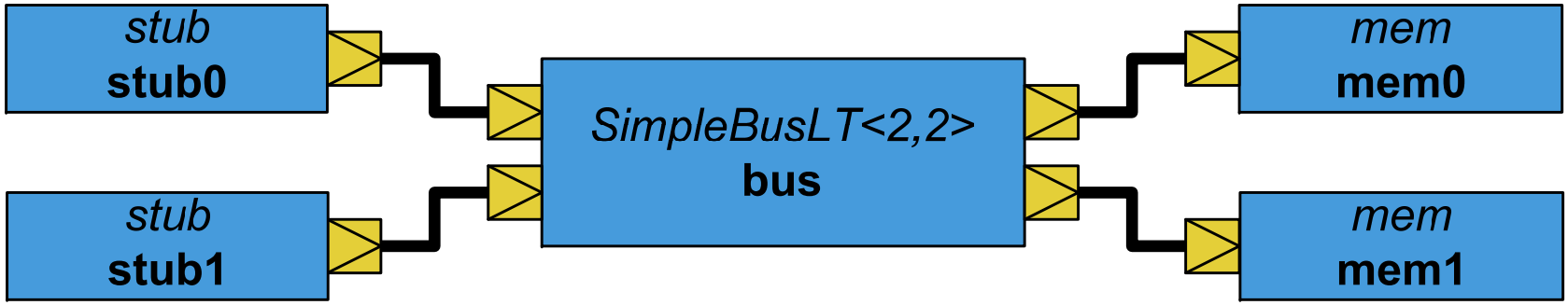
Digital Integrated Circuit Design

Module TLMSock

Transaction Level Modeling: Sockets

W. Rhett Davis

Example System



- CPUs modeled as *stubs* (initiators) that generate write and read transactions and send them to the memories (targets)
- Generic payload is passed between modules through a port-abstraction called a ***socket***
- See the Loosely Timed Transaction-Level Modeling Example (sc_tlm_tut.tar.gz) on the course web-page

Sockets

- Combined interfaces
  - » implements
    - – 3 transport interfaces: blocking, non-blocking, & debug
    - – direct memory interface
  - » two types
    - – forward (request) (tlm_fw_transport_if)
    - – backward (response) (tlm_bw_transport_if)
- Sockets
  - » implement both forward and backward combined interfaces

Normal Sockets

- Initiator Socket (tlm_initiator_socket)
  - » port for forward (request) path
  - » export for backward (response) path
    - – module must be derived from tlm_bw_transport_if
    - – module must implement nb_transport_bw() & invalidate_direct_mem_ptr()
    - – may implement b_transport() or transport_dbg() if desired
- Target Socket (tlm_target_socket)
  - » port for backward (response) path
  - » export for forward (request) path
    - – module must be derived from tlm_fw_transport_if
    - – module must implement nb_transport_fw() & get_direct_mem_ptr()
    - – may implement b_transport() or transport_dbg() if desired

Convenience Sockets

- Some modules require complex use of sockets
  - e.g. Multiple initiators and targets
  - What is an example of such a module?
- Convenience sockets are special sockets that allow registering callbacks
  - sockets can be used without deriving module from an interface class, no required methods
  - *e.g.* simple_target_socket_tagged, simple_initiator_socket_tagged
- This functionality is not well developed, so these classes are not considered "interoperable"
  - Included in the tlm_utils namespace, rather than tlm namespace

Parts of a TLM SystemC Module

- Declaration: class <*module_name*>*: sc_module*
  - » Port, Socket & Internal State-Variable declarations
  - » Instance, Process, & Transport Method declarations
- Constructor
  - » Structural Functionality Description
    - – Instantiate lower-level modules, bind to channels/sockets
    - – Register Transport Methods (convenience sockets only)
  - » Procedural Functionality Description
    - – Register thread & method processes with scheduler
- Procedural Functionality Description
  - » Thread & Method Process Definitions
  - » Transport Method Definitions

Initiator Socket Example

```
class stub: public sc_core::sc_module,
   virtual public tlm::tlm_bw_transport_if<>
   //defaults to 32-bit bus width if not specified
...
tlm::tlm_initiator_socket<> master;
...
void invalidate_direct_mem_ptr(sc_dt::uint64 start_range,
                               sc_dt::uint64 end_range);

tlm::tlm_sync_enum nb_transport_bw(
       tlm::tlm_generic_payload &gp,
       tlm::tlm_phase &phase, sc_core::sc_time  &delay);
};
```

Convenience Socket Example

```
class mem: public sc_core::sc_module
...
bool get_direct_mem_ptr(
       tlm::tlm_generic_payload &payload,
       tlm::tlm_dmi             &dmi_data);
...
tlm_utils::simple_target_socket<mem> slave;
...
void custom_b_transport(tlm::tlm_generic_payload &gp,
       sc_core::sc_time  &delay);
...
};
```

Multiple Socket Example

```
template <int NR_OF_INITIATORS, int NR_OF_TARGETS>
class SimpleBusLT : public sc_core::sc_module
{
...
 target_socket_type target_socket[NR_OF_INITIATORS];
 initiator_socket_type initiator_socket[NR_OF_TARGETS];
 SC_HAS_PROCESS(SimpleBusLT);
  SimpleBusLT(sc_core::sc_module_name name) :
    sc_core::sc_module(name)  {
     for (unsigned int i = 0; i < NR_OF_INITIATORS; ++i) {
       target_socket[i].register_b_transport(this,
                    &SimpleBusLT::initiatorBTransport, i);
       target_socket[i].register_transport_dbg(this,
                    &SimpleBusLT::transportDebug, i);
       target_socket[i].register_get_direct_mem_ptr(this,
                    &SimpleBusLT::getDMIPointer, i);
```

Convenience Socket Example

```
    for (unsigned int i = 0; i < NR_OF_TARGETS; ++i) {
       initiator_socket[i].
                 register_invalidate_direct_mem_ptr(this,
                 &SimpleBusLT::invalidateDMIPointers, i);
    }
}

void initiatorBTransport(int SocketId,
                         transaction_type& trans,
                         sc_core::sc_time& t)
```

- Transport calls are the same, except that the first argument is the socket index

Digital Integrated Circuit Design

Module TLMSock
Transaction Level Modeling: Sockets

W. Rhett Davis

Thanks for watching