

Experiment 5: Sequence Generator

A submission Report

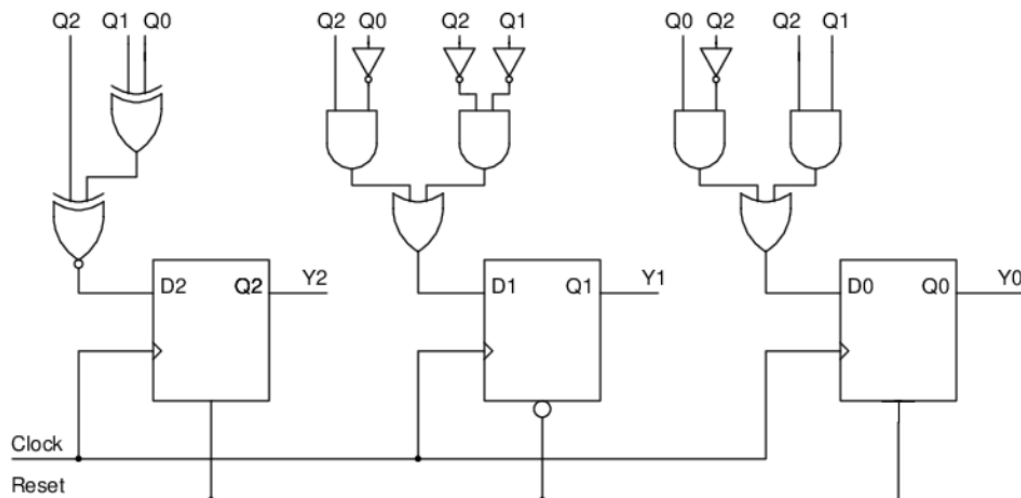
Hardik Panchal
Roll Number 200070054
EE-214, WEL, IIT Bombay
October 2, 2021

Overview of the experiment:

In this experiment we have to describe a sequence generator circuit in both structural and behavioral way. The desired sequence is given below in the design part. Also, the reset input was given so when reset in high we have to make some default value as output here it was 2.

I will be presenting here the design and code for this and also screenshots of successful RTL and Gate Level simulations.

Approach to the experiment:

