Experiment 5: Sequence Generator

Deepankar Sehra, Roll Number 20D070024
EE-214, WEL, IIT Bombay
September 29, 2021

Overview of the experiment:

In two to three paragraphs, summarize

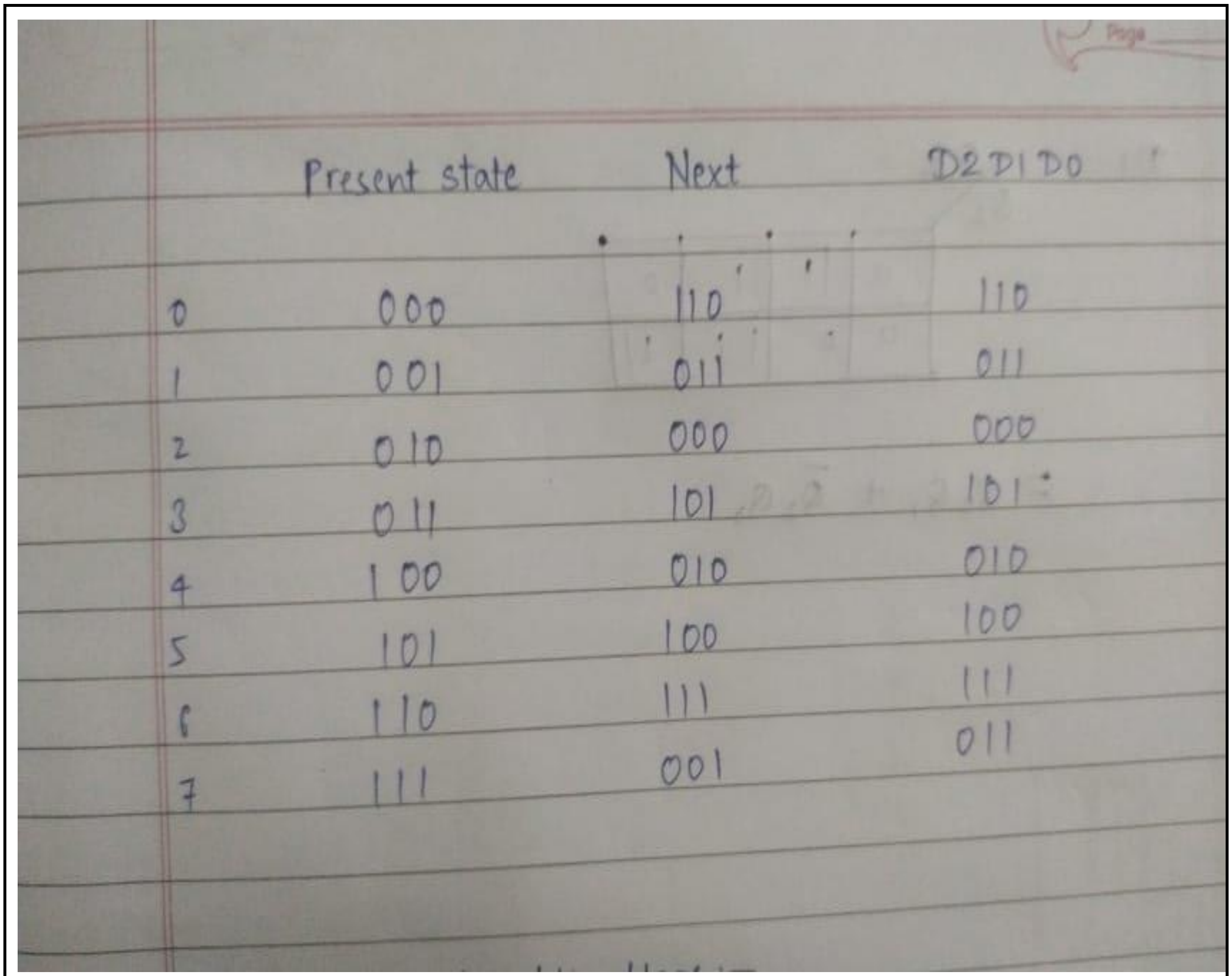
• The purpose of the experiment.
The purpose was to introduce flip flop behavior and implement in a sequential circuit. We had to make a sequence generator for a given sequence with the help of flip flops, with reset and clock inputs.

