ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/306355995

# Coupling gem5 with SystemC TLM 2.0 Virtual Platforms

**Presentation** · June 2015

DOI: 10.13140/RG.2.2.31607.21925

CITATIONS
4

READS
1,106

**2 authors:**

Matthias Jung
Fraunhofer Institute for Experimental Software Engineering IESE
**66** PUBLICATIONS **559** CITATIONS

Norbert Wehn
Technische Universität Kaiserslautern
**440** PUBLICATIONS **4,826** CITATIONS

SEE PROFILE

SEE PROFILE

**Some of the authors of this publication are also working on these related projects:**

Project Research Assitant - Microelectronics Research Group TU Kaiserslautern View project

Project TEMPO - Technology and hardware for neuromorphic computing View project

All content following this page was uploaded by Matthias Jung on 21 August 2016.

The user has requested enhancement of the downloaded file.

MICROELECTRONIC
SYSTEMS DESIGN
RESEARCH GROUP

# Coulpling gem5 with
# IEEE1666 SystemC TLM2.0

Dipl.-Ing. Matthias Jung

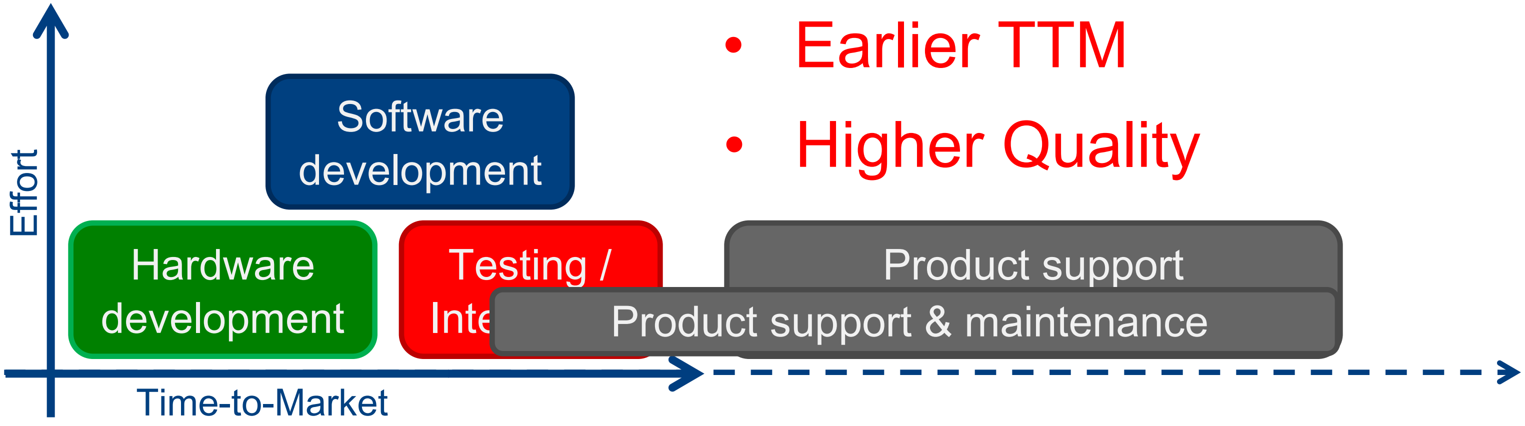
Microelectronic Systems Design Research Group

Sonntag, 21. August 2016

Technische Universität Kaiserslautern

# Virtual Prototypes In Industry

- **High-speed** functional **software models** of physical hardware
- Visibility and controllability over the entire system
- Powerful debugging and analysis tools
- Reuse of components for future projects
- Fast **Design space exploration** (for HW engineers)
- Easy to exchange, worldwide
- **Concurrent HW and SW development:**



