Digital Integrated Circuit Design

Module TLMIntro

Transaction Level Modeling: Introduction

W. Rhett Davis

Why a New Class Library?

- Although SystemC gives the basics of what is needed for TLM, the complexity of coding processor-based systems places it well beyond the capability of most designers

- The Open SystemC Initiative (OSCI) TLM class library is an extension of SystemC to simplify modeling of processor-based systems

OSCI TLM History

- 2002 – SystemC 2.0 released
- 2005 – TLM 1.0 released
- 2006 – SystemC 2.2 (IEEE 1666-2005) released
- 2008 – TLM 2.0 released
- 2009 – TLM 2.0.1 released
- 2012 – SystemC 2.3 released (incorporates TLM 2.0.1)

Generic Payload

- The Generic Payload is a class that encapsulates most of the information needed to pass any transaction through a memory-mapped bus
- Attributes:
  - Command (read, write, ignore)
  - Address
  - Data Length (total size of transaction, in bytes)
  - Streaming Width (bus width, size per bus cycle, in bytes)
  - Response Status (ok, incomplete, address error, etc.)
  - Data Pointer
- Allow reuse of code in a way that is bus-independent
- How does the generic payload increase simulation speed?

Generic Payload Usage

```
tlm::tlm_generic_payload  *gp = new tlm::tlm_generic_payload;
unsigned char *dp;
static const unsigned int data_size = 4;
sc_dt::uint64 address = 0x0000000010000100;
unsigned int data = 0xFFFFFFFF;
dp = gp.get_data_ptr();
*reinterpret_cast<unsigned int*>(dp)=data;
gp->set_command( tlm::TLM_WRITE_COMMAND );
gp->set_address( address );
gp->set_data_length( data_size );
gp->set_streaming_width( data_size );
gp->set_response_status( tlm::TLM_INCOMPLETE_RESPONSE );
```

- Three namespaces: tlm, sc_core, & sc_dt
- Code above creates a one-word write transaction

Transport Interfaces (1)

- Four types of interfaces
  - » Blocking Transport
    - – waits until complete, plus annotated time
    - – used only in thread processes

```
sc_core::sc_time delay(10,sc_core::SC_NS);
b_transport(*gp,delay);
```

  - » Non-Blocking Transport
    - – returns immediately
    - – phase and status used to synchronize processes

```
tlm::tlm_phase phase = tlm::BEGIN_REQ;
tlm::tlm_sync_enum status;
status = nb_transport_fw(*gp,phase,delay);
```

Transport Interfaces (2)

- Debug Transport
  - » allows communicating through a channel with no delays
  - » handy for setting up an initial state or checking the current state in a test-bench

```
unsigned int bytes_moved;
bytes_moved = transport_dbg(*gp);
```

- Direct Memory
  - » allows a target to give a pointer to a initiator so that its memory can be modified directly
  - » Much, much faster simulation (e.g. Mentor Graphics Seamless)

```
void invalidate_direct_mem_ptr(sc_dt::uint64 start_range,
                               sc_dt::uint64 end_range);
...
bool get_direct_mem_ptr(tlm::tlm_generic_payload &payload,
                        tlm::tlm_dmi             &dmi_data);
```

Digital Integrated Circuit Design

Module TLMIntro
Transaction Level Modeling: Introduction

W. Rhett Davis

Thanks for watching