• What you did to perform the experiment.
We were given a skeleton code for the structural and behavioral for the sequence generator and flip flops, and had to make necessary changes in them to get our sequential circuit to work as expected.

• Organization of your report and a summary of the data you will be presenting.
I will be explaining the approach to the experiment with sufficient kmaps and truth tables, and attaching necessary screenshots of RTL viewer and simulations with scanchain.

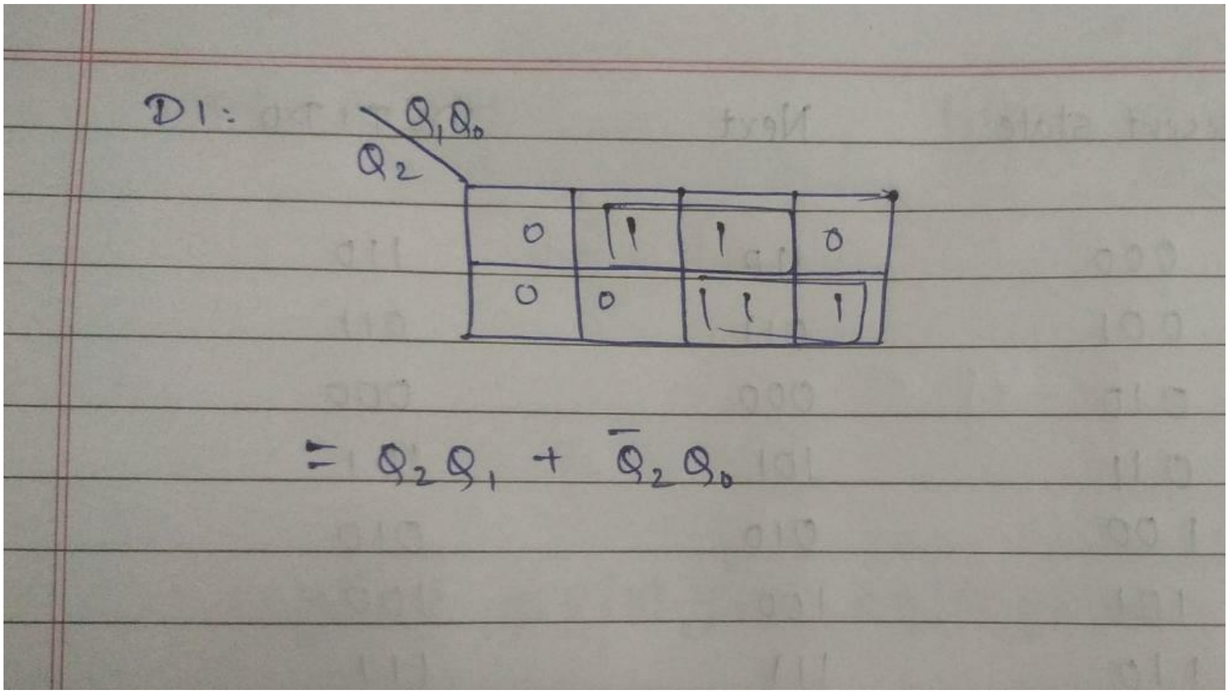
Approach to the experiment:

Firstly, wrote the states in a table to get an idea of the D-values of the 3 flipflops:



	Present state	Next	D2 D1 D0
0	000	110	110
1	001	011	011
2	010	000	000
3	011	101	101
4	100	010	010
5	101	100	100
6	110	111	111
7	111	001	011