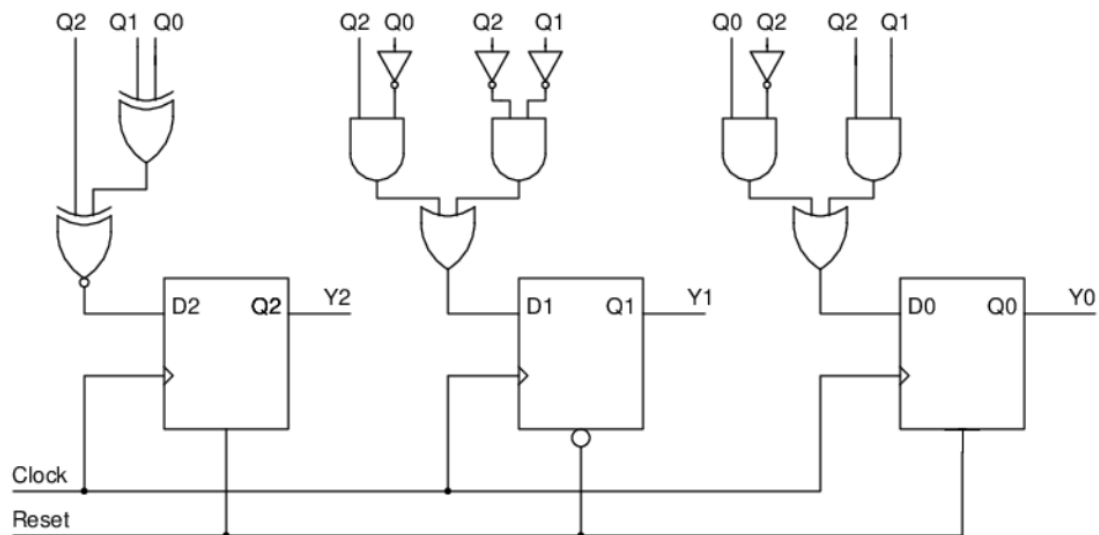
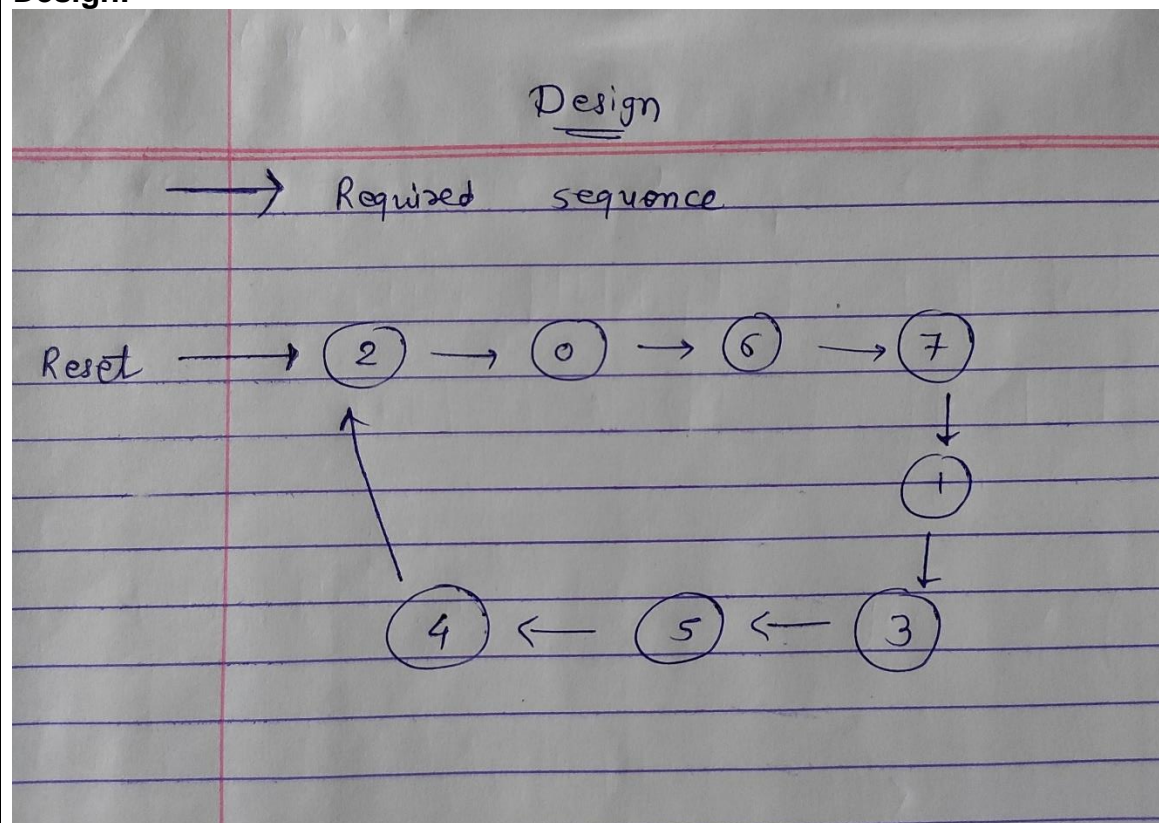


This circuit is my main guide for structural part. I have used three flipflops which were declared in the flipflops package. D2, D1 and D0 were determined using K-maps and their minimized expressions were used in the code, which can be seen in the above figure also. Reset was given according to that reset term. So, for example if by encountering reset our circuit has to output 2 then its binary representation is 010 in 3 bits. From this we can make the flipflop design and use them.

Behavioral part was very easy. It has just few case statements and using that we can code our needed functionality.

Design document and VHDL code if relevant:

Design:



Behavioral part:

Architecture of main logic:

architecture behav of sequence_behavior is

--state binary encoding

signal state:std_logic_vector(2 downto 0);

```
constant s_0:std_logic_vector(2 downto 0):="000";
constant s_1:std_logic_vector(2 downto 0):="001";
constant s_2:std_logic_vector(2 downto 0):="010";
constant s_3:std_logic_vector(2 downto 0):="011";
constant s_4:std_logic_vector(2 downto 0):="100";
constant s_5:std_logic_vector(2 downto 0):="101";
constant s_6:std_logic_vector(2 downto 0):="110";
constant s_7:std_logic_vector(2 downto 0):="111";
```

begin

-- process for next state and output logic

reg_process: process(clock,reset)

begin

if(reset='1')then

state<= s_2; -- write the reset state

elsif(clock'event and clock='1')then

case state is

--reset

when s_2=>

state<=s_0;

when s_0=>

state<=s_6;

