inst_mode)

7.665.1 Function Documentation

7.665.1.1 addSpinLoopDetection()

```
void addSpinLoopDetection (
            TRACE trace,
            INS ins,
             InstMode::inst_mode_t inst_mode )
```

Definition at line 59 of file spin_loop_detection.cc.

References INSTR_IF_DETAILED, INSTRUMENT_IF_PREDICATED, INSTRUMENT_PREDICATED, INSTRUM↩ ENT_THEN_PREDICATED, spinloopHandleBranch(), spinloopHandleRegWriteAfter(), spinloopHandleRegWrite↩ Before(), spinloopHandleWriteAfter(), spinloopHandleWriteBefore(), and spinloopIfInSpin().

Referenced by InstructionModeling::addInstructionModeling().

7.665.1.2 spinloopHandleBranch()

```
static void spinloopHandleBranch (
            THREADID thread_id,
            ADDRINT eip,
            BOOL taken,
            ADDRINT target )  [static]
```

Definition at line 18 of file spin_loop_detection.cc.

References localStore.

Referenced by addSpinLoopDetection().

7.665.1.3 spinloopHandleRegWriteAfter()

```
static void spinloopHandleRegWriteAfter (
            THREADID thread_id,
            UINT32 _reg,
            ADDRINT value )  [static]
```

Definition at line 51 of file spin_loop_detection.cc.

References localStore.

Referenced by addSpinLoopDetection().

7.665.1.4 spinloopHandleRegWriteBefore()

```
static void spinloopHandleRegWriteBefore (
            THREADID thread_id,
            UINT32 reg,
            ADDRINT value )  [static]
```

Definition at line 45 of file spin_loop_detection.cc.

References localStore.

Referenced by addSpinLoopDetection().

7.665.1.5 spinloopHandleWriteAfter()

```
static void spinloopHandleWriteAfter (
            THREADID thread_id )  [static]
```

Definition at line 33 of file spin_loop_detection.cc.

References localStore.

Referenced by addSpinLoopDetection().

7.665.1.6 spinloopHandleWriteBefore()

```
static void spinloopHandleWriteBefore (
            THREADID thread_id,
            ADDRINT addr )  [static]
```

Definition at line 26 of file spin_loop_detection.cc.

References localStore.

Referenced by addSpinLoopDetection().

7.665.1.7 spinloopIfInSpin()

```
static ADDRINT spinloopIfInSpin (
            THREADID thread_id )  [static]
```

Definition at line 12 of file spin_loop_detection.cc.

References localStore.

Referenced by addSpinLoopDetection().

7.666 pin/spin_loop_detection.h File Reference

```
#include "spin_loop_detector.h"
#include "inst_mode.h"
#include "pin.H"
```

Classes

- struct **std::hash**< **REG** >
- struct **SpinLoopDetectionState**

Namespaces

- **std**

Functions

- void **addSpinLoopDetection** (TRACE trace, INS ins, **InstMode::inst_mode_t** inst_mode)

7.666.1 Function Documentation

7.666.1.1 addSpinLoopDetection()

```
void addSpinLoopDetection (
             TRACE trace,
             INS ins,
              InstMode::inst_mode_t inst_mode )
```

Definition at line 59 of file spin_loop_detection.cc.

References INSTR_IF_DETAILED, INSTRUMENT_IF_PREDICATED, INSTRUMENT_PREDICATED, INSTRUM↩ ENT_THEN_PREDICATED, spinloopHandleBranch(), spinloopHandleRegWriteAfter(), spinloopHandleRegWrite↩ Before(), spinloopHandleWriteAfter(), spinloopHandleWriteBefore(), and spinloopIfInSpin().

Referenced by InstructionModeling::addInstructionModeling().

7.667 pin/toolreg.cc File Reference

```
#include "toolreg.h"
#include "log.h"
```

Functions

- void **initToolregs** (void)

Variables

- REG **g_toolregs** [**TOOLREGS_SIZE**]

7.667.1 Function Documentation

7.667.1.1 initToolregs()

```
void initToolregs (
             void  )
```

Definition at line 6 of file toolreg.cc.

References g_toolregs, LOG_ASSERT_ERROR, and TOOLREGS_SIZE.

Referenced by main().

7.667.2 Variable Documentation

7.667.2.1 g_toolregs

```
REG g_toolregs[ TOOLREGS_SIZE]
```

Definition at line 4 of file toolreg.cc.

Referenced by lite::addMemoryModeling(), initToolregs(), and traceCallback().

7.668 pin/toolreg.h File Reference

```
#include "fixed_types.h"
#include "pin.H"
```

Macros

- #define **TOOLREG_NUM_MEM** 3

Enumerations

- enum **toolreg_t** {
  TOOLREG_TEMP, **TOOLREG_MEM0**, **TOOLREG_MEM1**, **TOOLREG_MEM2**,
  TOOLREG_WRITEADDR, **TOOLREG_EA0**, **TOOLREG_EA1**, **TOOLREG_EA2**,
  TOOLREGS_SIZE }

Functions

- void **initToolregs** (void)

Variables

- REG **g_toolregs** [**TOOLREGS_SIZE**]

7.668.1 Macro Definition Documentation

7.668.1.1 TOOLREG_NUM_MEM

```
#define TOOLREG_NUM_MEM 3
```

Definition at line 11 of file toolreg.h.

7.668.2 Enumeration Type Documentation

7.668.2.1 toolreg_t

```
enum  toolreg_t
```

Enumerator

TOOLREG_TEMP	
TOOLREG_MEM0	
TOOLREG_MEM1	
TOOLREG_MEM2	
TOOLREG_WRITEADDR	
TOOLREG_EA0	
TOOLREG_EA1	
TOOLREG_EA2	
TOOLREGS_SIZE	

Definition at line 13 of file toolreg.h.

7.668.3 Function Documentation

7.668.3.1 initToolregs()

```
void initToolregs (
             void  )
```

Definition at line 6 of file toolreg.cc.

References g_toolregs, LOG_ASSERT_ERROR, and TOOLREGS_SIZE.

Referenced by main().

7.668.4 Variable Documentation

7.668.4.1 g_toolregs

```
REG g_toolregs[ TOOLREGS_SIZE]
```

Definition at line 4 of file toolreg.cc.

Referenced by lite::addMemoryModeling(), initToolregs(), and traceCallback().

7.669 pin/trace_rtn.cc File Reference

```
#include "trace_rtn.h"
#include "simulator.h"
#include "local_storage.h"
#include "routine_tracer.h"
```

Functions

- void **routineEnter** (THREADID threadIndex, **IntPtr** eip, **IntPtr** esp)
- void **routineExit** (THREADID threadIndex, **IntPtr** eip, **IntPtr** esp)
- void **routineAssert** (THREADID threadIndex, **IntPtr** eip, **IntPtr** esp)
- void **routineCallSite** (THREADID threadIndex, **IntPtr** eip)
- void **announceRoutine** (INS ins)
- void **announceRoutine** (RTN rtn)
- void **announceInvalidRoutine** ()
- void **addRtnTracer** (RTN rtn)
- void **addRtnTracer** (TRACE trace)

7.669.1 Function Documentation

7.669.1.1 addRtnTracer() [1/2]

```
void addRtnTracer (
             RTN rtn )
```

Definition at line 55 of file trace_rtn.cc.

References announceRoutine(), routineEnter(), and routineExit().

Referenced by lite::routineCallback(), and traceCallback().

7.669.1.2 addRtnTracer() [2/2]

```
void addRtnTracer (
             TRACE trace )
```

Definition at line 70 of file trace_rtn.cc.

References announceInvalidRoutine(), announceRoutine(), routineAssert(), and routineCallSite().

7.669.1.3 announceInvalidRoutine()

```
void announceInvalidRoutine ( )
```

Definition at line 50 of file trace_rtn.cc.

Referenced by addRtnTracer().

7.669.1.4 announceRoutine() [1/2]

```
void announceRoutine (
             INS ins )
```

Definition at line 27 of file trace_rtn.cc.

Referenced by addRtnTracer(), and announceRoutine().

7.669.1.5 announceRoutine() [2/2]

```
void announceRoutine (
             RTN rtn )
```

Definition at line 45 of file trace_rtn.cc.

References announceRoutine().

7.669.1.6 routineAssert()

```
void routineAssert (
             THREADID threadIndex,
              IntPtr eip,
              IntPtr esp )
```

Definition at line 17 of file trace_rtn.cc.

References localStore.

Referenced by addRtnTracer().

7.669.1.7 routineCallSite()

```
void routineCallSite (
             THREADID threadIndex,
              IntPtr eip )
```

Definition at line 22 of file trace_rtn.cc.

References localStore.

Referenced by addRtnTracer().

7.669.1.8 routineEnter()

```
void routineEnter (
             THREADID threadIndex,
              IntPtr eip,
              IntPtr esp )
```

Definition at line 6 of file trace_rtn.cc.

References localStore.

Referenced by addRtnTracer().

7.669.1.9 routineExit()

```
void routineExit (
             THREADID threadIndex,
              IntPtr eip,
              IntPtr esp )
```

Definition at line 12 of file trace_rtn.cc.

References localStore.

Referenced by addRtnTracer().

7.670 pin/trace_rtn.h File Reference

```
#include "fixed_types.h"
#include "pin.H"
```

Functions

- void **addRtnTracer** (RTN rtn)
- void **addRtnTracer** (TRACE trace)

7.670.1 Function Documentation

7.670.1.1 addRtnTracer() [1/2]

```
void addRtnTracer (
            RTN rtn )
```

Definition at line 55 of file trace_rtn.cc.

References announceRoutine(), routineEnter(), and routineExit().

Referenced by lite::routineCallback(), and traceCallback().

7.670.1.2 addRtnTracer() [2/2]

```
void addRtnTracer (
            TRACE trace )
```

Definition at line 70 of file trace_rtn.cc.

References announceInvalidRoutine(), announceRoutine(), routineAssert(), and routineCallSite().

Index