gssync

Generated by Doxygen 1.8.13

Contents

Chapter 1

Main Page

The GreenSocs Synchronization library provides a number of different policies for synchronizing between an external simulator (typically QEMU) and SystemC.

These are based on a proposed standard means to handle the SystemC simulator. This library provides a backwards compatibility layer, but the patched version of SystemC will perform better.

The libgssync library depends on the libgsutils and base-components libraries.

In addition the library contains utilities such as an thread safe event (async_event) and a real time speed limited for SystemC.

This patch adds four new basic functions to SystemC:

```
void sc_suspend_all(sc_simcontext* csc= sc_get_curr_simcontext())
void sc_unsuspend_all(sc_simcontext* csc= sc_get_curr_simcontext())
void sc_unsuspendable()
void sc_suspendable()
```

suspend_all/unsuspend_all : This pair of functions requests the kernel to ‘atomically suspend’ all processes (using the same semantics as the thread suspend() call). This is atomic in that the kernel will only suspend all the processes together, such that they can be suspended and unsuspended without any side effects. Calling suspend_all(), and subsiquently calling unsuspend_all() will have no effect on the suspended status of an individual process. A process may call suspend_all() followed by unsuspend_all, the calls should be ‘paired’, (multiple calls to either suspend_all() or unsuspend_all() will be ignored). Outside of the context of a process, it is the programmers responsibility to ensure that the calls are paired. As a consequence, multiple calls to suspend_all() may be made (within separate process, or from within sc_main). So long as there have been more calls to suspend_all() than to unsuspend_all(), the kernel will suspend all processes.

[note, this patch set does not add convenience functions, including those to find out if suspension has happened, these are expected to be layered ontop]

unsusbendable()/suspendable(): This pair of functions provides an ‘opt-out’ for specific process to the suspend↩ _all(). The consequence is that if there is a process that has opted out, the kernel will not be able to suspend_all (as it would no longer be atomic). These functions can only be called from within a process. A process should only call suspendable/unsuspendable in pairs (multiple calls to either will be ignored). *Note that the default is that a process is marked as suspendable.*

Use cases: *1 : Save and Restore* For Save and Restore, the expectation is that when a save is requested, ‘suspend_all’ will be called. If there are models that are in an unsuspendable state, the entire simulation will be allowed to continue until such a time that there are no unsuspendable processes.

2 : External sync When an external model injects events into a SystemC model (for instance, using an ‘async_↩ request_update()’), time can drift between the two simulators. In order to maintain time, SystemC can be prevented from advancing by calling suspend_all(). If there are process in an unsuspendable state (for instance, processing on behalf of the external model), then the simulation will be allowed to continue. NOTE, an event injected into the kernel by an async_request_update will cause the kernel to execute the associated update() function (leaving the suspended state). The update function should arrange to mark any processes that it requires as unsuspendable before the end of the current delta cycle, to ensure that they are scheduled.

It is possible to test the correct functioning of the different components with the tests that are proposed.

Once you have compiled your library, you will have a integration_tests folder in your construction directory.

In this test folder you will find several executable which each correspond to a test. You can run the executable with :

```
./build/tests/integration_tests/<name_of_component>_test
```

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

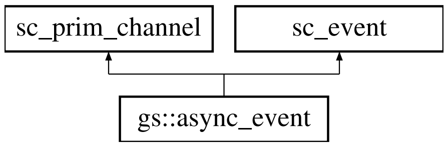
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Class Documentation

4.1 gs::async_event Class Reference

Inheritance diagram for gs::async_event:



Public Member Functions

- **async_event** (bool start_attached=true)
- void **async_notify** ()
- void **notify** (sc_core::sc_time delay=sc_core::sc_time(sc_core::SC_ZERO_TIME))
- void **async_attach_suspending** ()
- void **async_detach_suspending** ()
- void **enable_attach_suspending** (bool e)

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

- /home/thomas/Documents/GreenSocs/build-lib/libgssync/include/greensocs/libgssync/async_event.h

4.2 gs::RunOnSysC::AsyncJob Class Reference

Public Types

- using **Ptr** = std::shared_ptr< AsyncJob >

Public Member Functions

- **AsyncJob** (std::function< void()> &&job)
- **AsyncJob** (std::function< void()> &job)
- **AsyncJob** (const AsyncJob &)=delete
- void **operator()** ()
- void cancel ()

  Cancel a job.
- void **wait** ()
- bool **is_cancelled** () const

4.2.1 Member Function Documentation

4.2.1.1 cancel()

```
void gs::RunOnSysC::AsyncJob::cancel ( )  [inline]
```

Cancel a job.

Cancel a job by setting m_cancelled to true and by resetting the task. Any waiter will then be unblocked immediately.