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;

```

--DEFAULT CASE
when others=>
    state<= s_2;-- write the reset state
end case;

```

```

end if;
end process reg_process;
-- output logic concurrent statemet or one more process

```

```

y<=state;
end behav;

```

Structural part:

Architecture of main logic:

architecture struct of sequence_generator_structural is

```

signal D2,D1,D0 :std_logic;
signal Q:std_logic_vector(2 downto 0);
begin
    D2<= (Q(2) xnor (Q(1) xor Q(0)));
    D1<= ((Q(2) and (not(Q(0)))) or ((not(Q(2))) and (not(Q(1)))));
    D0<= ((Q(0) and (not(Q(2)))) or (Q(2) and Q(1)));

```

```

y(2)<= Q(2);
y(1)<= Q(1);
y(0)<= Q(0);

```

```

--Q0
dff_0 : dff0 port map(D0,clock,reset,Q(0));

```

```

--Q1
dff_1 : dff1 port map(D1,clock,reset,Q(1));

```

```

--Q2
dff_2 : dff2 port map(D2,clock,reset,Q(2));

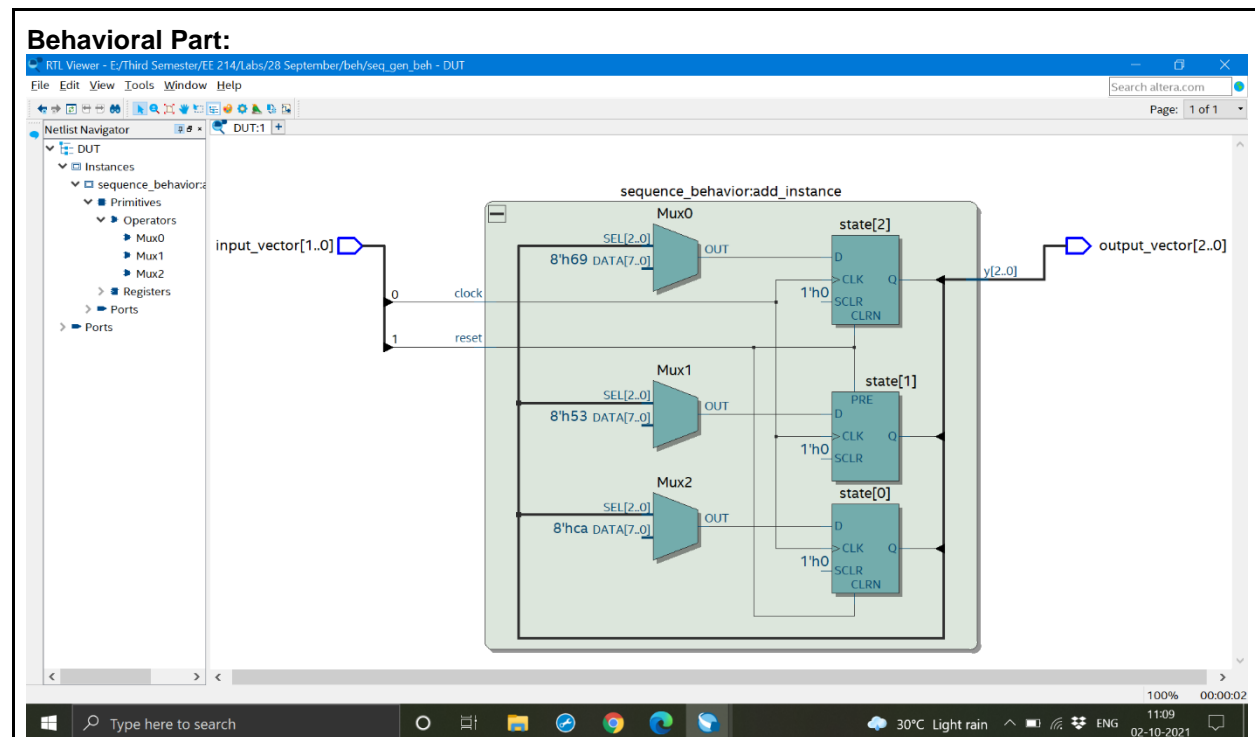
```

```

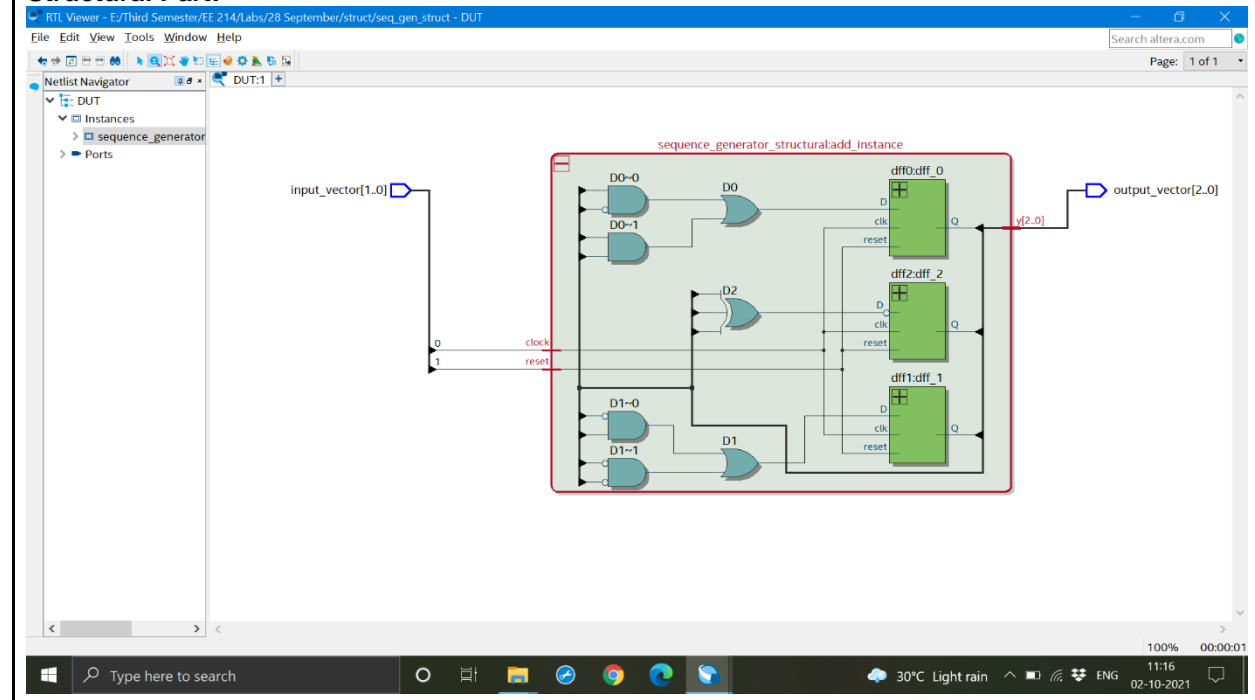
end struct;

