Experiment 5: Sequence Generator

Nishant Thakre Roll Number 200070051

EE-214, WEL, IIT Bombay

September 19, 2021

## Overview of the experiment:

|  |
| --- |
| * The purpose of the experiment was to use Behavioral and Structural modelling to describe a sequence generator * We used D-Flip flops for structural modelling and Case-When statements for Behavioral modelling. * In this report, I will describe the working of the circuit and show the subsequent simulations obtained from the said experiment. |

## Approach to the experiment:

|  |
| --- |
| . I used Case**-**Whenstatements and D-flip flopsrespectively to generate the behavioral and data flow modelling. I used the truth tables, expressions and skeleton codes provided in the Lab- Handout. When the reset input was set to ‘1’, the sequence was reset to start from ‘2’ again. |

## Design document and VHDL code if relevant:

|  |
| --- |
| VHDL code for behavioral modelling  library ieee;  use ieee.std\_logic\_1164.all;  entity sequence\_behavior is  port (reset,clock: in std\_logic; y:out std\_logic\_vector(2 downto 0));  end entity sequence\_behavior;  architecture behav of sequence\_behavior is  --state binary encoding  signal state:std\_logic\_vector(2 downto 0);  constant s\_2:std\_logic\_vector(2 downto 0):="010";  constant s\_0:std\_logic\_vector(2 downto 0):="000";  constant s\_1:std\_logic\_vector(2 downto 0):="001";  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";  --write the remaining constant declarations  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\_1=>  state<=s\_3;  when s\_3=>  state<=s\_5;  when s\_4=>  state<=s\_2;  when s\_5=>  state<=s\_4;  when s\_6=>  state<=s\_7;  when s\_7=>  state<=s\_1;  -- write the remaining choices  --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;  VHDL code for structural modelling  library ieee;  use ieee.std\_logic\_1164.all;  library work;  use work.flipflops.all;  entity sequence\_generator\_structural is  port (reset,clock: in std\_logic; y:out std\_logic\_vector(2 downto 0));  end entity sequence\_generator\_structural;  architecture struct of sequence\_generator\_structural is  signal D2,D1,D0,resetbar :std\_logic;  signal Q:std\_logic\_vector(2 downto 0);  begin  -- write the equations here  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));  -- Do the port mapping --asynchronous reset  --Q0  dff\_1 : dff1 port map(D=>D0,clk=>clock,reset=>reset,Q=>Q(0));  --Q1  dff\_2 : dff2 port map(D=>D1,clk=>clock,reset=>reset,Q=>Q(1));  --Q2  dff\_3 : dff3 port map(D=>D2,clk=>clock,reset=>reset,Q=>Q(2));  y(2)<=Q(2);  y(1)<= Q(1);  y(0)<=Q(0);  end struct; |

## RTL View:

|  |
| --- |
| For behavioral model    For structural |

## DUT Input/Output Format:

|  |
| --- |
| Input format=< reset clock >  Output format=< Y2 Y1 Y0>  Input MSB=reset  Input LSB=clock  Output MSB=Y2  Output LSB=Y0  Some testcases are:  10 010  11 010  00 010  01 000 |

## RTL Simulation:

|  |
| --- |
| For behavioral modelling    For structural modelling |

## Gate-level Simulation:

|  |
| --- |
| For behavioral modelling    For structural modelling |

## Observations\*:

|  |
| --- |
| We obseved that all testcases have been passed and all test cases in scan chain also passed and the output file obtained from scan-chain also had all tests as SUCCESS |

## 