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

- /home/thomas/Documents/GreenSocs/build-lib/libgssync/include/greensocs/libgssync/runonsysc.h

4.3 gs::global_pause Class Reference

Public Member Functions

- **global_pause** (const global_pause &)=delete
- void **suspendable** ()
- void **unsuspendable** ()
- void **unsuspend_all** ()
- void **suspend_all** ()
- bool **attach_suspending** (sc_core::sc_prim_channel ∗p)
- bool **detach_suspending** (sc_core::sc_prim_channel ∗p)
- void **async_wakeup** ()

Static Public Member Functions

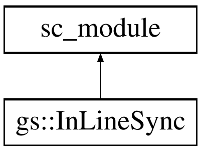
- static global_pause & **get** ()

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

- /home/thomas/Documents/GreenSocs/build-lib/libgssync/include/greensocs/libgssync/pre_suspending_sc↩
  _support.h
- /home/thomas/Documents/GreenSocs/build-lib/libgssync/src/pre_suspending_sc_support.cc

4.4 gs::InLineSync Class Reference

Inheritance diagram for gs::InLineSync:



Public Member Functions

- **SC_HAS_PROCESS** (InLineSync)
- **InLineSync** (const sc_core::sc_module_name &name)
- void **b_transport** (tlm::tlm_generic_payload &trans, sc_core::sc_time &delay)
- void **get_direct_mem_ptr** (tlm::tlm_generic_payload &trans, tlm::tlm_dmi &dmi_data)
- void **transport_dgb** (tlm::tlm_generic_payload &trans)
- void **invalidate_direct_mem_ptr** (sc_dt::uint64 start_range, sc_dt::uint64 end_range)

Public Attributes

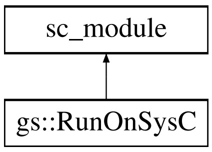
- tlm_utils::simple_target_socket< InLineSync > **target_socket**
- tlm_utils::simple_initiator_socket< InLineSync > **initiator_socket**

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

- /home/thomas/Documents/GreenSocs/build-lib/libgssync/include/greensocs/libgssync/inlinesync.h

4.5 gs::RunOnSysC Class Reference

Inheritance diagram for gs::RunOnSysC:



Classes

- class AsyncJob

Public Member Functions

- **RunOnSysC** (const sc_core::sc_module_name &n=sc_core::sc_module_name("run-on-sysc"))
- void cancel_pendings ()

  Cancel all pending jobs.
- void cancel_all ()

  Cancel all pending and running jobs.
- void **end_of_simulation** ()
- void **fork_on_systemc** (std::function< void()> job_entry)
- bool run_on_sysc (std::function< void()> job_entry, bool wait=true)

  Run a job on the SystemC kernel thread.

Protected Member Functions

- void **jobs_handler** ()
- void **cancel_pendings_locked** ()

Protected Attributes

- std::thread::id **m_thread_id**
- std::queue< AsyncJob::Ptr > **m_async_jobs**
- AsyncJob::Ptr **m_running_job**
- std::mutex **m_async_jobs_mutex**
- async_event **m_jobs_handler_event**

4.5.1 Member Function Documentation

4.5.1.1 cancel_all()

```
void gs::RunOnSysC::cancel_all ( )  [inline]
```

Cancel all pending and running jobs.

Cancel all the pending jobs and the currently running job. The callers will be unblocked if they are waiting for the job. Note that if the currently running job is resumed, the behaviour is undefined. This method is meant to be called after simulation has ended.

4.5.1.2 cancel_pendings()

```
void gs::RunOnSysC::cancel_pendings ( )  [inline]
```

Cancel all pending jobs.

Cancel all the pending jobs. The callers will be unblocked if they are waiting for the job.

4.5.1.3 run_on_sysc()

```
bool gs::RunOnSysC::run_on_sysc (
            std::function< void()> job_entry,
            bool wait = true )  [inline]
```

Run a job on the SystemC kernel thread.

Parameters

in	*job_entry*	The job to run
in	*wait*	If true, wait for job completion

Returns

true if the job has been succesfully executed or if `wait` was false, false if it has been cancelled (see `Run↩ OnSysC::cancel_all`).

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

- /home/thomas/Documents/GreenSocs/build-lib/libgssync/include/greensocs/libgssync/runonsysc.h

4.6 gs::semaphore Class Reference

Public Member Functions

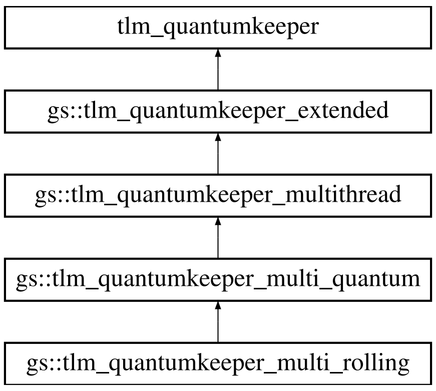
- void **notify** ()
- void **wait** ()
- bool **try_wait** ()

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

- /home/thomas/Documents/GreenSocs/build-lib/libgssync/include/greensocs/libgssync/semaphore.h

