Experiment 5: Sequential Circuit 1

Madan Y N Roll Number 200070040

EE-214, WEL, IIT Bombay

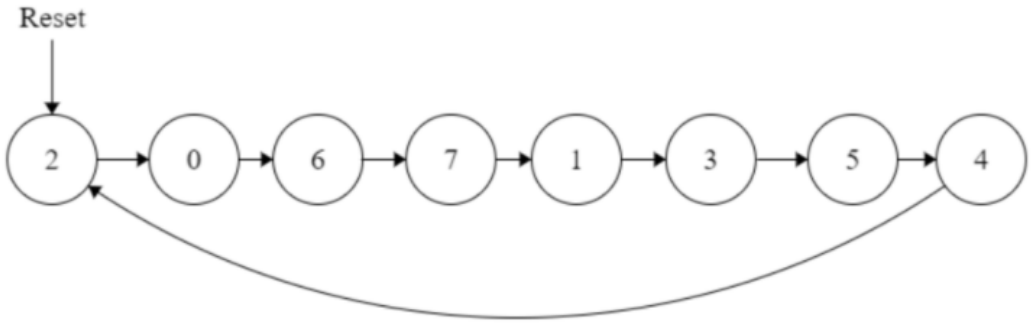
September 28, 2021

Overview of the experiment:

The purpose of the experiment is to build a Sequence Generator using structural and behavioral and dataflow modelling in VHDL and to test the correctness of Sequence Generator built using scanchain.
In this report I present in a brief how the experiment was conducted and would also present the results of the experiment.

Approach to the experiment:

The Sequence Generator is expected to generate a certain sequence of numbers with outputs changing at the rising edge of the clock. The output of the generator at a time is decided by what was the output in the previous edge of the clock. The sequence to be generated is as below



Reset is one check that we employ in the sequence generator. When reset is ‘1’ regardless of the present state of output, the next output of the sequence generator would be ‘2’.

In the structural description we made use of D-flipflops. Three flipflops were used, one each for an output.
State Table of the Sequence Generator

Present State(Q2Q1Q0)	Next state(nQ2nQ1nQ0)	D2D1D0
000(0)	110(6)	110
001(1)	011(3)	011
010(2)	000(0)	000
011(3)	101(5)	101
100(4)	010(2)	010
101(5)	100(4)	100
110(6)	111(7)	111
111(7)	001(1)	001

The logic expression for the inputs of D-flipflops in terms of their outputs are,

$D0 \ = \ (((not(Q(2)) \ and \ Q(0))) \ or \ (Q(2) \ and \ Q(1)))$
$D1 \ = \ ((Q(2) \ and \ (\ not \ (Q(0)))) \ or \ ((not \ (Q(2))) \ and \ (not \ (Q(1)))))$
$D2 \ = \ (Q(2) \ xnor \ (Q(1) \ xor \ Q(0)))$
$D2, \ D1, D0$ being inputs to 3 D-flipflops and $Q(2), Q(1), Q(0)$ being respective outputs.

Design document and VHDL code

For the structural part of the experiment an entity was created for the 'Sequence_Generator_Structural' with suitable inputs and outputs.D-flipflops were created in a separate VHDL file and loaded in as a package 'flipflops'.

The architecture of the structural part is as below

```
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<= (((not(Q(2)) and Q(0))) or (Q(2) and Q(1)));
  y(2)<= Q(2);
```

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

dff_0  : dff0 port map(D=>D0,clk=>clock,res=>reset,Q=>Q(0));

dff_1  : dff1 port map(D=>D1,clk=>clock,res=>reset,Q=>Q(1));

dff_2  : dff2 port map(D=>D2,clk=>clock,res=>reset,Q=>Q(2));

