LUCID VLSI

Learning Unlimited. Coding InDelible.

2

SystemVerilog LAB2

After completing this lab, you should be able to:

Generate random stimulus.
Implement pack operations using streaming operator.
Understand pack operations to pack the stimulus.
Use functions to pack the stream.
Use queues as argument to functions/tasks.
Implement generic drive method to drive stimulus into DUT.
Verify DUT behavior with waveform.

LAB2:

Copy the lab1 code as lab2

Step 1: Pack the "generated stimulus" into 8-bit stream using streaming operator (refer topic3 slides).

1) *Define queue named inp_stream with 8-bit wide element in testbench (testbench.v). Add this code in* ***Section 4*** *in testbench.sv*

   Ex: bit [7:0] ***inp_stream****[$];*

2) *Write a function named "pack" to pack the stimulus. Add this code in* ***Section 5*** *in testbench.sv*

   Ex:

   function automatic void pack (*ref bit[7:0] q_inp[$], input packet pkt);*

   1) *Function with* ***queue type*** *as argument with* ***ref*** *direction*
   2) ***packet type*** *as argument with* ***input*** *direction.*

3) *Inside pack method Pack all the data of input packet into q_inp(ref argument) queue with 8-bit wide* ***using streaming operator*** *(refer topic2 slides (slide 14)).*

Step 2: Write a function to generate random stimulus.

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

```
function automatic void generate_stimulus (ref packet
pkt);
   pkt.sa= $urandom_range(1,8);
   pkt.da= $urandom_range(1,8);
   pkt.payload=new[ $urandom_range(10,20)];
   foreach(pkt.payload[i])
     pkt.payload[i]=i+1;

   pkt.len=pkt.payload.size() + 1+1+4+4;
   pkt.crc=pkt.payload.sum();
   $display("[TB Generate] Packet (size=%0d) Generated at
   time=%0t",pkt.len,$time);
endfunction
```

Step 3: Drive stimulus into DUT (Design Under Test).

1) Write a task named drive with queue type as argument with input direction. Add this code in **Section 5** in testbench.sv

   Ex: **task drive (const ref bit[7:0] inp_stream [$]);**
2) 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 content of inp_stream queue into dut (8-bits/clock) by waiting on posedge clk.*

   ```
   Ex: foreach(inp_stream[i]) begin
         dut_inp<=inp_stream[i];
         @(posedge clk);
        end
   ```

4. *De-assert inp_valid for the last byte of the packet (assign value 0 to inp_valid).*

   ```
   Ex:  inp_valid<=0; //End of Packet
   ```

Step 4: Write initial block to start the verification flow.

1) *Implement initial block and call all the methods you have implemented so far.*
2) *Example:*

```
 //Add this code in Section4 in testbench.sv
 packet stimulus_pkt;
 bit [31:0] pkt_count;
       //Add this code in Section 6 in testbench.sv
       initial begin
             pkt_count=5;
             apply_reset ();
             repeat(pkt_count) begin
             generate_stimulus(stimulus_pkt);
             pack(inp_stream,stimulus_pkt);
             drive(inp_stream);
             repeat(5) @(posedge clk);
             end
             //Wait for dut to process the packet and to drive
             on output.
             wait(busy==0);
             repeat(10) @(posedge clk);
             $finish;
       end
```

