bus-slave-if
Listen (idp, addr, &len)
Acknowledge()
Send Read Data (data)
Receive Write Data (data)

Slave Component Main Thread:

while (1) {
    Listen();
    ① Check to see if request for this device
    L→ ② Acknowledge()
    L→ ③ call SendReadData() or ReceiveWriteData()
         for required data transfer
}

Master Component Main Thread:

... ① Request()
② WaitForAcknowledge() and check if request successful
③ Call WriteData() or ReadData for requested data transfer
TLM Simulation Principles:

- Transactions are called by sc-thread/sc-behavior with sc-module with corresponding port.
- Implementation is defined in sc-module/sc-channel implementing the interface.
- Timing and event processing can affect the calling sc-module/sc-thread/sc-method.

Note: channels can also have threads.

Bus Component Implementation:

![Diagram of bus component]

1) Waiting for acknowledgment
2) Waiting for data
3) Waiting for requests from master
4) Waiting for active bus request
5) Waiting for data

sc-events: Need events for synchronizing execution between threads.
Request/Acknowledge Sequence:

Master:
- Send request
- Wait for acknowledge
- Notified slave
- Wait for response from slave
- Notify master

Arbiter (Bus):
- Wait for request
- Notified slave at request
- Wait for acknowledge from slave
- Notify master

Slave:
- Wait for active bus request
- Notify arbiter request received and valid

Transfer Sequence (write operation):

Master:
- Send data
- Wait for data to be received
- Notified slave
- Wait for data to be received

Slave:
- Wait for data
- Notified master
- Data received
- Wait for data
- Notified master
- Data received

Arbiter (Bus):
- Wait for data transfer
  (monitor transfer & known when burst transfer & complete)
Transfer sequence (load operation):

**Master:**
- wait for
- data
- notify
- servant
- data
- received

**Slave:**
- send data
- wait for
- notify
- master
- data to
- be received

**Master (Mem):**
- wait for data transfer

---

**System C Events:**

* Notify on event will only be seen by thread waiting on that event.*

* Knowledge of simulation engine is needed to ensure correct modeling behavior.*
// Bus Master Interface
class bus_master_if : virtual public sc_interface
{
  public:
    virtual Request() = 0;
    virtual WaitForAcknowledge() = 0;
    virtual ReadData() = 0;
    virtual WriteData() = 0;
};

// Bus Servant Interface
class bus_servant_if : virtual public sc_interface
{
  public:
    virtual Listen() = 0;
    virtual Acknowledge() = 0;
    virtual SendReadData() = 0;
    virtual ReceiveWriteData() = 0;
};