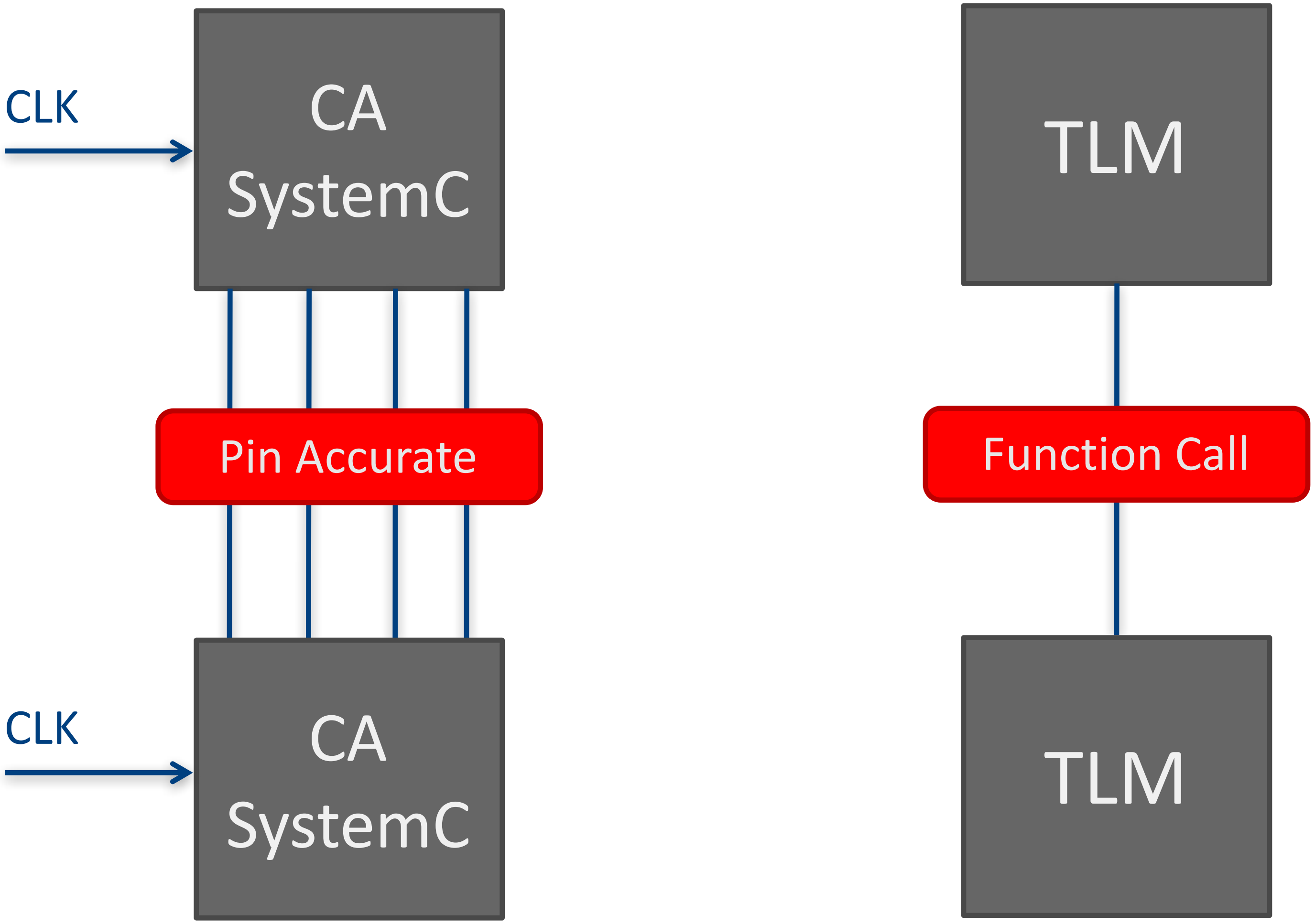
- Earlier TTM
- Higher Quality

# SystemC IEEE 1666

- Modeling language for HW and SW components
- Extends C++ to an event-driven simulation kernel
- Different levels of accuracy
- IEEE Standard, Maintained by Accellera
- 10-100x Faster than CA VHDL/Verilog Simulation

→**However, standard CA SystemC is not fast enough to boot, an operating system.**

