CSCE 616 – Hardware Design Verification

Lab – 5 Stimulus

Objective – Write Sequences and Test for HTAX

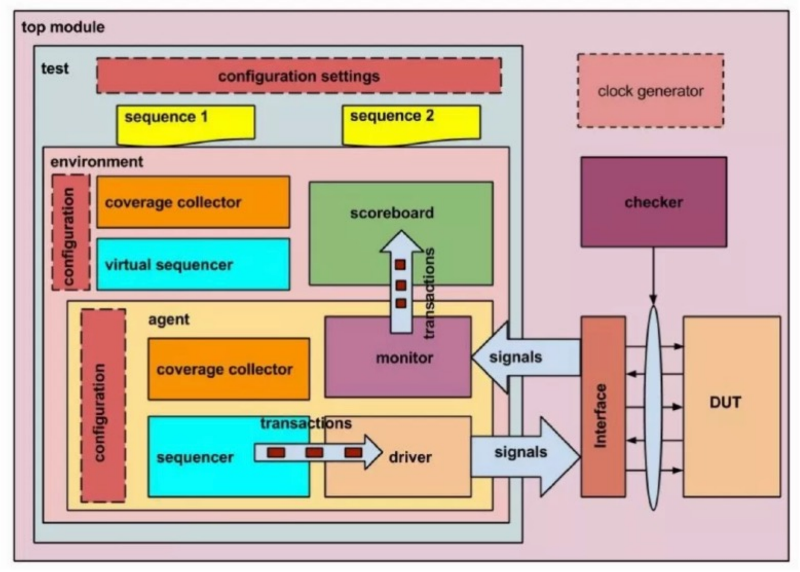
- UVM Topology
- Abstract view of Test, Sequence and Sequencer
- UVM Sequence
- UVM Test

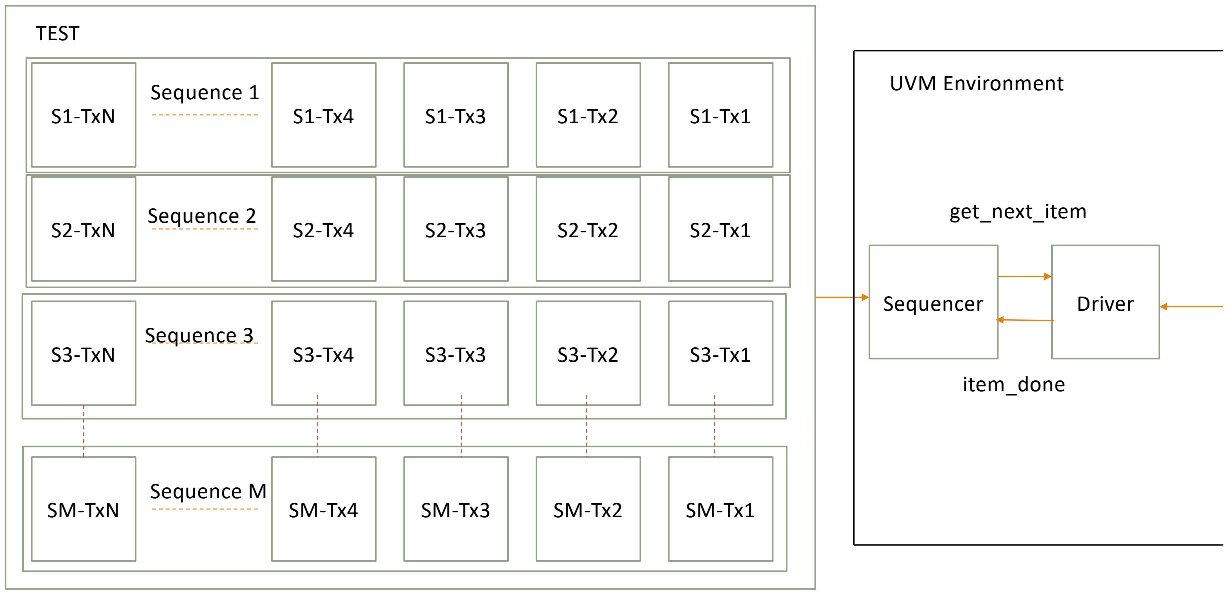
UVM Topology

UVM Objects
- Sequences
- Sequence item (packet)

UVM Components
- Test
- Sequencer
- Driver
- Monitor
- Agent
- Environment
- Scoreboard

Independent
- Interface and checkers





UVM Base-Sequence

```
//Base sequence
class htax_base_seq extends uvm_sequence #(htax_packet_c); //HTAX packet class is passed as parameter

  parameter PORTS = `PORTS;
  parameter VC    = `VC;
  parameter WIDTH = `WIDTH;

        //Factory Registration
        `uvm_object_utils(htax_base_seq)

        //Constructor
        function new ( string name = "htax_base_seq");
                super.new(name);
        endfunction : new

endclass : htax_base_seq
```

UVM Sequence

```
//Sequence 1
//Fix Destination Port Sequence extends from base sequence
class fix_dest_port_seq extends htax_base_seq;

        //Factory Registration
        `uvm_object_utils(fix_dest_port_seq)

        //Constructor
        function new ( string name = "fix_dest_port_seq");
    super.new(name);
  endfunction : new

        //Body task --
        virtual task body();
                int i;
                i=$urandom_range(0,3);
                `uvm_info(get_type_name(),"Executing fix destination port sequence with 5 transactions", UVM_NONE)
                //Generate a sequence with 5 packets
                repeat(5) begin
                        `uvm_do_with (req, {req.dest_port==i;}) //fix dest_port
                end
        endtask

endclass : fix_dest_port_seq
```

UVM Test

Sequence Handle
fix_dest_port_seq fix_dest_port_seq_; //fix_dest_port sequence handle

Instanciation in **Build Phase**
fix_dest_port_seq_ = fix_dest_port_seq::type_id::create("fix_dest_port_seq_"); //UVM instance of sequence using factory

Starting Sequence in **Run Phase**
fix_dest_port_seq_.start(sequencer); //Starting sequence on sequencer

Additional Notes from previous Lab Lectures:

System Verilog Implication constraints (→)

The implication operator can be used to declare conditional constraints.

expression → constraint

For e.g. If value of a==1 then value of is b greater than 10

Constraint a_and_b {(a==1) → (b >10);}

Thank you