end struct;
```

For the behavioural an entity was created for ‘Sequence_Generator_Behavioural’ with suitable inputs and outputs. Behavioural description was done for the sequence generator.

The architecture of the Sequence_Generator_Behavioural is as follows

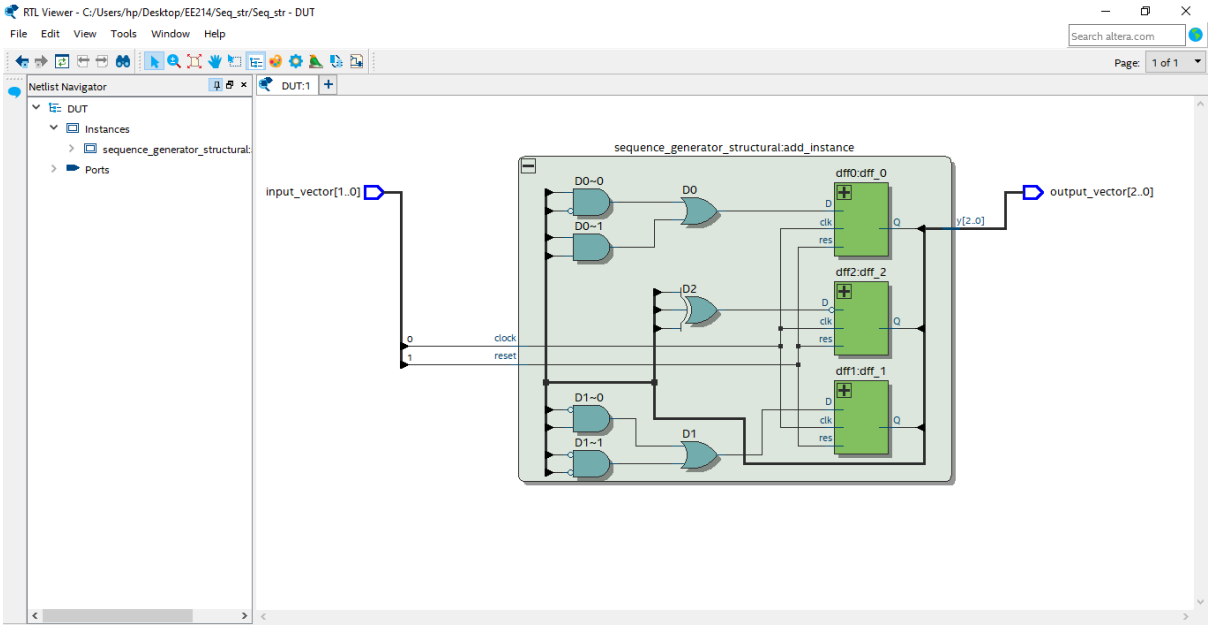
```
architecture behav  of sequence_behavior is

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

reg_process: process(clock,reset)
begin

if(reset='1')then
state<= s_2;
elsif(clock'event and clock='1')then
case state is

     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;

     when others=>
    state<=s_2;
     end case;
end if;
end process reg_process;

y<=state;
end behav;
```

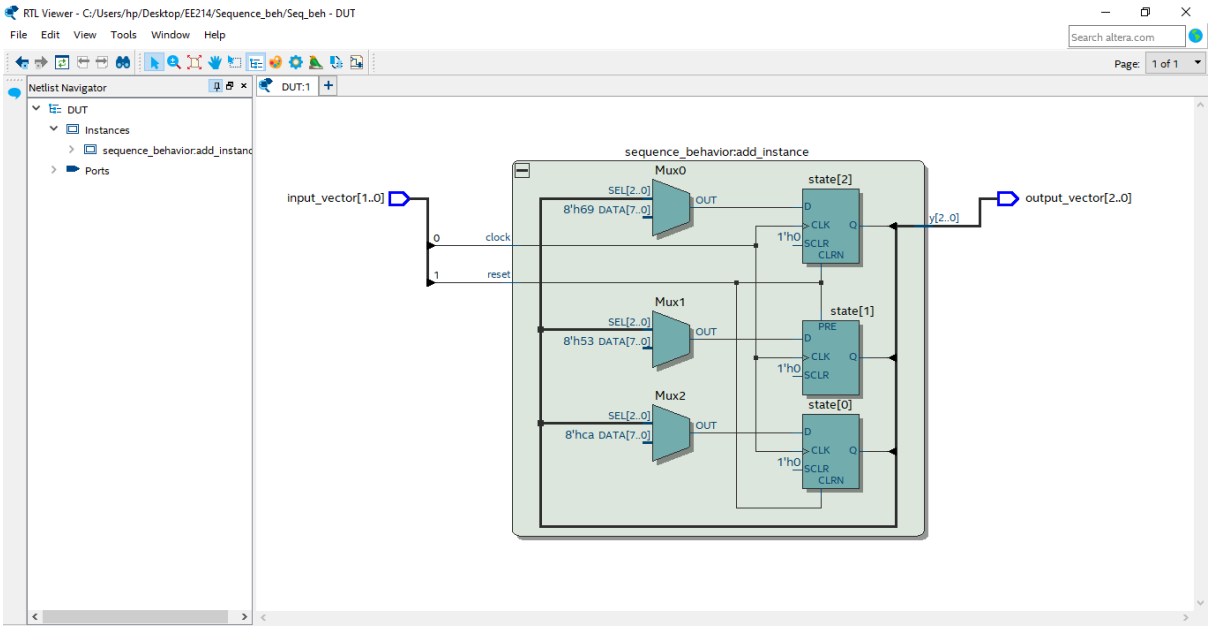
In order to test the correctness of the design on hardware we made use of scanchain. This also allowed us to check if our design could actually be implemented on hardware. We generated a svf file which was loaded onto the krypton board,which was later used to run the scanchain test.

RTL View:

RTL view of the Structural description



RTL view of the Behavioural description



DUT Input/Output Format:

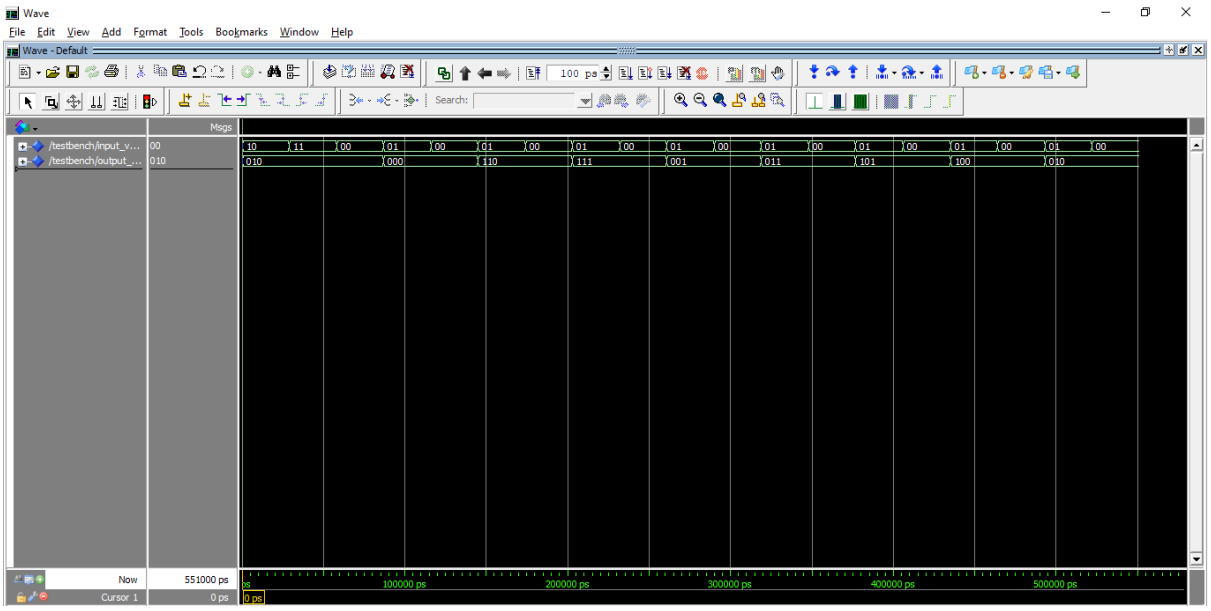
The input vector to both structural and behavioural Sequence Generator is of length 2 i.e a single bit clock and another single bit reset. Reset is the msb of the input and clock is the lsb.
Output of the Sequence Generator is a 3 bit number $Y2Y1Y0$ as it is evident $Y2$ is the msb and $Y0$ is the lsb.

Some test cases from the tracefile are

Input	Output
01	111
00	111
01	001
00	001
01	011
00	011

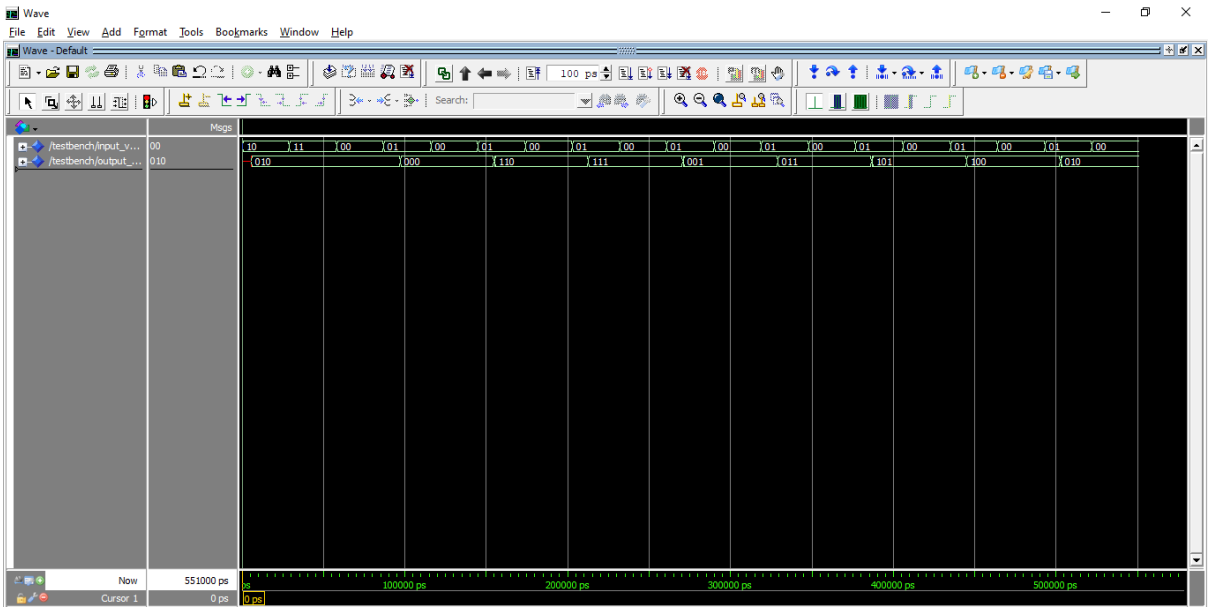
RTL Simulation:

RTL simulation of Sequence Generator



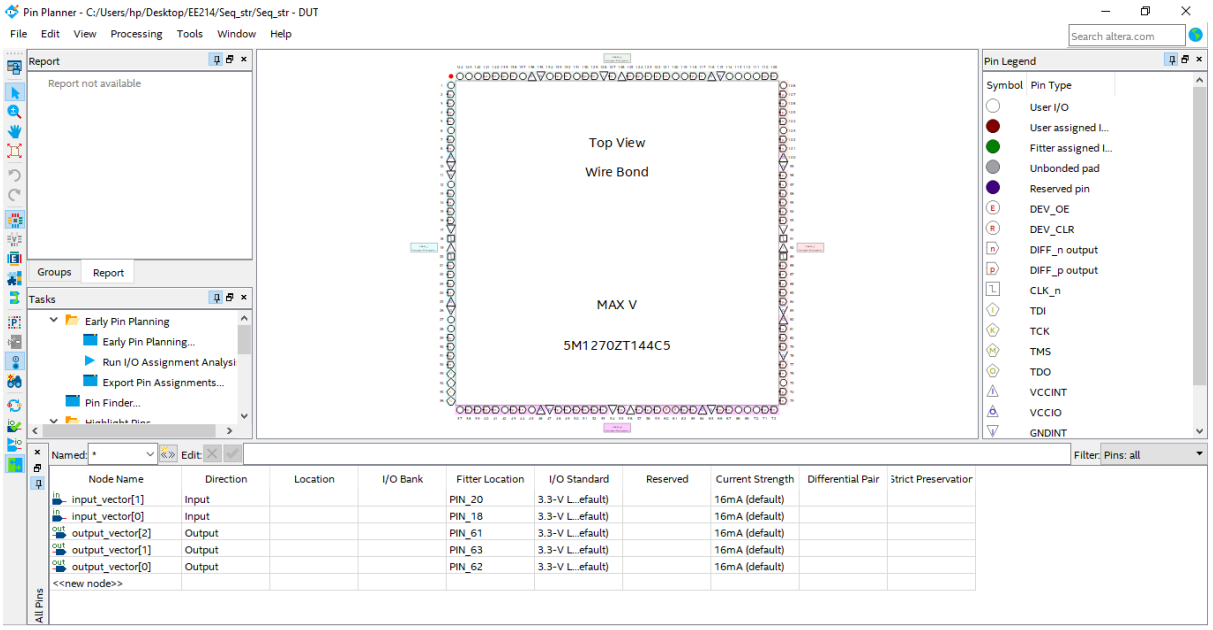
Gate-level Simulation:

Gate level simulation of Sequence Generator



Krypton Board

The pin planning of the krypton board is as below



Observation:

Upon testing using scanchain we got an output file *out.txt* which contained the result of the experiment. In it we get to see if the design made by us is correct and if the design could be implemented on hardware. All the test cases passed in scanchain which assures the correctness of the design.

In this lab we did both structural and behavioural descriptions of the same problem. I observed using both description methods together we can make the code nice and short rather than sticking to either one of them.