# Transaction Level Modeling

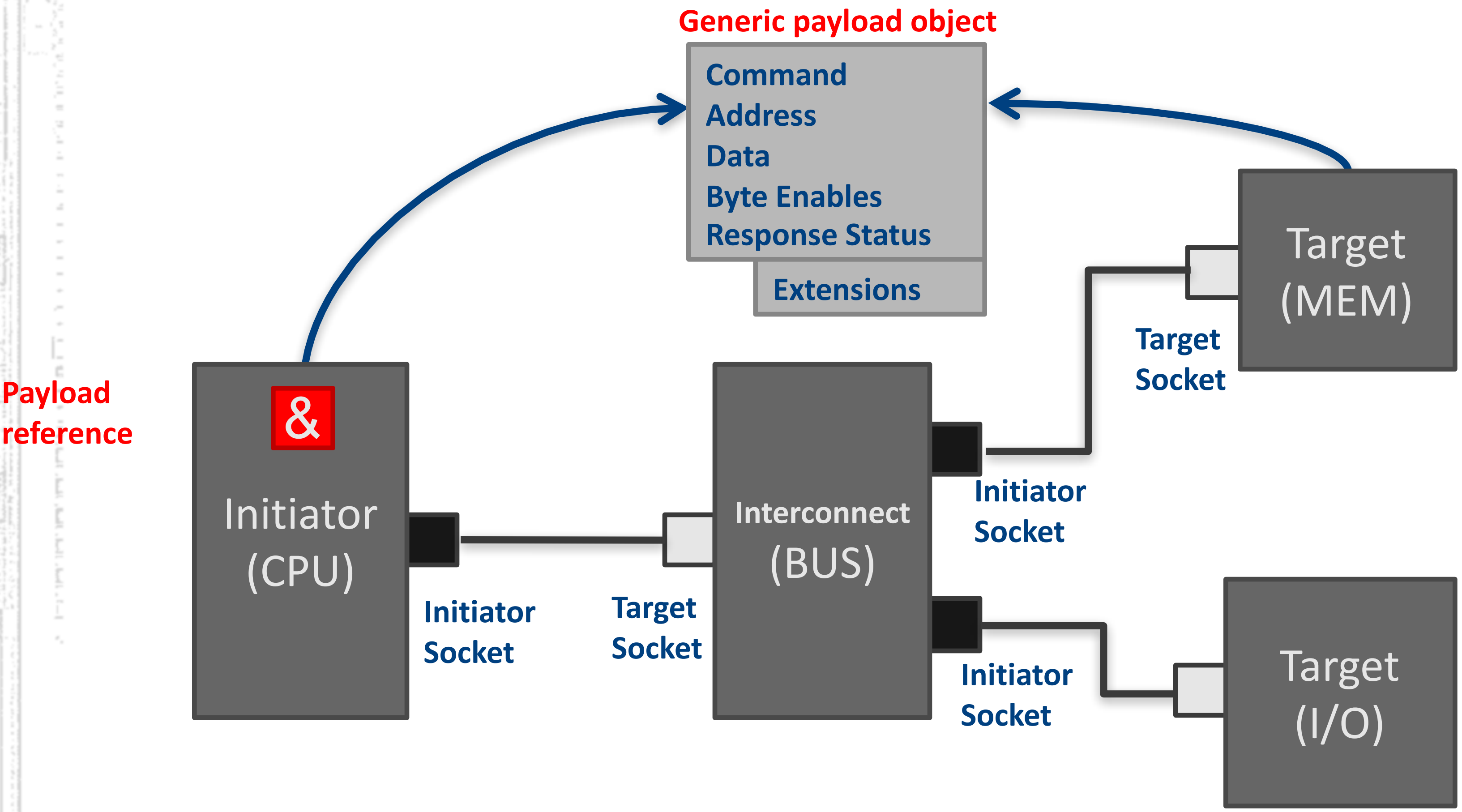


Simulate every event!!

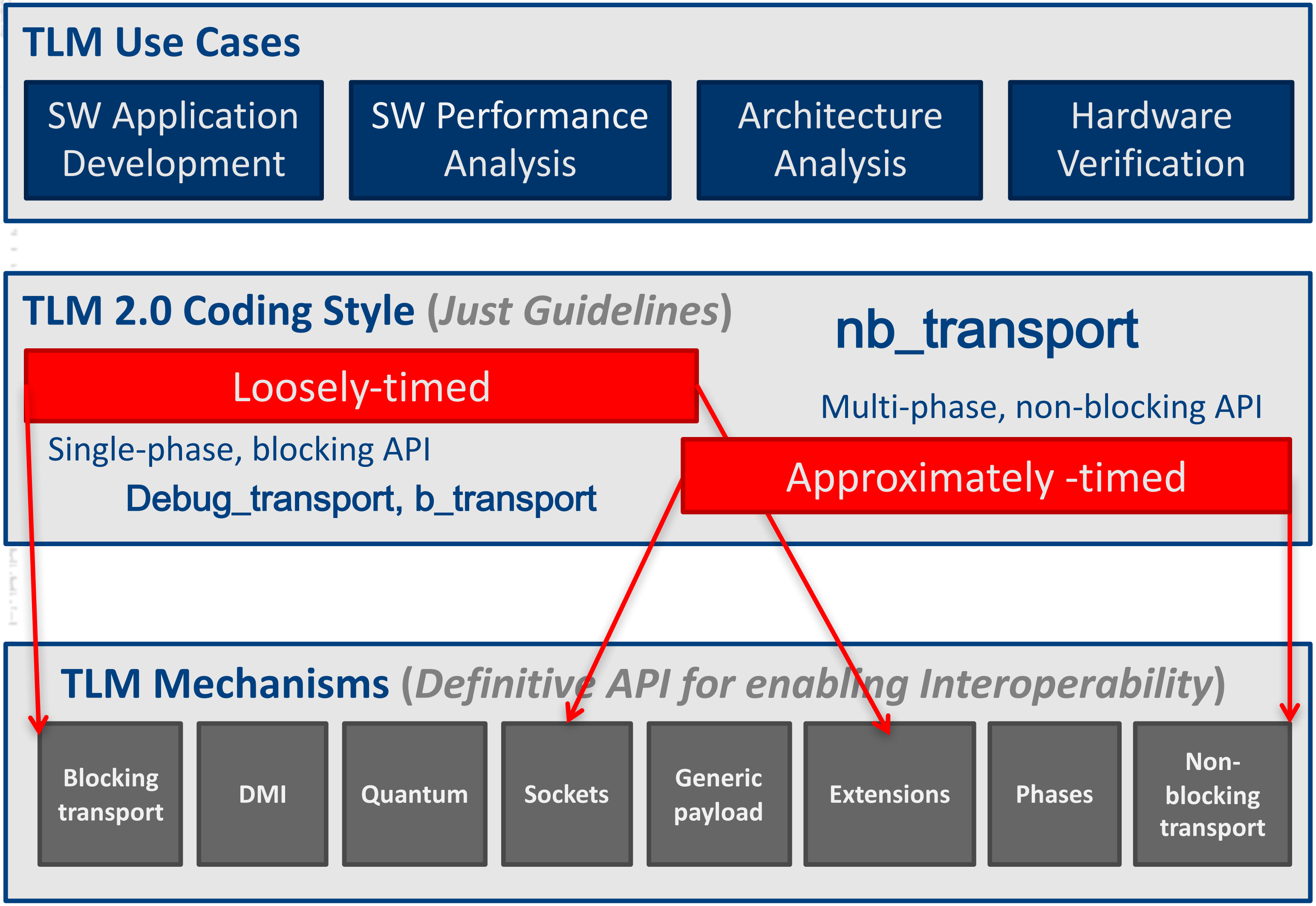
100-10,000 X faster simulation!

