|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Hardware Description Language Lab (2UXL401)** | **Semester:** | **IV** |
| **Date of Performance:** |  | **Batch No:** |  |
| **Faculty Name:** |  | **Roll No:** |  |
| **Faculty Sign & Date:** |  | **Grade/Marks:** |  |

**Experiment No: 6**

**Title:** FSM implementation: Sequence detection

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| Write a VHDL code for implementing a Moore type, non-overlapping sequence detector which detects “10101” sequence and gives output high. Write a test bench to verify your results.  To study FSM implementation in VHDL and to understand use of test bench for simulation. |

|  |
| --- |
| **COs to be achieved:** |
| **CO 2**: Test a VHDL code and verify the circuit model.  **CO 3**: Synthesize and Implement the designed circuits on CPLD/ FPGA. |

|  |
| --- |
| **Work to be done** |
| Upload VHDL codes for sequence generator FSM. Also upload test bench and simulation for the same. |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:** |
| Upload Answer of following question before coming to next laboratory.  **Q1.** What changes will you make in VHDL code if the same sequence generator is mealy overlapping type?  **Q2.** **Examine** **the** **following** **VHDL** **code** **and** **complete** **the** **following** entity Problem  Port (X, CLK : in bit; Z1, Z2 : out bit);  end Problem;  architecture Table of Problem is  signal State, Nextstate: integer range 0 to 3 :=0; begin  process(State, X) begin  case State is when 0 =>  if X = „0‟ then Z1 <= „1‟; Z2 <= „0‟; Nextstate < = 0; else Z1 < =‟0‟; Z2 <=‟0‟;Nextstate < = 1; end if;  when 1 =>  if X = „0‟ then Z1 <=„1‟; Z2 <= „1‟; Nextstate < = 1; else Z1 < =„0‟; Z2 <=‟1‟; Nextstate < = 2; end if;  when 2 =>  if X = „0‟ then Z1 <=„0‟; Z2 <= „1‟; Nextstate < = 2; else Z1 < =„0‟; Z2 <=‟1‟; Nextstate < = 3; end if;  when 3 =>  if X = „0‟ then Z1 <=„0‟; Z2 <= „0‟; Nextstate < = 0; else Z1 < =„1‟; Z2 <=‟0‟; Nextstate < = 1; end if; end case;  end process;  process(CLK) begin  if CLK‟event and CLK =‟1‟ then State <= Nextstate;  end if;  end process; end Table;  (a) **Draw a block diagram of the circuit implemented by this code**  **(b) Write the state table that is implemented by this code**  **(c) Write the type of state machine** |

|  |
| --- |
| **Conclusion:** |

|  |
| --- |
| **Signature of faculty in-charge with Date:** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Hardware Description Language Lab (2UXL401)** | **Semester:** | **IV** |
| **Date of Performance:** |  | **Batch No:** |  |
| **Faculty Name:** |  | **Roll No:** |  |
| **Faculty Sign & Date:** |  | **Grade/Marks:** |  |

**Experiment No: 7**

**Title:** FSM Implementation : Word Problem

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| Word problem for development of state diagram and design using VHDL.  To study FSM implementation in VHDL and to understand use of test bench for simulation. |

|  |
| --- |
| **COs to be achieved:** |
| **CO 1**: Use basic Concurrent and Sequential statements in VHDL and write codes for simple applications  **CO 2**: Test a VHDL code and verify the circuit model. |

|  |
| --- |
| **Work to be done** |
| Draw FSM for given problem. Upload VHDL codes for FSM developed from word problem. Also upload test bench and simulation for the same. |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:** |
| Upload Answer of following question before coming to next laboratory. Q1. Write a VHDL code for FSM implementation of a following state diagram.Write a test bench for the same. http://yue-guo.com/wp-content/uploads/2018/11/1001_moore-134x300.png Q2. Analyse the code and draw the state diagram library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL; use IEEE.STD\_LOGIC\_ARITH.ALL;  use IEEE.STD\_LOGIC\_UNSIGNED.ALL;  entity fsm is  Port ( ip1 : in std\_logic; ip2 : in std\_logic; ip3 : in std\_logic; op1 : out std\_logic; op2 : out std\_logic; op3 : out std\_logic; reset : in std\_logic; clk : in std\_logic);  end fsm;  architecture fsm\_a of fsm is  type state\_t is (s\_idle,s1,s2,s3);  signal present\_state, next\_state : state\_t;  process(present\_state,ip1,ip2,ip3) begin  case present\_state is when s\_idle =>  if (ip1 ='1') then  op1<='1'; op2<='0'; op3<='0';  next\_state<= s1; elsif (ip2='1') then  op1<='0'; op2<='1'; op3<='0';  next\_state<= s2; elsif (ip3='1') then  op1<='0'; op2<='0'; op3<='1';  next\_state<= s1;  else  op1<='0'; op2<='0'; op3<='0';  next\_state<= s\_idle;  end if; when s1 =>  if (ip1 ='1') then  op1<='1'; op2<='0'; op3<='0';  next\_state<= s1; else  op1<='0'; op2<='0'; op3<='0';  next\_state<= s\_idle; end if;  when s2 =>  if (ip2 ='1') then  op1<=‘0'; op2<=‘1';op3<='0';  next\_state<= s2; else  op1<='0'; op2<='0'; op3<='0';  next\_state<= s\_idle; end if;  when s3 =>  if (ip3 ='1') then  op1<='0'; op2<='0'; op3<='1';  next\_state<= s3;  else  op1<='0'; op2<='0'; op3<='0';  next\_state<= s\_idle;  end if; end case;  end process; process(clk,reset) begin  if reset = '1' then present\_state <= s\_idle;  elsif clk'event and clk='1' then present\_state<= next\_state;  end if; end process; end fsm\_a; |

|  |
| --- |
| **Conclusion:** |

|  |
| --- |
| **Signature of faculty in-charge with Date:** |