```

RTL View:



Structural Part:



DUT Input/Output Format:

Input: reset clock **LSB** = clock **MSB** = reset

Output: y2 y1 y0 **LSB** = y0, **MSB** = y1

Some Test Cases from TRACEFILE.txt

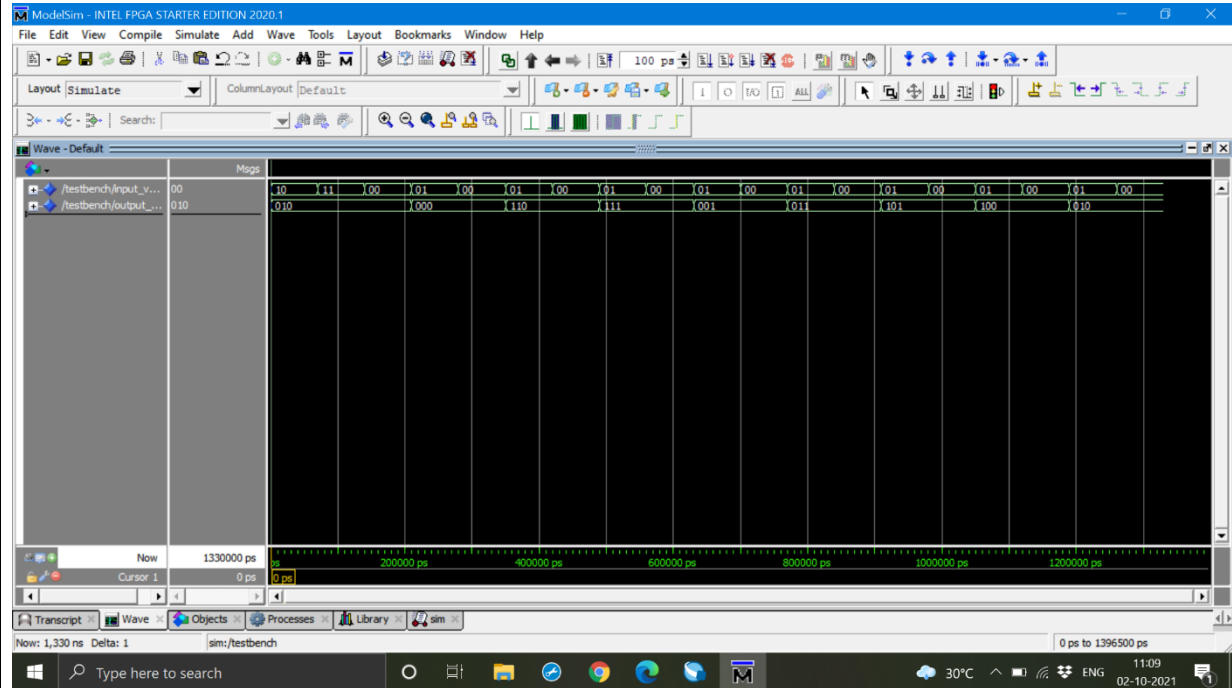
Format: reset clock y2 y1 y0

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

RTL Simulation:

Behavioral Part:

Waveform



Transcript

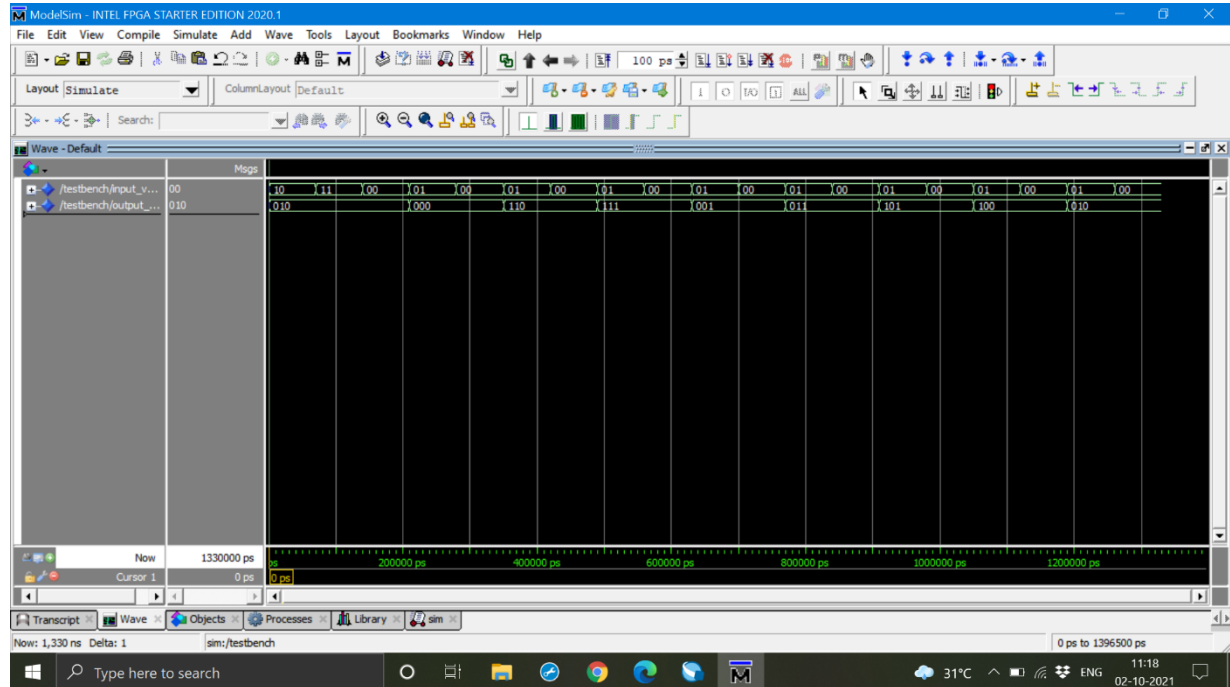
```
ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help

# Model Technology ModelSim - Intel FPGA Edition vcom 2020.1 Compiler 2020.02 Feb 28 2020
# Start time: 11:09:24 on Oct 02, 2021
# vcom -reportprogress 300 -93 -work work E:/Third Semester/EE 214/Labs/28 September/beh/Testbench.vhdl
# -- Loading package STANDARD
# -- Loading package TEXTIO
# -- Loading package std_logic_1164
# -- Compiling entity Testbench
# -- Compiling architecture Behave of Testbench
# End time: 11:09:24 on Oct 02, 2021, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
#
# vsim -t lps -L altera -l lpm -L sgate -L altera_mf -L altera_insim -L maxv -L rtl_work -L work -voptargs="+acc" Testbench
# vsim -t lps -L altera -l lpm -L sgate -L altera_mf -L altera_insim -L maxv -L rtl_work -L work -voptargs="+acc" Testbench
# Start time: 11:09:24 on Oct 02, 2021
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behav)
# Loading work.dut(dutwrap)
# Loading work.sequence_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: 1330 ns Iteration: 0 Instance: /testbench

VSIM 2>
```

