

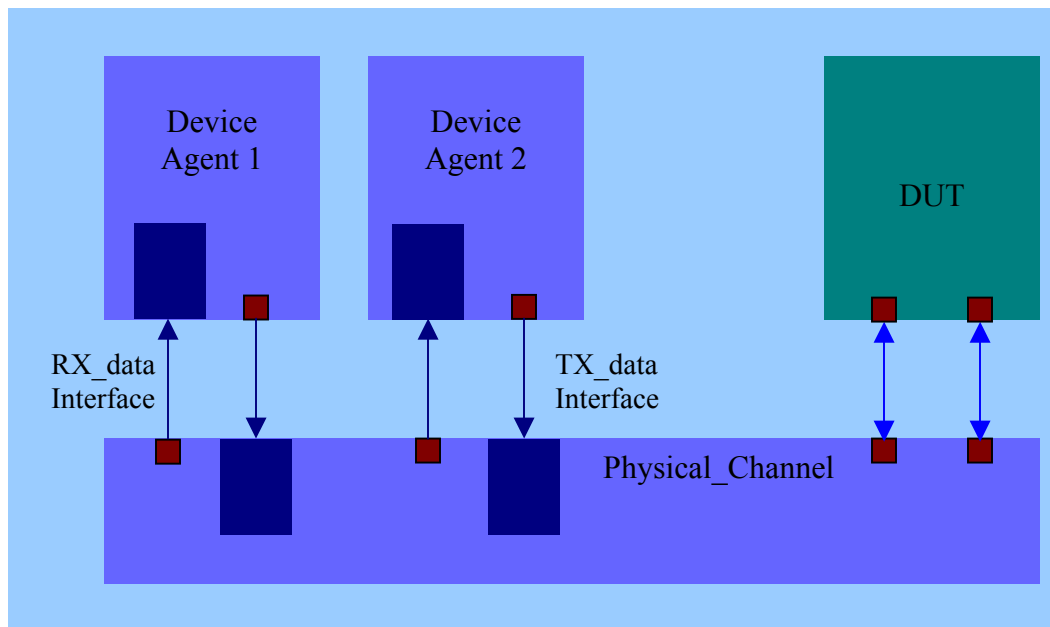
Modeling interface-channel in SystemC for Multi-agent Verification Environments

For the parallel bus protocols such as PCI and AMBA, it is the toughest task for the designers to verify the device, since it requires to model multi-agent environment.

Using interfaces and a channel, engineers can use a configurable-behavioral model of device agents as shown in the figure below and instantiate multiple device agents. This helps in verifying complex designs and find even typical corner case bugs. There are two apparent advantages of this kind of interface-channel implementation.

- 1) Avoids formal connections of each device agents and their manual connections
- 2) Provides a connection facility for DUT at each level of hierarchy in the design.

Multi-agent Verification Environment



The code below describes physical interface of agents to model multi-agent verification environments in SystemC

```
//Receive interface of Device
Class RX_data_if : sc_interface
{
    Virtual Received_data_indication(sc_uint<8> data_byte ) = 0;
}

//Transmit interface of Device
Class TX_data_if : sc_interface
{
    Virtual Transmit_data_request(sc_uint<8> data_byte ) = 0;
}
```

Considering the data flow, the receive and transmit interfaces are implemented in respective entities as shown below:

```
//Channel inheriting and implementing only TX_data_if interface
Class Physical_Channel : public TX_data_if
{
}
```

```
sc_port<RX_data_request> RX_port;  
sc_port<RX_data_request> RX_DUT_port;  
sc_port<TX_data_request> TX_DUT_port;
```

```
    Transmit_data_request(sc_uint<8> data_byte )  
    {  
        //send 'data_byte' to other agents or do requisite processing;  
    };  
};
```

//Device agent inheriting and implementing only RX_data_if interface

```
Class Device : public RX_data_if  
{  
    sc_port<TX_data_request> TX_port;  
  
    Received_data_indication(sc_uint<8> data_byte )  
    {  
        //do processing on received 'data_byte';  
    };  
};
```

This model can be expanded to include 'n' number of agents in Verification Environments. This kind of implementation requires skillful implementation of Physical Channels.

The most useful feature out of this implementation is that user can configure as many agents as he wants in his verification environment without getting into complexity of interfacing each agent separately.