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:

|  |
| --- |
| **Behavioral Part:**    **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**    **Structural Part:**  **Waveform**    **Transcript** |

## Gate-level Simulation:

|  |
| --- |
| **Behavioral Part:**  **Waveform**    **Transcript**    **Structural Part:**  **Waveform**    **Transcript** |

## Krypton board:

|  |
| --- |
| We have used scanchain for this experiment. So **out.txt** has output which I got using scanchain.  **Behavioral part:**      **Structural part:**      **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. |