LUCID VLSI

Learning Unlimited. Coding InDelible.

# 1 SystemVerilog LAB1



*After completing this lab, you should be able to:*

- *Understand the design specification.*
- *Use structures, dynamic arrays and queues.*
- *Implement user defined data types using typedef.*
- *Implement functions/tasks with structure as argument.*
- *Use ref direction with methods.*
- *Generate and drive stimulus into DUT.*
- *Verify DUT behavior with waveform.*

LUCID VLSI

Learning Unlimited. Coding InDelible.

*LAB1:*

*Step 1: Instantiate dut (router_dut) in testbench along with ports connections.*

1. *Open testbench.sv .*
2. *Define required signals(clk,reset,dut_inp.....) for port connections. Refer specification to know the required signals(I/O Pins). Add this code in Section 1 in testbench.sv.*
3. *Instantiate router_dut with instance name dut_inst along with port connections. Use name based port mapping.*
4. *Add the below code in Section 2 in testbench.sv.*

```
router_dut  dut_inst (.clk(clk),.reset(reset),.....);
```

*Step 2: Generate clock and reset in testbench.*

1. *Initialize clk to 0 in initial block and generate clock.*
2. *Add the below code in Section 3 in testbench.sv*

```
initial clk=0;
always #5 clk = ~clk;
```

3. *Write task named apply_reset to apply reset to DUT.*
4. *Add the below code in Section 5 in testbench.sv*

```
task apply_reset ();
$display ("[TB Reset] Applied reset to DUT");
reset<=1;
repeat (2) @(posedge clk);
reset<=0;
$display ("TB Reset] Reset completed");
endtask
```

LUCID VLSI

Learning Unlimited. Coding InDelible.

Step 3: Create Packet type with "Packet format" in TB (refer topic2 slides).

1. Use structure to create packet format (refer topic2, slide 13 and dut specification). Add this code in Section 4 in testbench.sv
2. Define the packet (from above step) as user defined structure type using typedef. Add this code in Section 4 in testbench.sv
3. Define payload using dynamic array
   (refer topic2 slides, slide13 ).



Step 4: Write a function to generate stimulus.

1) Define void function with packet type (from step3.2) as argument with ref direction.
2) Add the below code in Section 5 in testbench.sv

```
function automatic void generate_stimulus (ref packet pkt);
```



3) Inside generate_stimulus method, do the following:
   1) Generate random stimulus on pkt.sa,pkt.da fields of packet. (refer topic2 slides, slide13 ).
   2) Generate random payload sizes (from 2bytes-to-1990 bytes). (refer topic2 slides, slide13 ).
   3) Assign complete packet length to "Length" field of packet. (refer topic2 slides, slide13 ).
   4) Assign sum of payload elements to "crc" fled of packet. (refer topic2 slides, slide13 ).

LUCID VLSI

Learning Unlimited. Coding InDelible.

## Step 5: Drive stimulus into DUT (Design Under Test).

1) Write a task named drive with packet as argument with input direction.
2) Add the below code in Section 5 in testbench.sv

```
task drive ( input packet pkt );
```

3) Inside drive method, do the following

1. Wait for buys signal to become 0.
   Ex: wait (busy==0);
2. Assert inp_valid when driving first byte of the packet (assign value 1 to inp_valid).
   Ex: @(posedge clk);
   inp_valid<=1; //Start of Packet
3. Drive complete packet into dut (8-bits/clock) by waiting on posedge clk.
   Ex: dut_inp<=pkt.sa;
   @(posedge clk);
   dut_inp<=pkt.da;
   //drive rest of the packet in the same manner (refer topic2 slides, slide13 ).
4. De-assert inp_valid for the last byte of the packet (assign value 0 to inp_valid).
   Ex: @(posedge clk);
   inp_valid<=0; //End of Packet

LUCID VLSI

Learning Unlimited. Coding InDelible.

## Step 6: Write initial block to start the verification flow.

1) Implement initial block and call all the methods you have implemented so far.
2) Add the below code in Section 6 in testbench.sv

```
packet   stimulus_pkt;
initial begin
        apply_reset();
        generate_stimulus(stimulus_pkt);
        drive(stimulus_pkt);
        repeat(5) @(posedge clk);

        //Wait for dut to process the packet and to drive on output
        wait(busy==0);
        repeat(10) @(posedge clk);
        $finish;
end
```



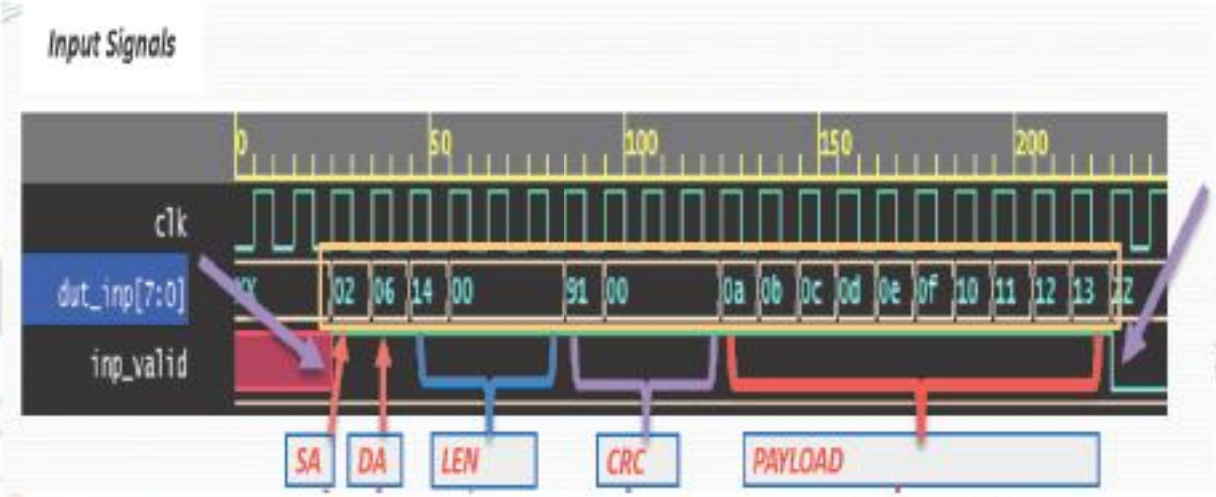
## Step 7: Validate the output of DUT with waveform.

1. Use $dumpvars to dump waveform.
   Add the below code in Section 7 in testbench.sv

```
initial begin
        $dumpfile("dump.vcd");
        $dumpvars(0,testbench.dut_inst);
end
```

2. Add signals dut_inp,inp_valid,outp_valid and dut_outp to waves.
3. Compare input packet driven into dut with the output packet from dut and check if it is matching or mismatching.
4. If all packets match, then test is "Passed" or "Failed".

Reference input waveform:






Reference output waveform: