ECE 720 – ESL & Physical Design

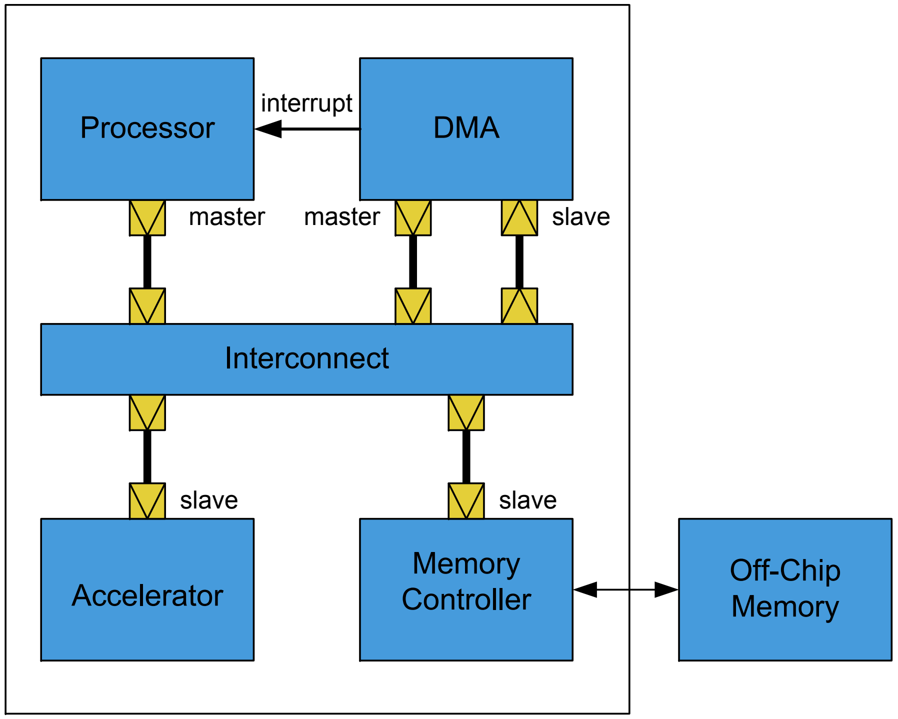
Lecture 10: TLM Processors and Busses

W. Rhett Davis
NC State University

Announcements

- Homework 3 Due Today
- No Lecture on Thursday
  (class will not meet)
- Homework 4 Due in 1 week
- Project 2 Requirements Discussion
  in 1 week

Where we want to go



- We have the basic building blocks of payloads, protocols, & sockets
- How to model this system?
- Today: Buses and Processors

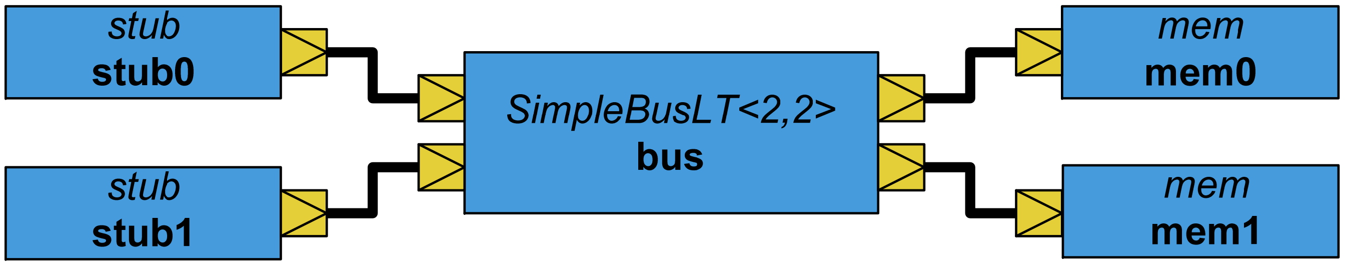
Today's Lecture

→ ● Transaction-Level Modeling of Busses

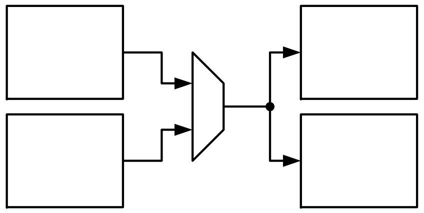
● Transaction-Level Modeling
with Instruction Set Simulation (TLM+ISS)

● Alternative Modeling Approaches

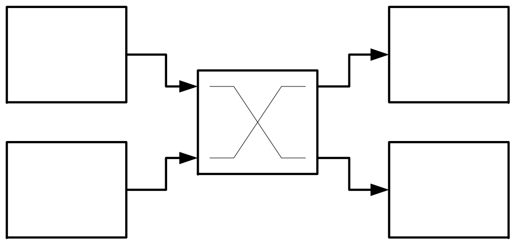
Modeling Bus Behavior



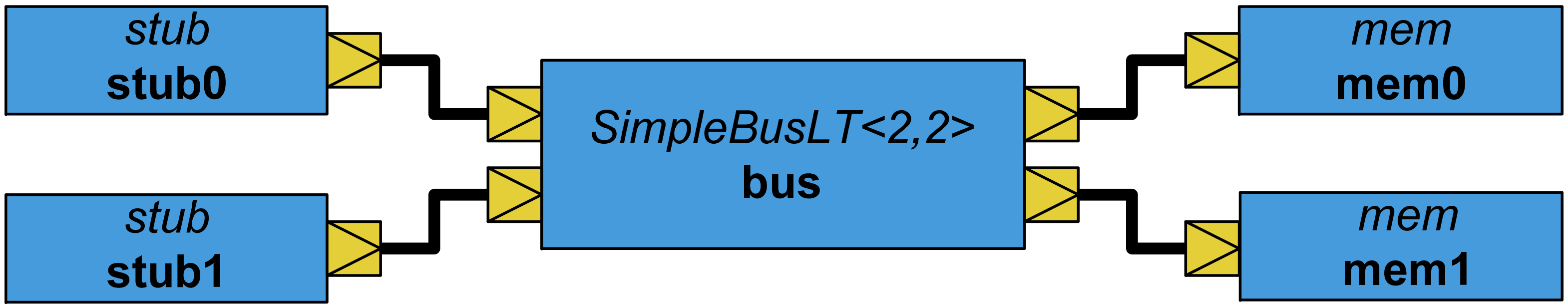
- Does the current Simple LT 2x2 system model
  - a central-MUX-style bus?