*Source: Doulos Ldt. www.doulos.com*

# Generic Payload



# TLM Coding Styles and Mechanisms



**TLM Use Cases**

- SW Application Development
- SW Performance Analysis
- Architecture Analysis
- Hardware Verification

**TLM 2.0 Coding Style** (*Just Guidelines*)

Loosely-timed

Single-phase, blocking API

**Debug_transport, b_transport**

**nb_transport**

Multi-phase, non-blocking API

Approximately -timed

**TLM Mechanisms** (*Definitive API for enabling Interoperability*)

- Blocking transport
- DMI
- Quantum
- Sockets
- Generic payload
- Extensions
- Phases
- Non-blocking transport

*Source: Doulos Ldt. www.doulos.com*

# Tool Vendors for TLM2.0 VP

**TLM is widely used in Industry:**

**The market of virtual platform tools:**

- Synopsys - Platform Architect
- Cadence - Virtual System Platform
- Mentor Graphics - Vista Virtual prototyping
- Imperas – OpenVP

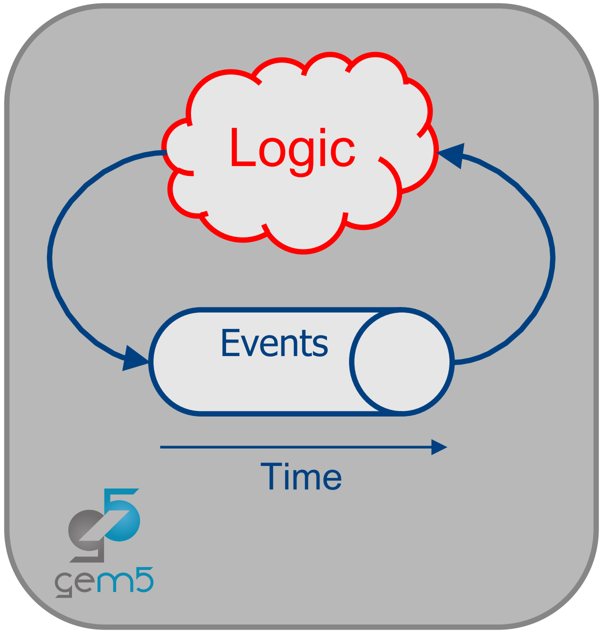
**Virtual Platform Core Models:**

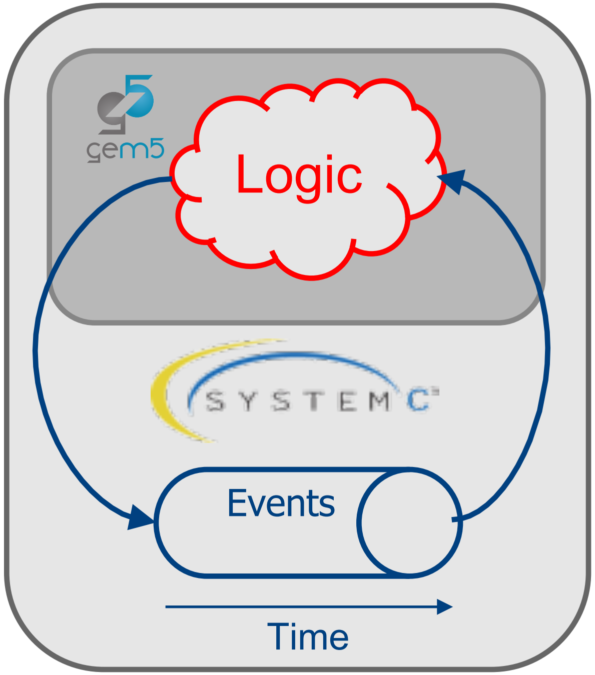
- ARM (Fastmodels):
  - only LT models based on JIT, non-free
- Carbon Design Systems:
  - Cycle Accurate (CA) Models in TLM Wrapper, non-free
- Imperas / OVP:
  - only LT, Free

→ An accurate, free available and changeable core model is needed

# Couple gem5 with SystemC

- Since June 2014 gem5 supports a SystemC coupling. However, no TLM
- Gem5 is build as a C++ library
- It is hooked onto SystemC's event loop.
- The hosting is achieved by replacing 'simulate' with a SystemC object, which implements an event loop using the SystemC scheduler mechanisms.
- Currently only one gem5 instance can be in a SystemC simulation.





# Transaction Models in gem5

## Timing

- The most detailed access: queuing delay + resource contention
- Similar to the TLM **nb_transport** interface.

## Atomic

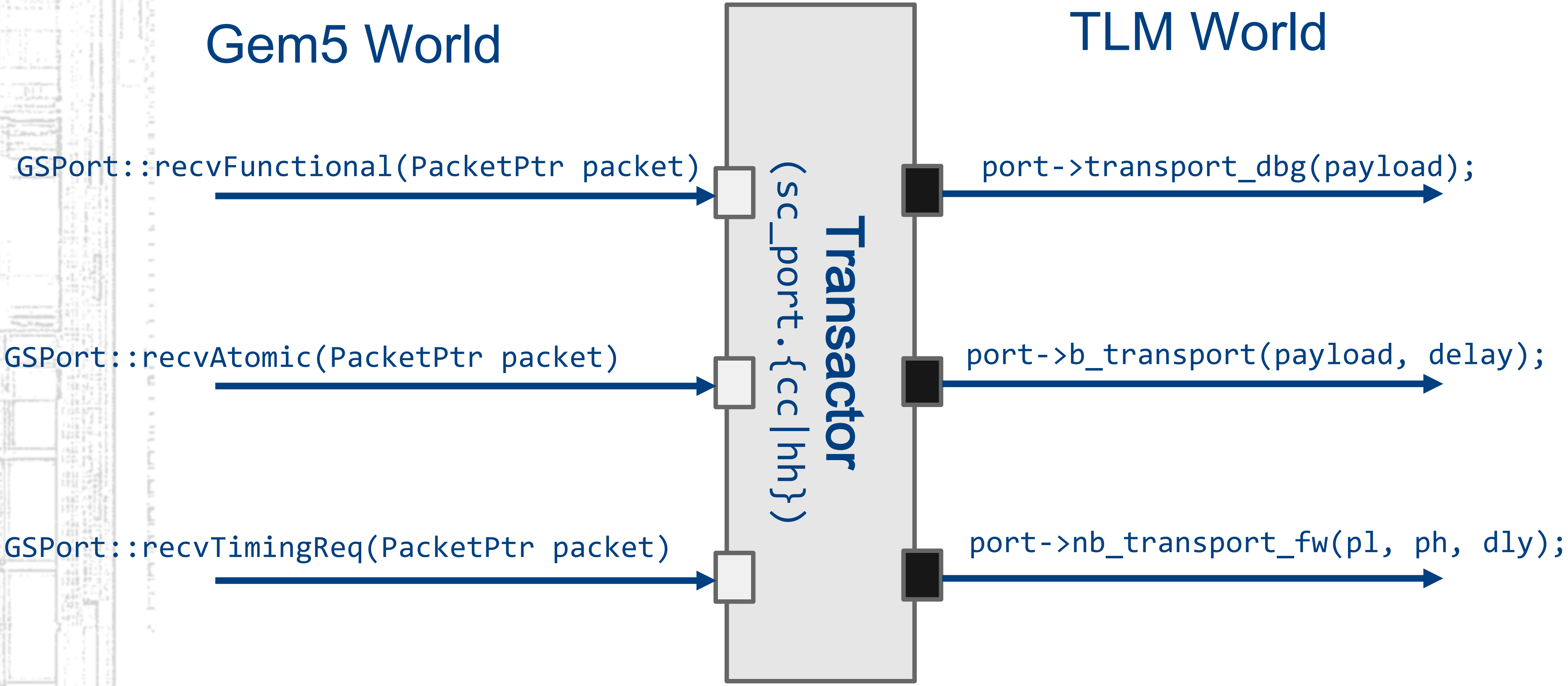
- Accesses are a faster than detailed access
- Used for **fast forwarding** and **warming up caches**
- Similar to the TLM **b_transport** interface
- Not good for performance simulation

## Functional

- Similar to **transport_debug**

e.g. loading binaries, avoiding deadlocks in multi-level cache coherent networks

# Coupling gem5 with SystemC-TLM

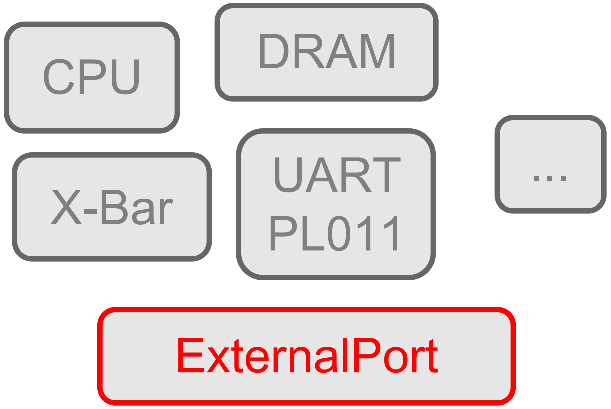


Gem5 World
TLM World
GSPort::recvFunctional(PacketPtr packet)
GSPort::recvAtomic(PacketPtr packet)
GSPort::recvTimingReq(PacketPtr packet)
Transactor
(sc_port.{cc|hh})
port->transport_dbg(payload);
port->b_transport(payload, delay);
port->nb_transport_fw(pl, ph, dly);

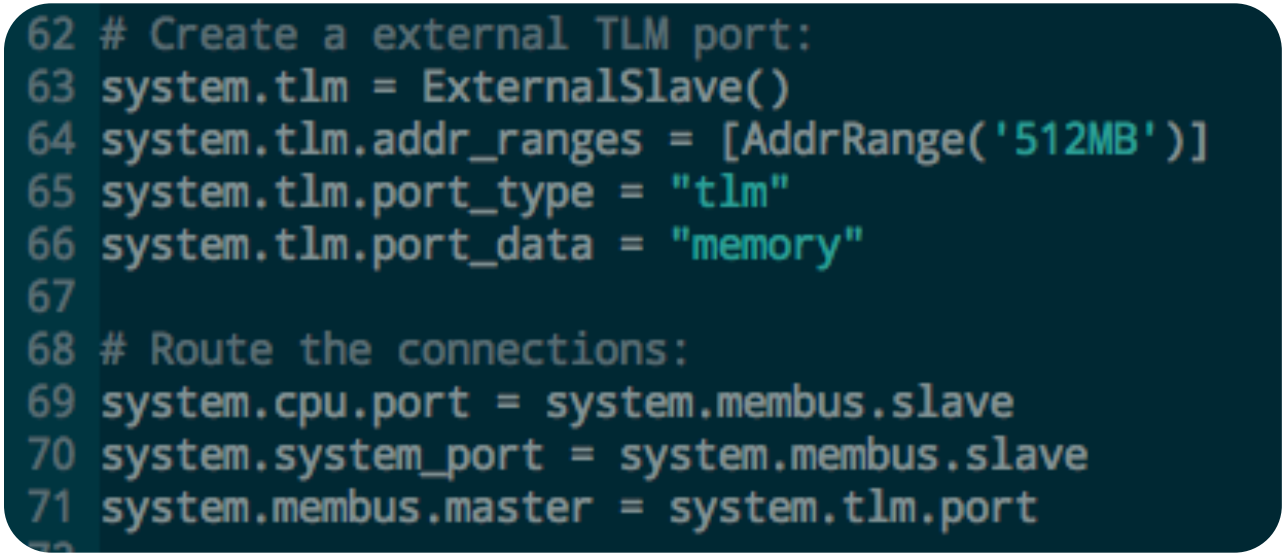
**Transactor:**

- Gem5-Packet to TLM-Payload converter
- Uses gem5 event queue as TLM-Payload-Event queue
- Implements fully TLM2.0 compliant functionality

# Setting up an external Port in gem5



Python Config:



```python
# Create a external TLM port:
system.tlm = ExternalSlave()
system.tlm.addr_ranges = [AddrRange('512MB')]
system.tlm.port_type = "tlm"
system.tlm.port_data = "memory"

# Route the connections:
system.cpu.port = system.membus.slave
system.system_port = system.membus.slave
system.membus.master = system.tlm.port
```

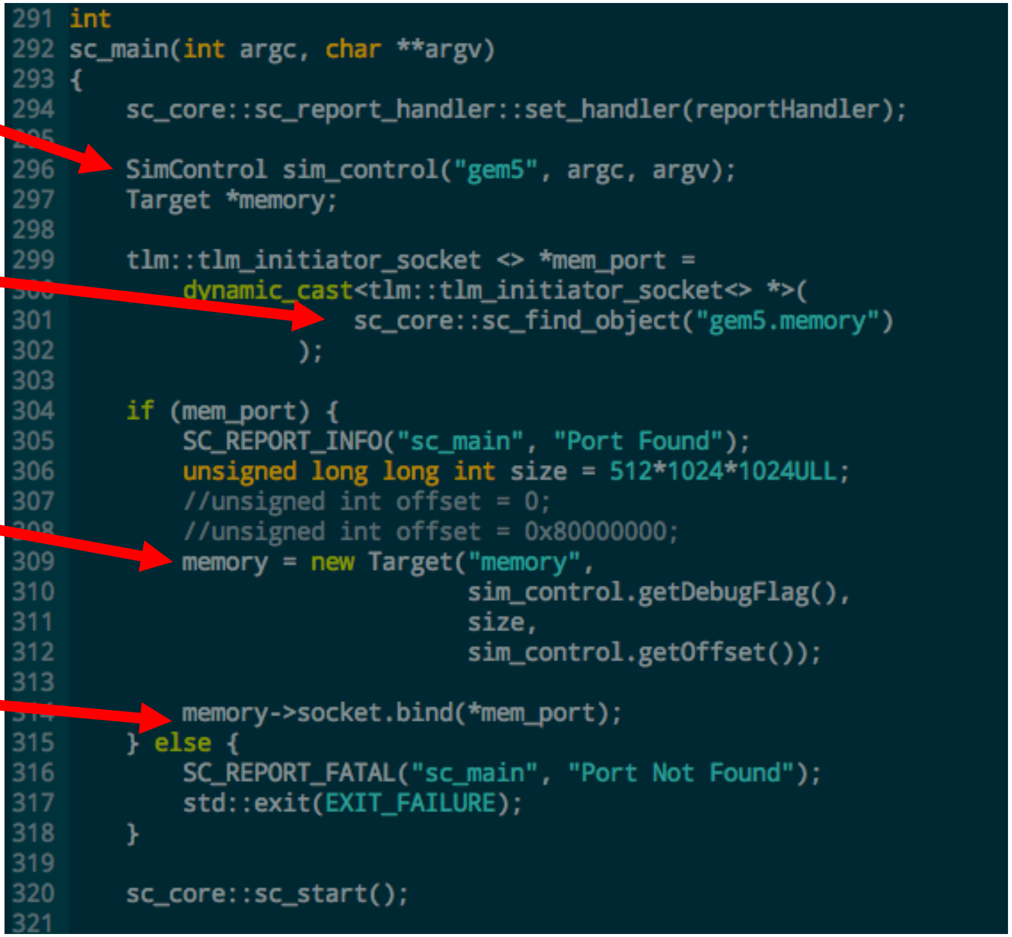
# Picking Ext. Port up in SystemC World

Instantiate gem5 as sc_module

Find external Port with sc_find_object

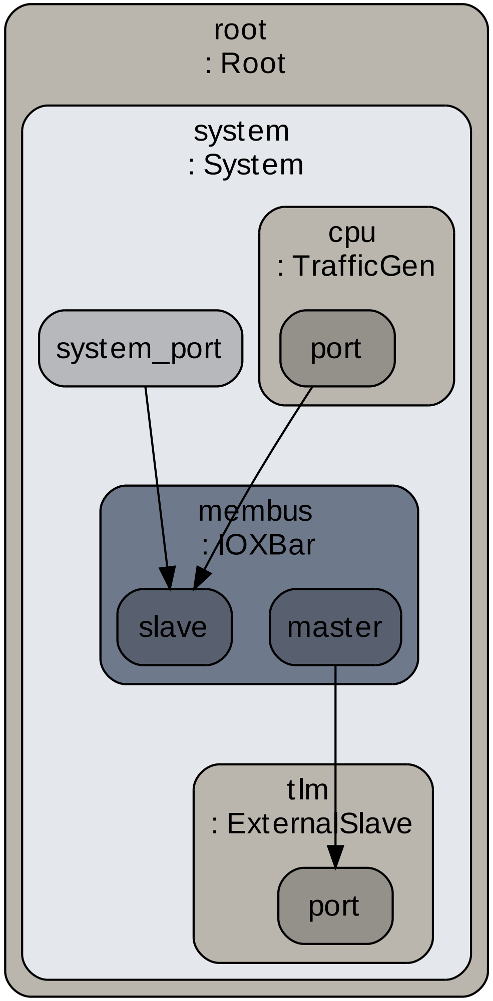
Instantiate a TLM target

Bind the target’s port with the external port of gem5

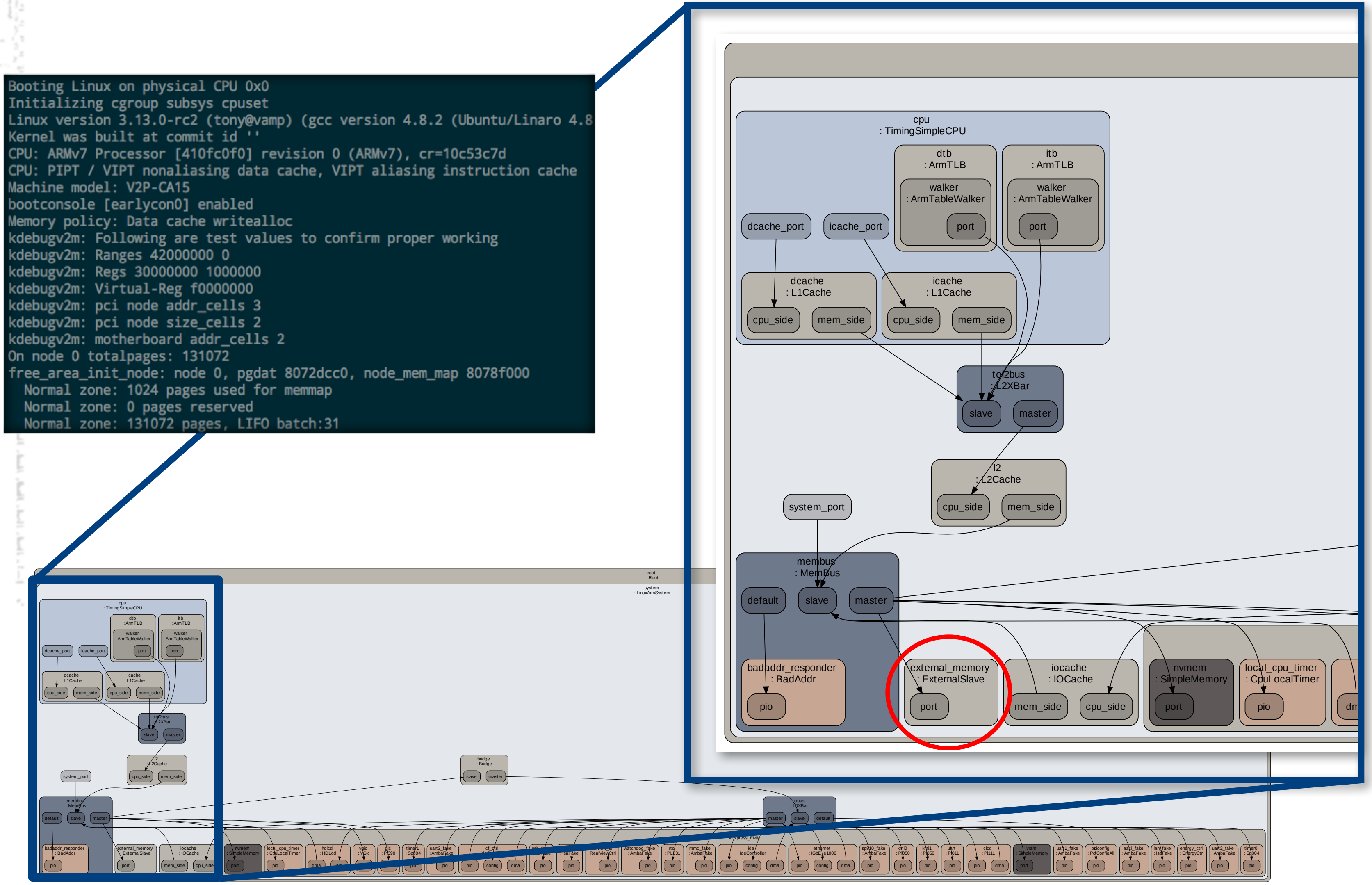


```
int
sc_main(int argc, char **argv)
{
    sc_core::sc_report_handler::set_handler(reportHandler);

    SimControl sim_control("gem5", argc, argv);
    Target *memory;

    tlm::tlm_initiator_socket <> *mem_port =
        dynamic_cast<tlm::tlm_initiator_socket<> *>(
                  sc_core::sc_find_object("gem5.memory")
              );

    if (mem_port) {
        SC_REPORT_INFO("sc_main", "Port Found");
        unsigned long long int size = 512*1024*1024ULL;
        //unsigned int offset = 0;
        //unsigned int offset = 0x80000000;
        memory = new Target("memory",
                            sim_control.getDebugFlag(),
                            size,
                            sim_control.getOffset());

        memory->socket.bind(*mem_port);
    } else {
        SC_REPORT_FATAL("sc_main", "Port Not Found");
        std::exit(EXIT_FAILURE);
    }

    sc_core::sc_start();
```