Structural Part:

Waveform



Transcript

ModelSim - INTEL FPGA STARTER EDITION 2020.1

File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help

Layout Simulate ColumnLayout Default

Transcript

```
-- Loading package TEXTIO
-- Loading package std_logic_1164
-- Compiling entity Testbench
-- Compiling architecture Behave of Testbench
End time: 11:16:52 on Oct 02, 2021, Elapsed time: 0:00:00
Errors: 0, Warnings: 0
#
# vsim -t lps -L altera -L lpm -L sgate -L altera_mf -L altera_insim -L maxv -L rtl_work -L work -voptargs="+acc" Testbench
# vsim -t lps -L altera -L lpm -L sgate -L altera_mf -L altera_insim -L maxv -L rtl_work -L work -voptargs="+acc" Testbench
# Start time: 11:16:52 on Oct 02, 2021
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behav)
# Loading work.dut(dutwrap)
# Loading work.flipflops
# Loading work.sequence_generator_structural(struct)
# Loading work.dff0(behav)
# Loading work.dff1(behav)
# Loading work.dff2(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: 1330 ns Iteration: 0 Instance: /testbench

VSIM 2>
```

Now: 1330 ns Delta: 1

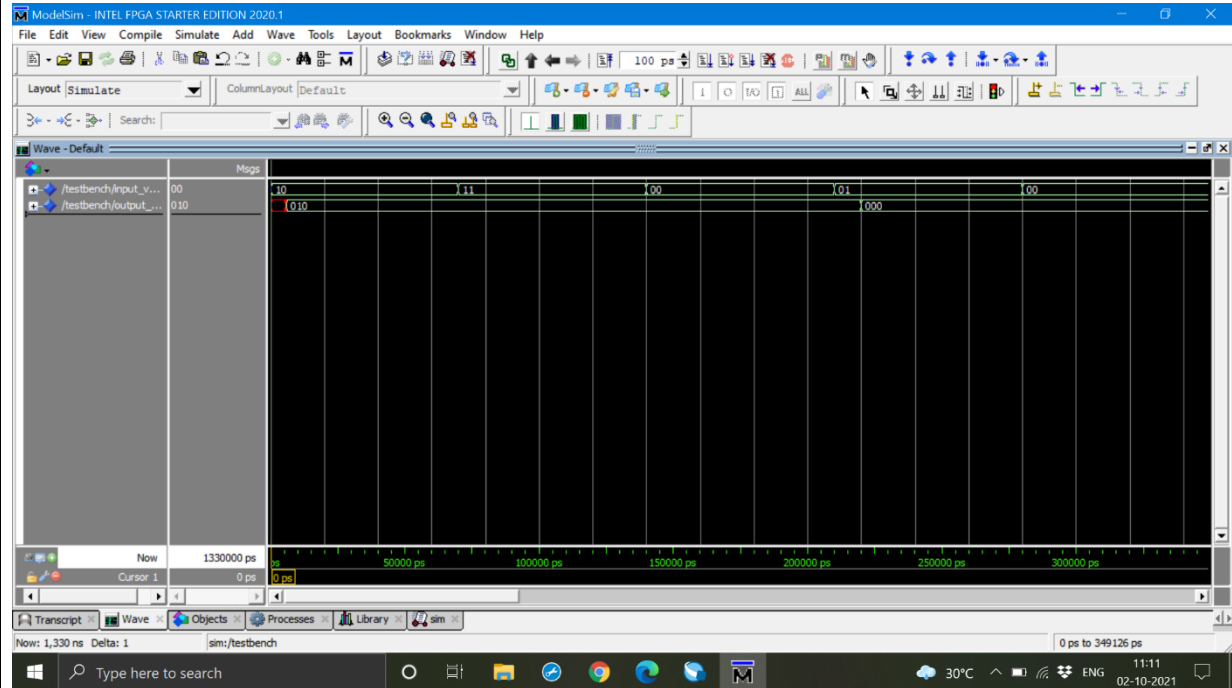