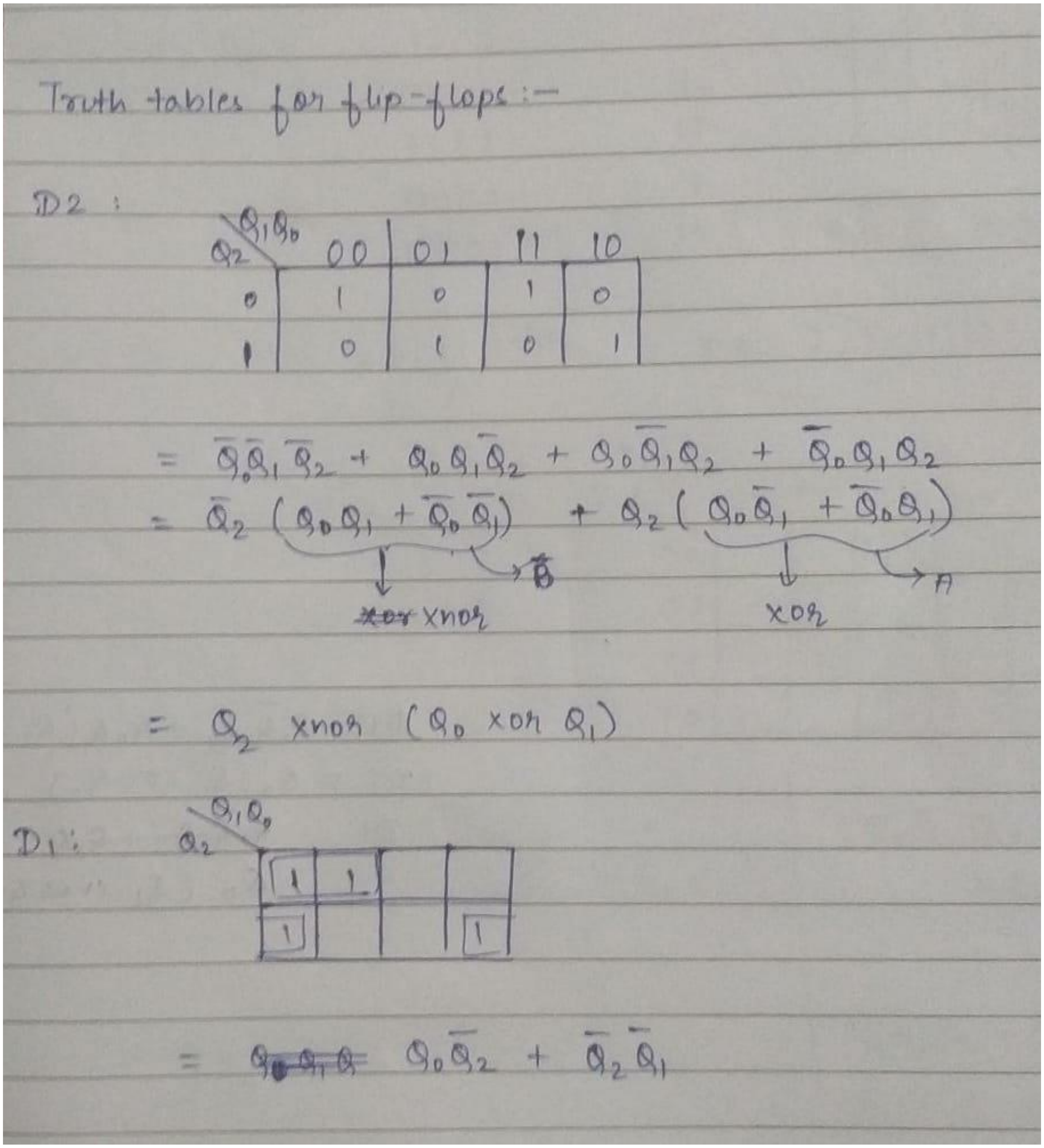
Then, with the found out values of D’s, tried minimizing logic with help of kmap as follows:



D1: $Q_2 \backslash Q_1Q_0$

0	1	1	0
0	0	1	1

$= Q_2Q_1 + \bar{Q}_2Q_0$



Truth tables for flip-flops :—

D2 :