# Simple Traffic Generator Example



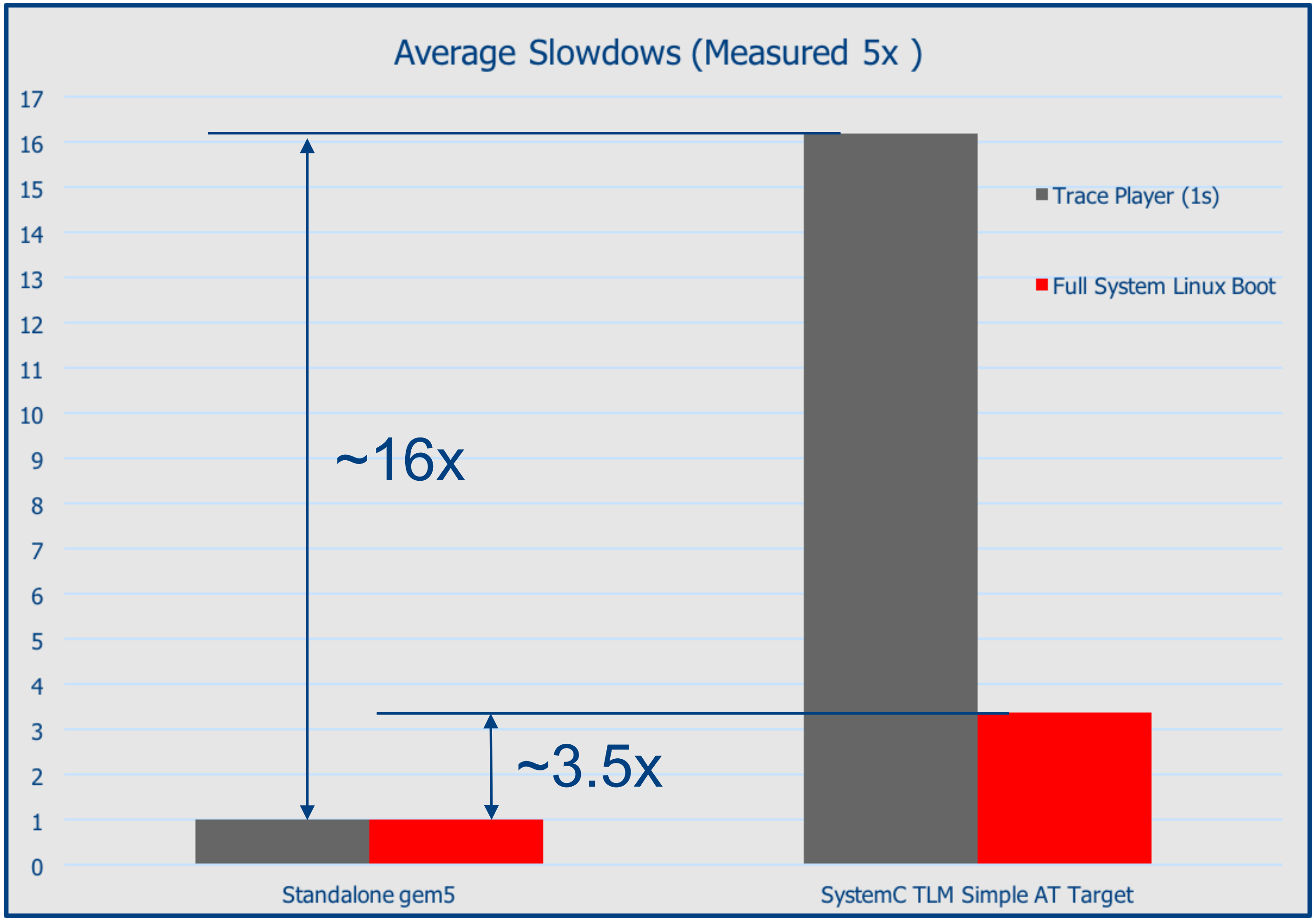
# Full System Linux Boot Example



```
Booting Linux on physical CPU 0x0
Initializing cgroup subsys cpuset
Linux version 3.13.0-rc2 (tony@vamp) (gcc version 4.8.2 (Ubuntu/Linaro 4.8
Kernel was built at commit id ''
CPU: ARMv7 Processor [410fc0f0] revision 0 (ARMv7), cr=10c53c7d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine model: V2P-CA15
bootconsole [earlycon0] enabled
Memory policy: Data cache writealloc
kdebugv2m: Following are test values to confirm proper working
kdebugv2m: Ranges 42000000 0
kdebugv2m: Regs 30000000 1000000
kdebugv2m: Virtual-Reg f0000000
kdebugv2m: pci node addr_cells 3
kdebugv2m: pci node size_cells 2
kdebugv2m: motherboard addr_cells 2
On node 0 totalpages: 131072
free_area_init_node: node 0, pgdat 8072dcc0, node_mem_map 8078f000
  Normal zone: 1024 pages used for memmap
  Normal zone: 0 pages reserved
  Normal zone: 131072 pages, LIFO batch:31
```

MICROELECTRONIC SYSTEMS DESIGN RESEARCH GROUP

# Results Slowdown



MICROELECTRONIC
SYSTEMS DESIGN
RESEARCH GROUP

# Outlook

- Use a TLM memory manager to be fully TLM compliant
- Improve speed (e.g by implementing a TLM memory manager)
- Fix some bugs regarding Atomic Access and Retry
- Connect gem5 to DRAMSys (inhouse DRAM TLM Simulator)
- Use gem5 in Synopsys Platform Architect

Currently on the gem5 review board:

**#2753**

http://reviews.gem5.org/r/2753/

Please test and give feedback

View publication stats