sim:/testbench

11:18 02-10-2021

Gate-level Simulation:

Behavioral Part:

Waveform



Transcript

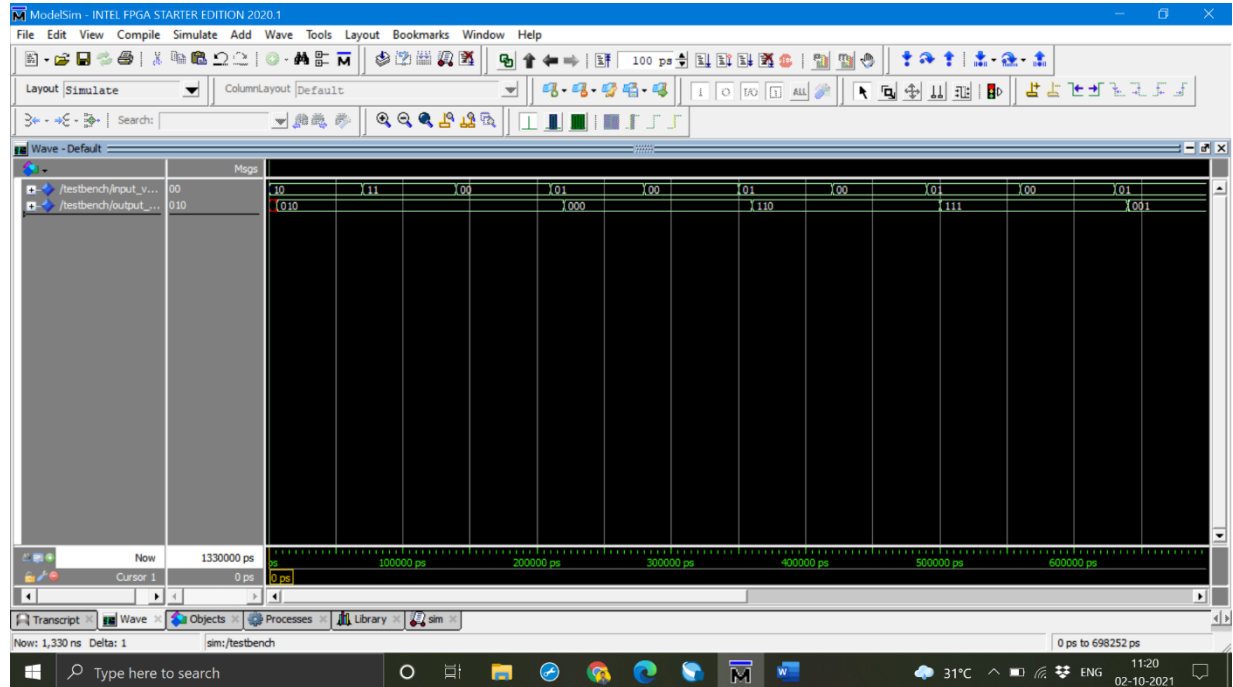
```
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behavior)
# SDF 2020.1 Compiler 2020.02 Feb 28 2020
#
# Loading ieee.vital_timing(body)
# Loading ieee.vital_primitives(body)
# Loading maxv.maxv_atom_pack(body)
# Loading maxv.maxv_components
# Loading work.dut(structure)
# Loading ieee.std_logic_arith(body)
# Loading maxv.maxv_io(behavior)
# 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: (vaim-3587) SDF Backannotation Successfully Completed.
#   Time: 0 ps Iteration: 0 Instance: /testbench File: E:/Third Semester/EE 214/Labs/28 September/beh/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: 1330 ns Iteration: 0 Instance: /testbench

VSIM 2>
```

The transcript window shows the simulation setup and results. It includes the loading of standard libraries, the SDF compiler, and the successful completion of the simulation at 1,330 ns. The status bar at the bottom indicates the current time is 1,330 ns and the delta is 1 ps.

