# CSCE 616 – Hardware Design Verification

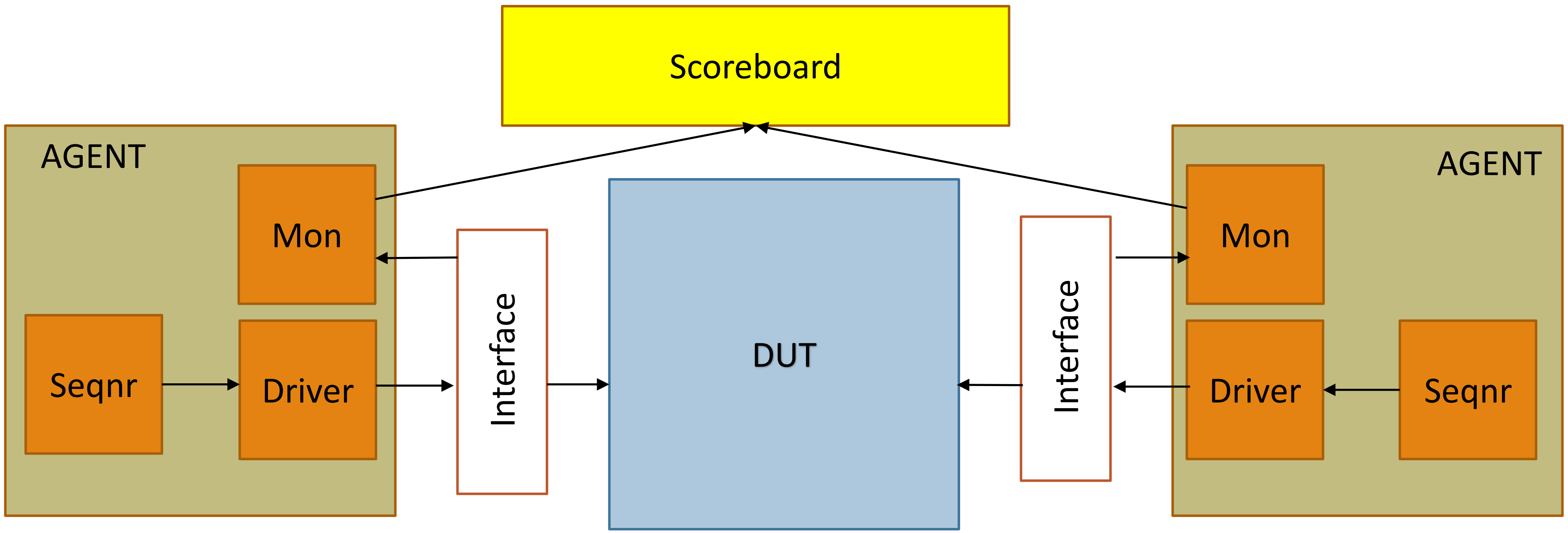
# Lab – 7  HTAX TX Driver and Monitor

--SAUMIL GOGRI

# Objective – Write TX Driver and Monitor code

- UVM Driver
- TX Driver guideline
- UVM Monitor Packet
- UVM Monitor
- TX Monitor guideline

# Big Picture


Scoreboard
AGENT
Mon
Seqnr
Driver
Interface
DUT
Interface
AGENT
Mon
Driver
Seqnr

# UVM Driver

Driver is an **active** component which drives the transaction/packet on the signals of a particular interface in the design.

Driver makes sure that interface protocols are adhered.

**UVM Driver Methods:**

- get_next_item(req)  - This method blocks until a REQ sequence_item is available in the sequencer.
- item_done() - The non-blocking item_done() method completes the driver-sequencer handshake and it should be called after a get_next_item()

Drive Method: Add driving logic i.e. after getting the seq_item, drive it to DUT signals in the run_phase().

# UVM Driver

```
// run phase
 virtual task run_phase(uvm_phase phase);
   forever begin
   seq_item_port.get_next_item(req);
    ......
    .. driving logic ..
    ......
   seq_item_port.item_done();
   end
 endtask : run_phase
```

# TX Driver guideline

- Wait for pkt.delay.
- Set destination port i.e. set value of port tx_outport_req depending on value of pkt.dest_port. Remember it is one hot encoded signal.
- Request for VC using tx_vc_req port from pkt.vc.
- Wait for grant for VC request.
- As soon as grant is given deassert both requests. Set tx_sot for VC granted. And start driving data on tx_data.
- Wait and then make tx_sot =0.
- Release gnt when second last payload data is transferred.
- When beginning transfer of last data, assert tx_eot.

# UVM Monitor Packet

We need to recreate the monitor transaction/packet from the interface signals.

This packet is then sent to scoreboard for external checking

In most of the case, monitor packet is a subset of transaction packet i.e. we have fewer fields in monitor packet as compared to transaction packet

| Transaction Packet | Monitor Packet |
|---|---|
| • delay<br>• dest_port<br>• vc<br>• length<br>• data[] | • dest_port<br>• data[] |

# UVM Monitor

- Monitor is a **passive** component which captures the signal activity from interface in the design and translates into transaction level object to send it to scoreboard or other components

- Essentially UVM Monitor collects bus or signal information through the virtual interface and exports it in form of Monitor packet via **analysis port**.

- Declare analysis port

```
uvm_analysis_port  #(my_data) mon_analysis_port;
```

- Run Phase

```
virtual task run_phase (uvm_phase phase);
       <<create mon_packet from interface activity>>
       mon_analysis_port.write(mon_packet);
endtask
```

# TX Monitor guideline

- Wait till tx_vc_gnt
- Set dest_port of Monitor packet from tx_outport_req
- Start collecting data from tx_data and append it in Monitor packet's data field
- After the tx_eot, write the monitor packet on analysis port (for this lab we are only printing the monitor packet)

# SV Dynamic Array Resizing

```
//memory allocation
d_array1 = new[4];  //dynamic array of 4 elements
//array initialization
d_array1 = {0,1,2,3};

// change the length of the array after declaration/initialization
d_array1 = new[10];  //dynamic array of  10 elements

//allocate 6 new elements and retain values of 4 elements.
d_array1 = new[10](d_array1);
```

Source: (https://www.verificationguide.com/p/systemverilog-dynamic-array.html)

# Thank you