Q_2 \ Q_1Q_0	00	01	11	10
0	1	0	1	0
1	0	1	0	1

$$= \bar{Q}_0\bar{Q}_1\bar{Q}_2 + Q_0Q_1\bar{Q}_2 + Q_0\bar{Q}_1Q_2 + \bar{Q}_0Q_1Q_2$$

$$= \bar{Q}_2\,(\underbrace{Q_0Q_1 + \bar{Q}_0\bar{Q}_1}_{\substack{\downarrow \\ \cancel{\text{xor}}\ \text{xnor}}}) \to \bar{B} \quad + Q_2\,(\underbrace{Q_0\bar{Q}_1 + \bar{Q}_0Q_1}_{\substack{\downarrow \\ \text{xor}}}) \to A$$

$$= Q_2 \text{ xnor } (Q_0 \text{ xor } Q_1)$$

D1 :

Q_2 \ Q_1Q_0				
	1	1		
	1			1

$$= \cancel{Q_0 Q_1 Q}\ \ Q_0\bar{Q}_2 + \bar{Q}_2\bar{Q}_1$$

After this, just had to give the respective calculated inputs for D-values in the flip flops to get the next states of Y.

BEHVIOURAL:

In behavioural style of modelling, we straight forwardly had to link the next states to the previous ones and the rest of the cases can be linked to the reset pin.

Design document and VHDL code if relevant:

You should give a brief description of whatever designs you have constructed and a sketch (architecture of main logic) of the code you have written as part of the experiment.

STRUCTURAL:

This is the block of code I wrote for the flip flop inputs based on the Q-values:

```
Qnot(2) <= not Q(2);
Qnot(1) <= not Q(1);
Qnot(0) <= not Q(0);
P <= Q(1) xor Q(0);

D(2) <= Q(2) xnor P;

D(1) <= (Q(2) and Qnot(0)) or (Qnot(2) and Qnot(1));

D(0) <= (Qnot(2) and Q(0)) or (Q(2) and Q(1));
```

BEHAVIOURAL:

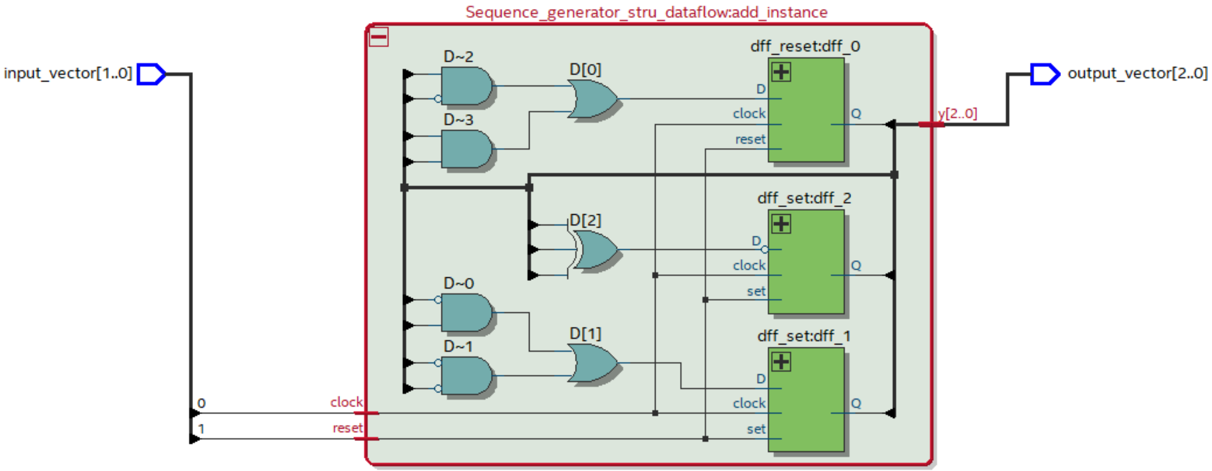
The majority part of the code that was used in this is attached followed, fulfilling the purpose of linking the expected new state with the previous state:

```
case state is
    when s_6 =>
    state<=s_7;
               when s_7 =>
    state<=s_1;
               when s_1 =>
    state<=s_3;
               when s_3 =>
    state<=s_5;
               when s_5 =>
    state<=s_4;
               when s_4 =>
    state<=s_2;
               when s_2 =>
    state<=s_0;
               when s_0 =>
    state<=s_6;

    when others=>
     state <= s_6;
    end case;
```