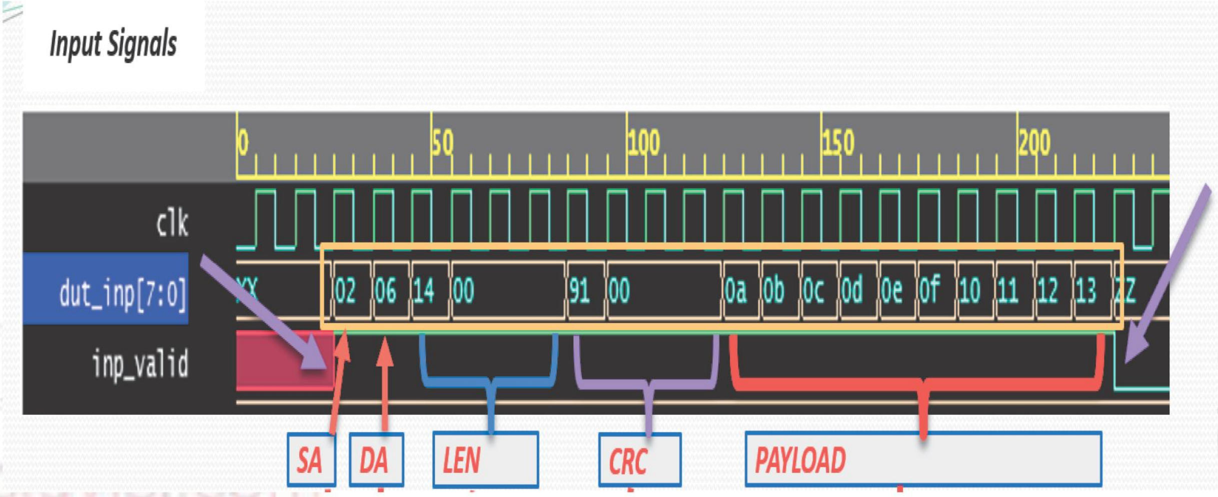
Step 5: Validate the output of DUT with waveform.

1. *Add below shown code to **Section 7** (testbench.sv) dump waveform.*

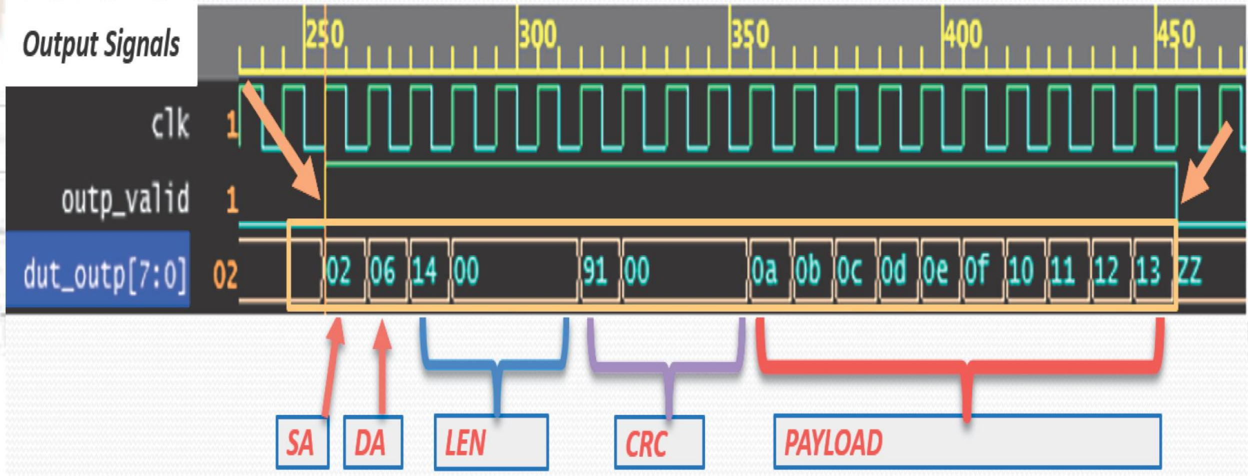
   ```
   Ex: initial begin
           $dumpfile("dump.vcd");
           $dumpvars(0,testbench.dut_inst);
       end
   ```

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

Reference input waveform:



Reference output waveform:


Output Signals
250
300
350
400
450
clk 1
outp_valid 1
dut_outp[7:0] 02
02
06
14
00
91
00
0a
0b
0c
0d
0e
0f
10
11
12
13
ZZ
SA
DA
LEN
CRC
PAYLOAD