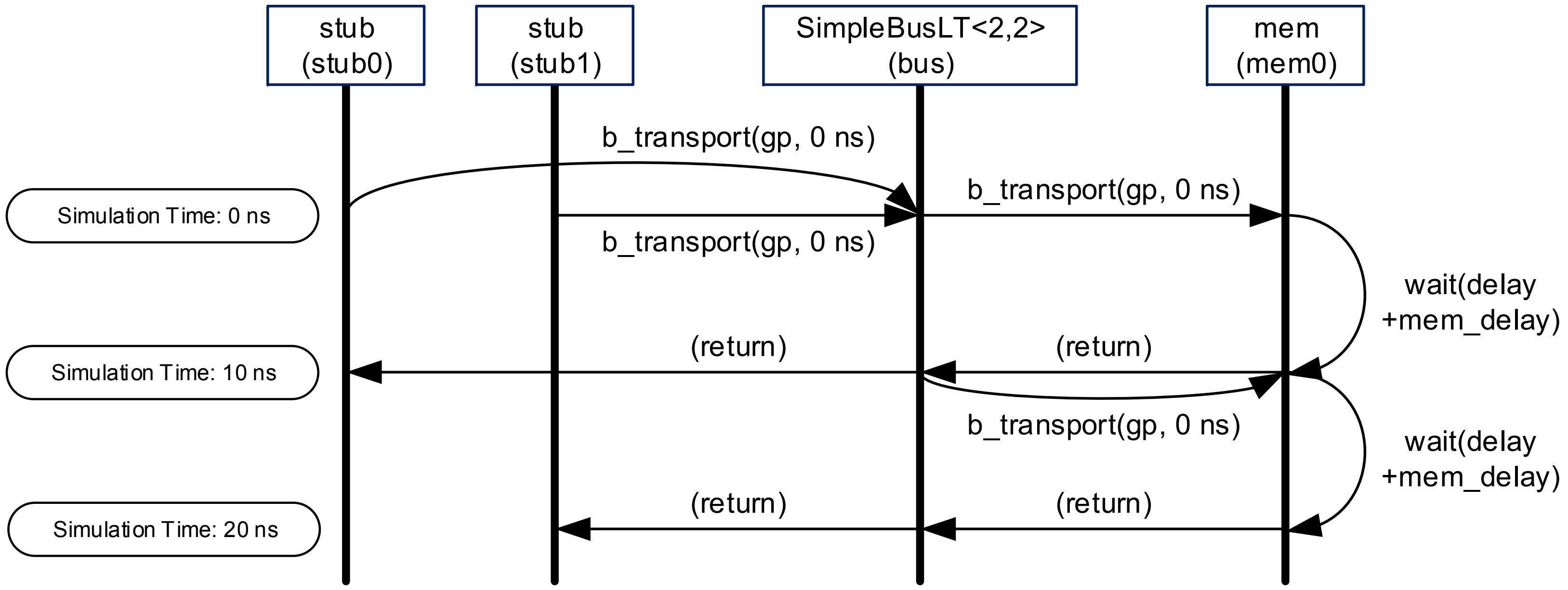


  - a crossbar-style bus?



Sequence Diagram for a MUX-Bus





- How to model this behavior?

Useful State-modeling Constructs

- Events (sc_core::sc_event)
  - Waiting & Triggering –
  - Check State –
  - Update State –
- Payload Event Queues (PEQ) (tlm_utils::peq_with_get)
  - Waiting & Triggering –
  - Check State –
  - Update State –

Today's Lecture

- Transaction-Level Modeling of Busses
- → Transaction-Level Modeling
  with Instruction Set Simulation (TLM+ISS)
- Alternative Modeling Approaches

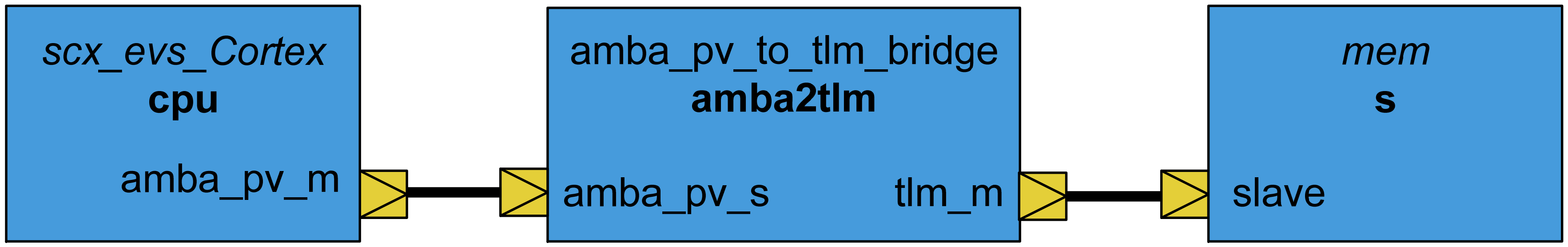
Our Processor

- Download & unpack cortexm0ds.tar.gz
- Two subtrees
  - v – Verilog RTL (and later, gate-level) simulation for ***ARM Cortex-M0 Design-Start*** processor
    - Obfuscated & stripped of debug support
    - Detailed enough to demonstrate cycle-accurate simulation (and later, the entire physical design flow)
  - sc – SystemC / TLM 2.0 simulation for ***ARM Fast Models for the Cortex-M0*** Processor
    - Can run binaries compiled for Cortex-M0
- **Our Goal:** Compare Simulation Times and cycles/sec performance of RTL & TLM