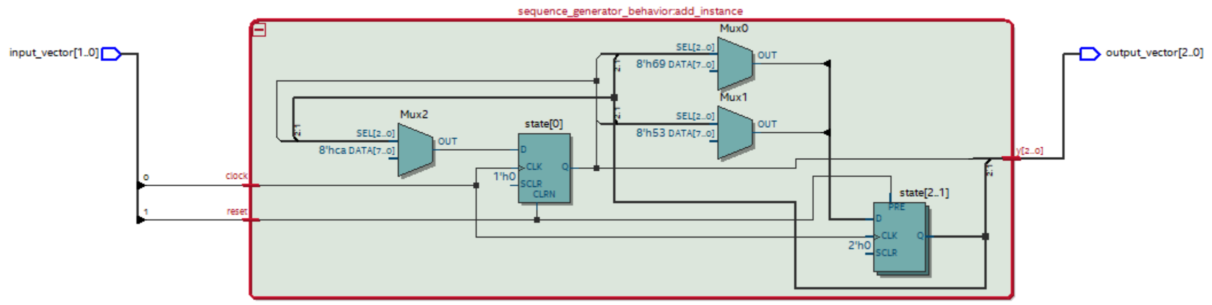
RTL View:

Attach screen-shot of the RTL view generated by Quartus.

STRUCTURAL:



BEHAVIOURAL:



DUT Input/Output Format:

Mention the format (LSB/MSB of input and output) and few test cases from trace-file.

reset,clock: in std_logic;
y:out std_logic_vector(2 downto 0)

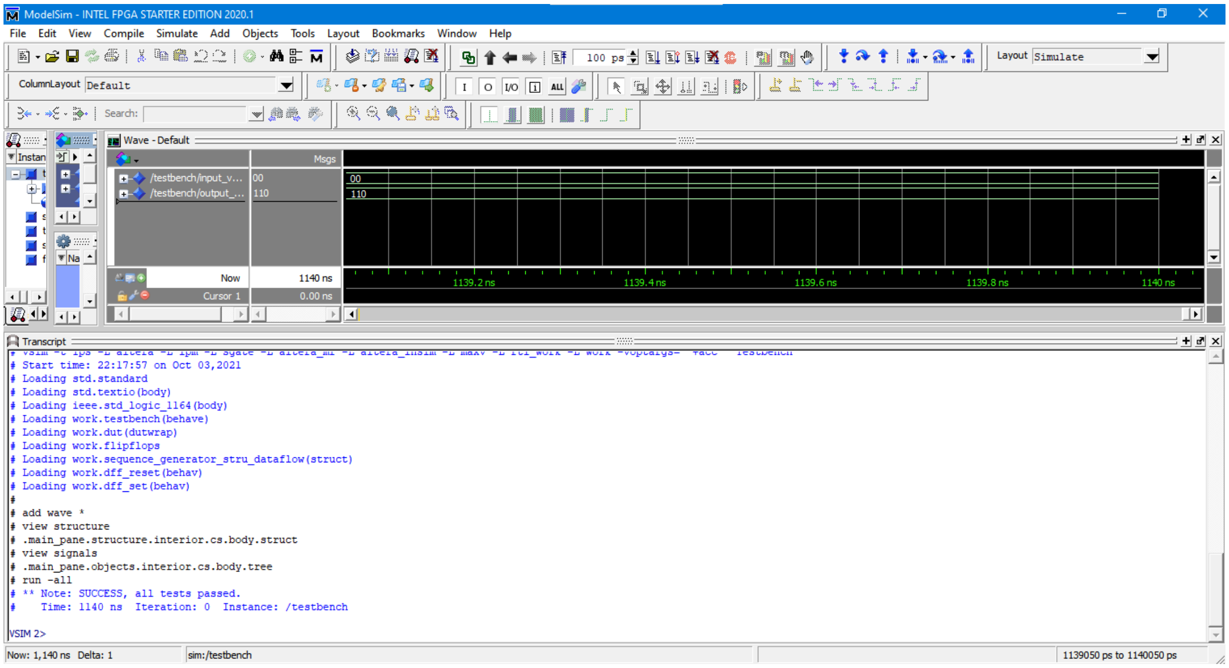
Reset,clock   y2,y1,y0

```
10 110
11 110
00 110
01 111
00 111
01 001
00 001
01 011
```

RTL Simulation:

Attach the clearly visible screen-shot of RTL simulation waveforms.