Structural Part:

Waveform



Transcript

ModelSim - INTEL FPGA STARTER EDITION 2020.1

File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help

Layout Simulate ColumnLayout Default

Transcript

```
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behavior)
# SDF 2020.1 Compiler 2020.02 Feb 28 2020
#
# Loading ieee.vital_timing(body)
# Loading ieee.vital_primitives(body)
# Loading maxv.maxv_atom_pack(body)
# Loading maxv.maxv_components
# Loading work.dut(structure)
# Loading ieee.std_logic_arith(body)
# Loading maxv.maxv_io(behavior)
# 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: E:/Third Semester/EE 214/Labs/28 September/struct/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: 1330 ns Iteration: 0 Instance: /testbench
```

VSIM 2>

Now: 1,330 ns Delta: 1 sim:/testbench

Krypton board:

We have used scanchain for this experiment. So **out.txt** has output which I got using scanchain.

Behavioral part:

```
E:\Third Semester\EE 214\UrJTAG\urjtag_windows\jtag.exe

UrJTAG 0.10 #1502
Copyright (C) 2002, 2003 ETC s.r.o.
Copyright (C) 2007, 2008, 2009 Kolja Waschk and the respective authors

UrJTAG is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
There is absolutely no warranty for UrJTAG.

WARNING: UrJTAG may damage your hardware!
Type "quit" to exit, "help" for help.

jtag> cable ft232
Connected to libftd2xx driver.
jtag> detect
IR length: 10
Chain length: 1
Device Id: 00000010000010100011000011011101 (0x0000000020A30DD)
  Manufacturer: Altera
  Part(0):      5M1270
  Stepping:     1
  Filename:     e:\third semester\ee 214\urjtag\urjtag_windows\data\altera\5m1270\5m1270
jtag> svf beh.svf progress
Warning svf: unimplemented mode 'ABSENT' for TRST
Parsing 40830/40835 ( 99%)
Scanned device output matched expected TDO values.
jtag> _
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

E:\Third Semester\EE 214\Labs\28 September\beh>scan_vjtag.exe TRACEFILE.txt out.txt
{'type': 6, 'id': 67330064, 'description': 'b'Dual RS232-HS A', 'serial': 'b'A'}

E:\Third Semester\EE 214\Labs\28 September\beh>_
```

Structural part:

```
E:\Third Semester\EE 214\UrJTAG\urjtag_windows\jtag.exe
UrJTAG 0.10 #1502
Copyright (C) 2002, 2003 ETC s.r.o.
Copyright (C) 2007, 2008, 2009 Kolja Waschk and the respective authors

UrJTAG is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
There is absolutely no warranty for UrJTAG.

WARNING: UrJTAG may damage your hardware!
Type "quit" to exit, "help" for help.

jtag> cable ft2232
Connected to libFtd2xx driver.
jtag> detect
IR length: 10
Chain length: 1
Device Id: 00000010000010100011000011011101 (0x0000000020A30DD)
  Manufacturer: Altera
  Part(0):      5M1270
  Stepping:    1
  Filename:     e:\third semester\ee 214\urjtag\urjtag_windows\data\altera\5m1270\5m1270
jtag> svf struct.svf progress
Warning svf: unimplemented mode 'ABSENT' for TRST
Parsing 40830/40835 ( 99%)
Scanned device output matched expected TDO values.
jtag>
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

E:\Third Semester\EE 214\Labs\28 September\struct>scan_vjtag.exe TRACEFILE.txt out.txt
{'type': 6, 'id': 67330064, 'description': b'Dual RS232-HS A', 'serial': b'A'}

E:\Third Semester\EE 214\Labs\28 September\struct>
```

Some outputs from out.txt(For both behavioral and Structural part):

```
10 010 Success
11 010 Success
00 010 Success
01 000 Success
00 000 Success
01 110 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
```

Observations:

The main observation and learning outcome from this experiment consisting of sequential circuit was that describing this type of circuit is very easy in behavioral style while structural style demands more effort. Using behavioral style of coding we can easily describe the sequential circuits. In structural style we had to use the sequential element flip flop which was again described in behavioral style.

References:

My main reference was our course webpage it contained many useful things such as a sample code and many other specifications. It also had one handout for this experiment which was very helpful. This I used as my reference.