TLM Simulation

- Contents
  - » fibonacci.c program (image directory)
    - – identical to version in RTL tree
  - » ARM System Generator Canvas (SGC) source to build simulations w/ Cortex-M0 Fast Models (sgc directory)
  - » SystemC / TLM 2.0 source code (sgc/src directory)
- Usage
  - » “cd image; make” to build image
  - » “cd ../sgc; make” to build simulation
  - » “make sim” to simulate
  - » “make dbg; make gdb” to debug

SystemC / TLM 2.0 System



- SGC-generated module ***scx_evs_Cortex***
  - Contains one *AMBA-PV-style* initiator socket
- AMBA-PV-to-TLM Bridge (provided by ARM)
  - Translates AMBA-PV to generic transactions
  - Contains one generic initiator socket
- Memory module with a generic target socket
  - Models 1.5 GB Memory (all addrs up to 0x60000000)

sc/sgc/src/main.cpp Code

```
int sc_main(int argc , char * argv[]) {
    // Initialize simulation
    scx::scx_initialize("Cortex");
    // Components
    amba_pv::amba_pv_to_tlm_bridge<64> amba2tlm("amba2tlm");
    mem s("Memory", 0x60000000);
    scx_evs_Cortex cpu("Cortex");
    // Simulation configuration
    scx::scx_parse_and_configure(argc, argv);
    // Bindings
    cpu.amba_pv_m.bind(amba2tlm.amba_pv_s);
    amba2tlm.tlm_m.bind(s.slave);
    // Start of simulation
    sc_core::sc_start();
    std::cout << "Simulation Time: " << sc_core::sc_time_stamp() << std::endl;
    return EXIT_SUCCESS;
}
```

Simulation Quantum

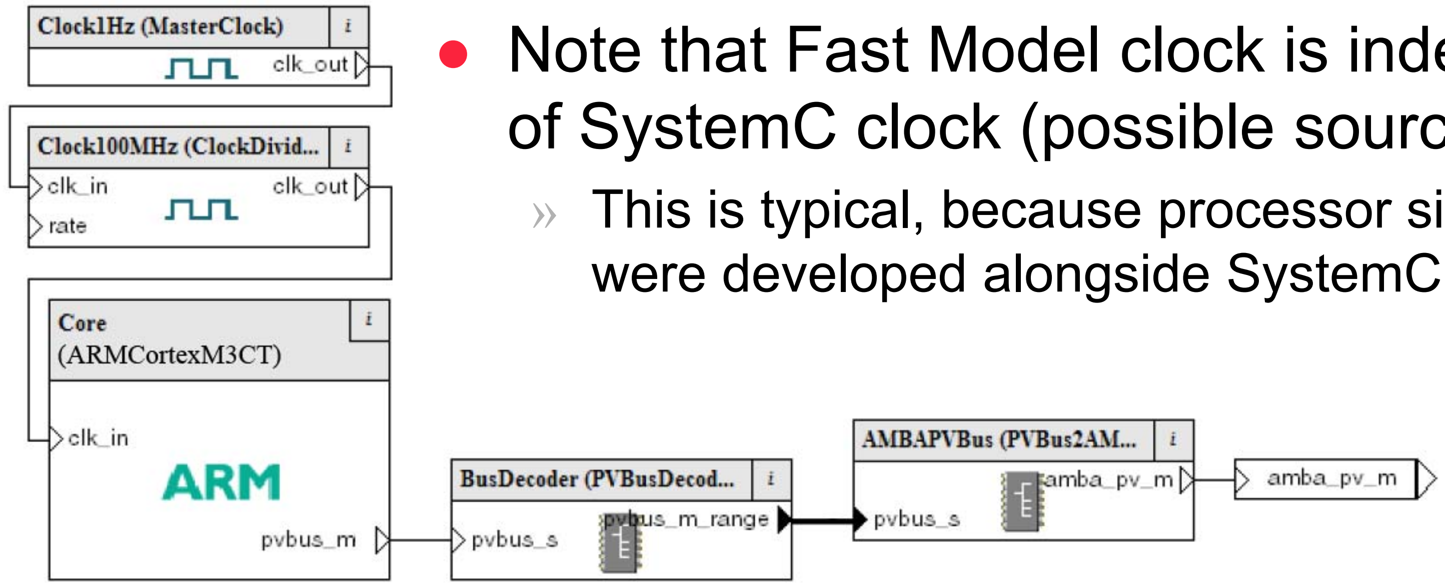
- TLM 2.0 Offers a mechanism to allow modules to run “out of sync” with each other
- Do this by defining the “Quantum”, which is the simulation time to wait before synchronizing modules
- To use, uncomment the following lines:

```
double instructions_per_quantum = 10000.0;
tlm::tlm_global_quantum::instance().set(
       sc_core::sc_time(instructions_per_quantum/100000000,sc_core::SC_SEC));
```