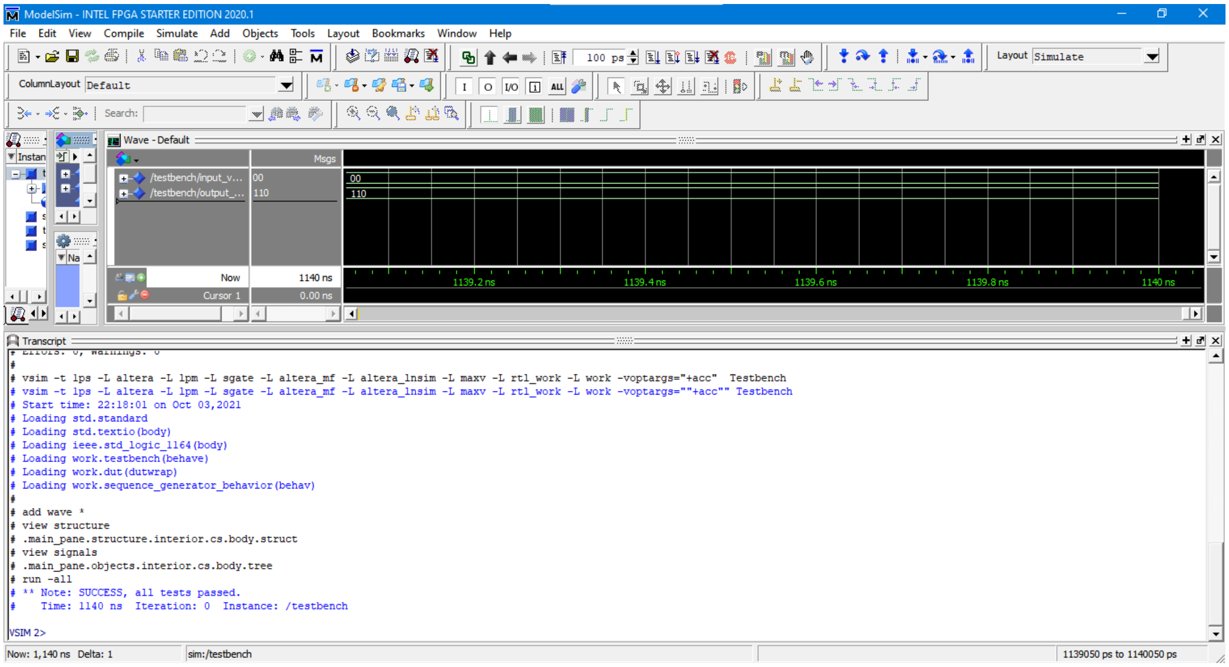
STRUCTURAL:



```
# vsim -t 1ps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L maxv -L rtl_work -L work -voptargs="+acc"  testbench
# Start time: 22:17:57 on Oct 03,2021
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behave)
# Loading work.dut(dutwrap)
# Loading work.flipflops
# Loading work.sequence_generator_stru_dataflow(struct)
# Loading work.dff_reset(behav)
# Loading work.dff_set(behav)
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
#    Time: 1140 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