4.7 gs::tlm_quantumkeeper_extended Class Reference

Inheritance diagram for gs::tlm_quantumkeeper_extended:


tlm_quantumkeeper
gs::tlm_quantumkeeper_extended
gs::tlm_quantumkeeper_multithread
gs::tlm_quantumkeeper_multi_quantum
gs::tlm_quantumkeeper_multi_rolling

Public Member Functions

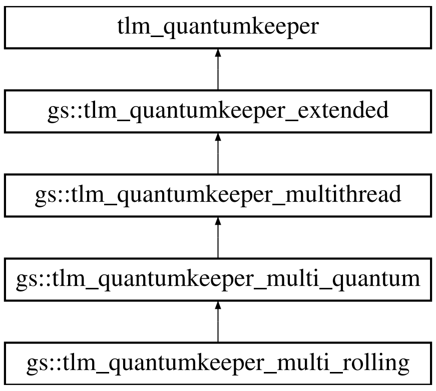
- virtual sc_core::sc_time **time_to_sync** ()
- virtual void **stop** ()
- virtual void **start** (std::function< void()> job=nullptr)
- virtual SyncPolicy::Type **get_thread_type** () const
- virtual bool **need_sync** ()
- virtual bool **need_sync** () const override
- virtual void **run_on_systemc** (std::function< void()> job)

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

- /home/thomas/Documents/GreenSocs/build-lib/libgssync/include/greensocs/libgssync/qk_extendedif.h

4.8 gs::tlm_quantumkeeper_multi_quantum Class Reference

Inheritance diagram for gs::tlm_quantumkeeper_multi_quantum:


tlm_quantumkeeper
gs::tlm_quantumkeeper_extended
gs::tlm_quantumkeeper_multithread
gs::tlm_quantumkeeper_multi_quantum
gs::tlm_quantumkeeper_multi_rolling

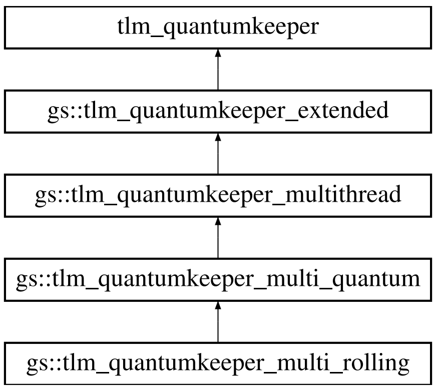
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libgssync/include/greensocs/libgssync/qkmulti-quantum.h

4.9 gs::tlm_quantumkeeper_multi_rolling Class Reference

Inheritance diagram for gs::tlm_quantumkeeper_multi_rolling:


tlm_quantumkeeper
gs::tlm_quantumkeeper_extended
gs::tlm_quantumkeeper_multithread
gs::tlm_quantumkeeper_multi_quantum
gs::tlm_quantumkeeper_multi_rolling

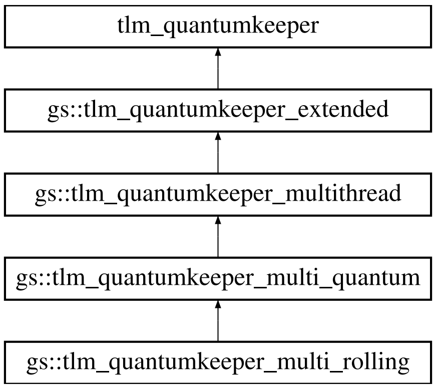
Additional Inherited Members

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

- /home/thomas/Documents/GreenSocs/build-lib/libgssync/include/greensocs/libgssync/qkmulti-rolling.h

4.10 gs::tlm_quantumkeeper_multithread Class Reference

Inheritance diagram for gs::tlm_quantumkeeper_multithread:


tlm_quantumkeeper
gs::tlm_quantumkeeper_extended
gs::tlm_quantumkeeper_multithread
gs::tlm_quantumkeeper_multi_quantum
gs::tlm_quantumkeeper_multi_rolling

Public Member Functions

- virtual SyncPolicy::Type **get_thread_type** () const override
- virtual void **start** (std::function< void()> job=nullptr) override
- virtual void **stop** () override
- virtual sc_core::sc_time **time_to_sync** () override
- void **inc** (const sc_core::sc_time &t) override
- void **set** (const sc_core::sc_time &t) override
- virtual void **sync** () override
- void **reset** () override
- sc_core::sc_time **get_current_time** () const override
- sc_core::sc_time **get_local_time** () const override
- virtual bool **need_sync** () override

Protected Types

- enum **jobstates** { **NONE**, **RUNNING**, **STOPPED** }

Protected Member Functions

- virtual bool **is_sysc_thread** () const

Protected Attributes

- enum gs::tlm_quantumkeeper_multithread::jobstates **status**
- async_event **m_tick**

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

- /home/thomas/Documents/GreenSocs/build-lib/libgssync/include/greensocs/libgssync/qkmultithread.h
- /home/thomas/Documents/GreenSocs/build-lib/libgssync/src/qkmultithread.cc

Index