- Note: Alters the behavior (no. of instructions & transactions) significantly
- Why is this useful?

scx_evs_Cortex Module

- System pictured here illustrates the contents of the SGC System
  - see sgc/Cortex-M0.lisa for details
  - see "SystemC Export" chapter of *Fast Models User Guide (FMUG)* for more details
- Note that Fast Model clock is independent of SystemC clock (possible source of error)
  - This is typical, because processor simulators were developed alongside SystemC / TLM 2.0



Other Notes

- Console writes (to 0x40000000) are sent to stdout by mem module
- Define XACT_DUMP macro in mem.cpp to dump transactions
- Address statistics printed automatically by *mem* module (no need to dump transactions)
- mem module ends simulation when end-of-file character 0x0D is printed to console

TLM Simulation Output

```
starting...
fib(0) = 0
fib(1) = 1
fib(2) = 1
fib(3) = 2
fib(4) = 3
fib(5) = 5
finishing...
SystemC: simulation stopped by user.
Simulation Time: 111800 ns
0.75user 0.72system 0:01.43elapsed 19%CPU (0avgtext+0avgdata
6422960maxresident)k
0inputs+0outputs (5140major+9307minor)pagefaults 0swaps
```

- How does this differ with RTL sim?
- What is the cycles/sec performance?

Address Statistics

```
STATISTICS for addr range 0x0 to 0xffff
Number of Blocking Reads:  5160
Number of Blocking Writes: 15
Minimum address accessed:  0x0
Maximum address accessed:  0x8454
STATISTICS for addr range 0x10000 to 0x1fffffff
Number of Blocking Reads:  801
Number of Blocking Writes: 410
Minimum address accessed:  0x1ff28
Maximum address accessed:  0x1fff8
STATISTICS for addr range 0x20000000 to 0xffffffff
Number of Blocking Reads:  0
Number of Blocking Writes: 92
Minimum address accessed:  0x40000000
Maximum address accessed:  0x40000000
```

- How does this differ from RTL sim?

Cycles-per-Instruction

- How would you expect that the Cycles-per-Instruction (CPI) to differ between RTL and TLM simulations?

- How could you model this difference?

Today's Lecture

- Transaction-Level Modeling of Busses
- Transaction-Level Modeling
  with Instruction Set Simulation (TLM+ISS)
- → Alternative Modeling Approaches

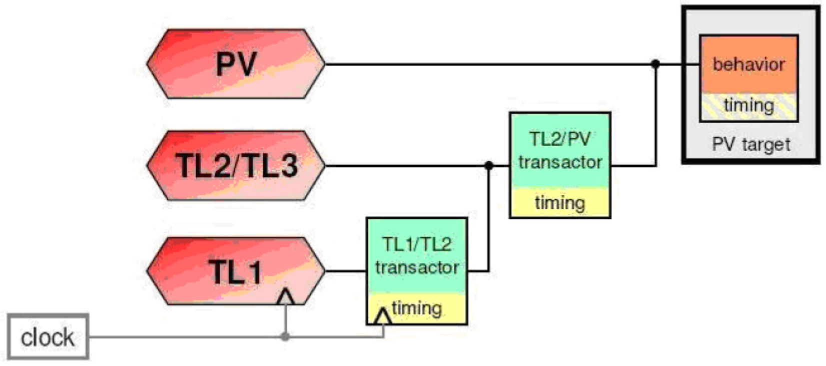
Terminology

- Programmer's View (PV)
  - A register-accurate representation of the system sufficient to write low-level software drivers and specify RTL behavior
- Programmer's View with Time (PV+T)
  - Functionally identical to PV model
  - Annotated with Timing (and Power) information to allow performance analysis without authoring new RTL