```
Now: 1,140 ns  Delta: 1    sim:/testbench    1139050 ps to 1140050 ps
```

BEHAVIOURAL:



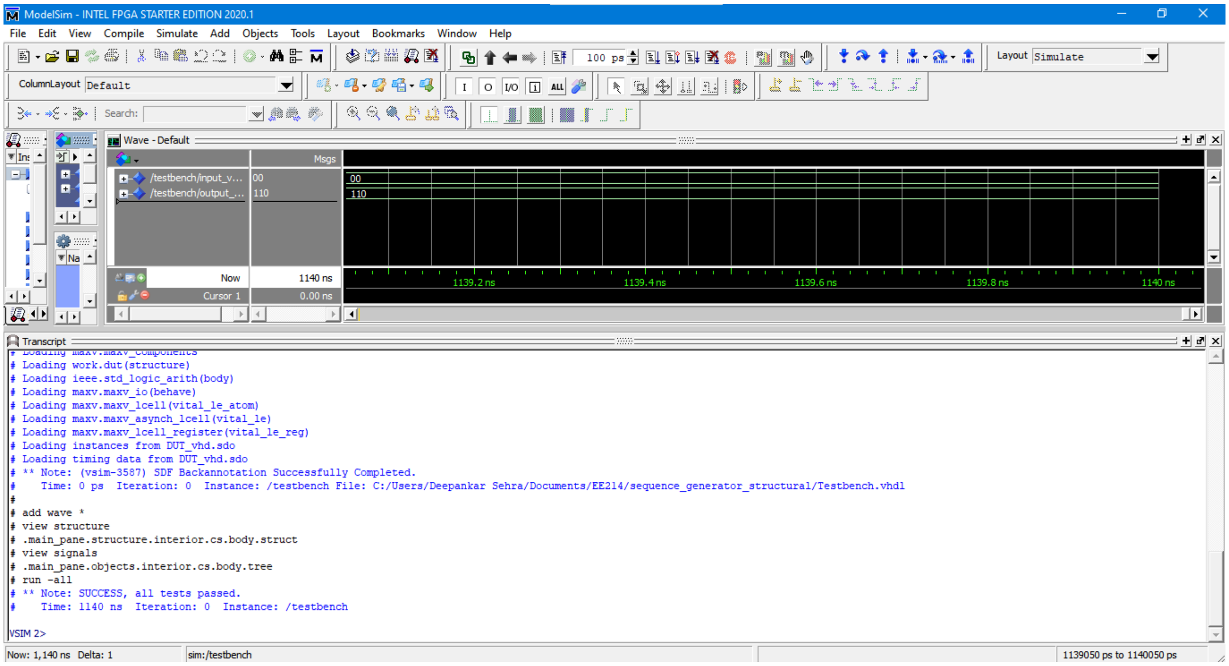
```
# Errors: 0, Warnings: 0
#
# vsim -t 1ps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L maxv -L rtl_work -L work -voptargs="+acc"  Testbench
# vsim -t 1ps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L maxv -L rtl_work -L work -voptargs=""+acc"" Testbench
# Start time: 22:18:01 on Oct 03,2021
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behave)
# Loading work.dut(dutwrap)
# Loading work.sequence_generator_behavior(behav)
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
#    Time: 1140 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

Now: 1,140 ns Delta: 1   sim:/testbench

Gate-level Simulation:

Attach the clearly visible screen-shot of Gate-level Simulation.