- See [Donlin, CODES’04] for more explanation of these terms
- These terms are no longer in fashion, rather people tend to use LT for PV and AT for PV+T
- We will focus on PV/LT for now

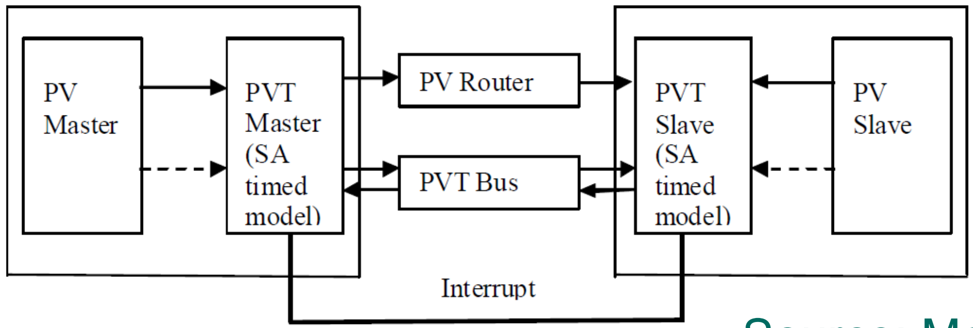
Published PV+T Approaches

- Open Core Protocol Partnership [OCP-IP 2007]



Source: OCP-IP © 2007

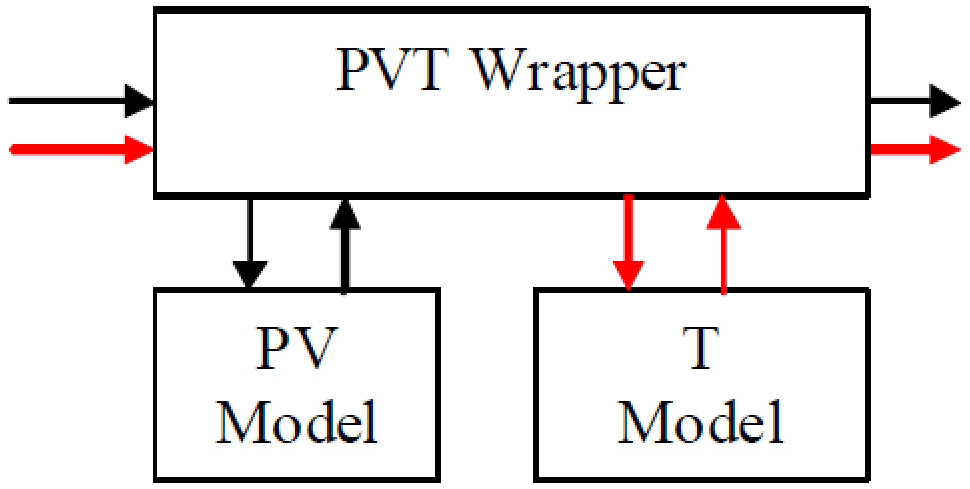
- ST [Ghenassia 2005]



Source: Model Builder Modeling Guide
© 2009 Mentor Graphics

- It remains to be seen which style will prevail

The Vista Model Builder Approach



Source: Model Builder Modeling Guide
© 2009 Mentor Graphics

- PV transactions run parallel to T transactions
- T implements a "policy engine"
  - initiates/responds to T xacts & associates them with PV xacts
  - looks for a T xact & policy that are the "cause" for each PV xact

References

[1] Donlin, CODES+ISSS'04

[2] Open Core Protocol International Partnership (www.ocp-ip.org) 2007, acquired by Accellera in 2013.

[3] Ghenassia, ed., "Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems", Springer 2005