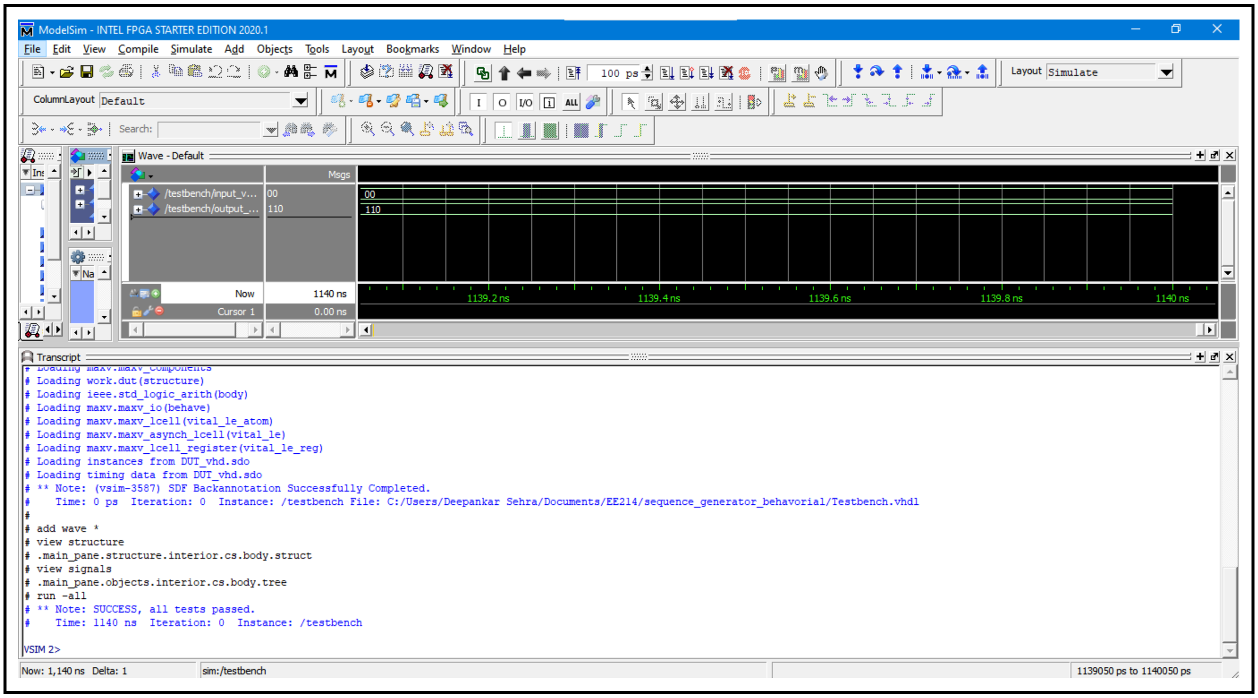
STRUCTURAL:



```
# Loading maxv.maxv_components
# Loading work.dut(structure)
# Loading ieee.std_logic_arith(body)
# Loading maxv.maxv_io(behave)
# Loading maxv.maxv_lcell(vital_le_atom)
# Loading maxv.maxv_asynch_lcell(vital_le)
# Loading maxv.maxv_lcell_register(vital_le_reg)
# Loading instances from DUT_vhd.sdo
# Loading timing data from DUT_vhd.sdo
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
#    Time: 0 ps  Iteration: 0  Instance: /testbench File: C:/Users/Deepankar Sehra/Documents/EE214/sequence_generator_structural/Testbench.vhdl
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
#    Time: 1140 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

Now: 1,140 ns Delta: 1   sim:/testbench

BEHAVIOURAL:

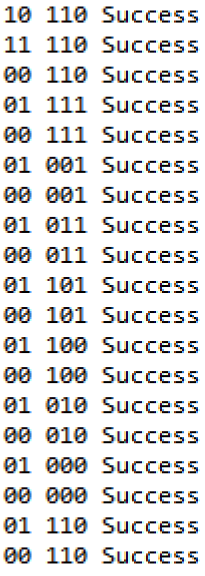


```
# Loading maxv.maxv_components
# Loading work.dut(structure)
# Loading ieee.std_logic_arith(body)
# Loading maxv.maxv_io(behave)
# Loading maxv.maxv_lcell(vital_le_atom)
# Loading maxv.maxv_asynch_lcell(vital_le)
# Loading maxv.maxv_lcell_register(vital_le_reg)
# Loading instances from DUT_vhd.sdo
# Loading timing data from DUT_vhd.sdo
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
#    Time: 0 ps  Iteration: 0  Instance: /testbench File: C:/Users/Deepankar Sehra/Documents/EE214/sequence_generator_behavorial/Testbench.vhdl
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
#    Time: 1140 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

Now: 1,140 ns Delta: 1   sim:/testbench   1139050 ps to 1140050 ps

Krypton board*:

Attaching the screenshot of scanchain outputs, as they would be same for both structural and behavioural:



```
10 110 Success
11 110 Success
00 110 Success
01 111 Success
00 111 Success
01 001 Success
00 001 Success
01 011 Success
00 011 Success
01 101 Success
00 101 Success
01 100 Success
00 100 Success
01 010 Success
00 010 Success
01 000 Success
00 000 Success
01 110 Success
00 110 Success
```

Observations*:

You must summarize your observations, either in words, using figures and/or tables.

References:

You may include the references if any.

* To be submitted after the tutorial on ”Using Krypton.