|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Hardware Description Language Lab (2UXL401)** | **Semester:** | **IV** |
| **Date of Performance:** | **27 / 04 / 2021** | **Batch No:** | **B2** |
| **Faculty Name:** | **Prof. Bhargavi Kaslikar** | **Roll No:** | **1912052** |
| **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. |
| library ieee;  use ieee.std\_logic\_1164.all;  entity word\_pro\_vedant is  port(  x: in std\_logic\_vector(1 downto 0);  clk, rst : std\_logic;  y : out std\_logic\_vector(1 downto 0)  );  end word\_pro\_vedant;  architecture word\_pro\_vedant arch of word\_pro\_vedant is  type state is (reset, go\_left, go\_right, emergency);  signal ps, ns : state := reset;  begin  process(clk, rst)  begin  if(rst = '1')then  ps <= reset;  elsif(clk'event and clk = '1' )then  ps <= ns;  end if;  end process;  process(ps, x)  begin  if(x = "00") then  ns <= reset;  elsif (x = "01") then  ns <= go\_right;  elsif (x = "10") then  ns <= go\_left;  else  ns <= emergency;  end if;  end process;    process (ps) begin  case ps is  when reset => y <= "00";  when go\_left => y <= "10";  when go\_right => y <= "01";  when emergency => y <= "11";  end case;  end process;  end word\_pro\_vedant \_arch; |
| library ieee;  use ieee.std\_logic\_1164.all;  entity word\_pro\_vedant \_tb is  end word\_pro\_vedant \_tb;  architecture word\_pro\_vedant\_tb\_arch of word\_pro\_vedant tb is  component word\_pro\_vedant is  port(  x: in std\_logic\_vector(1 downto 0);  clk, rst : std\_logic;  y : out std\_logic\_vector(1 downto 0)  );  end component;  signal x,y:std\_logic\_vector(1 downto 0);  signal rst:std\_logic;  signal clk:std\_logic:='0';  begin  uut: word\_pro\_naik port map(x,clk,rst,y);  clk<= not clk after 5ns;  process  begin  rst <= '0';  x<="00";  wait for 10ns;  x<="01";  wait for 10ns;  x<="10";  wait for 10ns;  x<="11";  wait for 10ns;  x<="01";  wait for 10ns;  x<="11";  wait for 10ns;  x<="10";  wait for 10ns;  x<="01";  wait for 10ns;  x<="00";  wait for 10ns;  x<="11";  wait for 10ns;  x<="10";  wait for 10ns;  x<="00";  wait for 10ns;  end process;  end word\_pro\_vedant tb\_arch; |
|  |

|  |
| --- |
| **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  Ans:  Main code:  library ieee;  use ieee.std\_logic\_1164.all;  entity postlab\_vedant is  port(  clk,rst,i:in std\_logic;  y:out std\_logic  );  end postlab\_vedant;  architecture postlab\_vedant \_arch of postlab\_vedant is  type mystate is (s0, s1, s2, s3, s4);  signal ps,ns:mystate:=s0;  begin  process(clk,rst)  begin  if(rst='1')then  ps<= s0;  elsif(clk'event and clk = '1' )then  ps<= ns;  end if;  end process;  process(ps,i) begin  case ps is  when s0 =>  y <= '0';  if(i = '0')then  ns <= s0;  else  ns <= s1;  end if;  when s1 =>  y <= '0';  if(i = '0')then  ns <= s2;  else  ns <= s1;  end if;    when s2 =>  y <= '0';  if(i = '0')then  ns <= s3;  else  ns <= s1;  end if;  when s3 =>  y <= '0';  if(i = '0')then  ns <= s0;  else  ns <= s4;  end if;    when s4 =>  y <= '1';  if(i = '0')then  ns <= s0;  else  ns <= s1;  end if;  end case;  end process;  end postlab\_vedant\_arch;  TestBench:  library ieee;  use ieee.std\_logic\_1164.all;  entity postlab\_vedant\_tb is  end postlab\_vedant\_tb;  architecture postlab\_vedant\_tb \_arch of postlab\_vedant\_tb is  component postlab\_vedant is  port(  clk,rst,i:in std\_logic;  y:out std\_logic  );  end component;  signal rst,i,y:std\_logic;  signal clk:std\_logic:='0';  begin  t1: postlab\_naik port map(clk,rst,i,y);  process begin  clk <= '0';  wait for 5ns;  clk <= '1';  wait for 5ns;  end process;  process  begin  rst<= '0';  i<='1';  wait for 10ns;  i<='0';  wait for 10ns;  i<='0';  wait for 10ns;  i<='1';  wait for 10ns;  i<='1';  wait for 10ns;  i<='1';  wait for 10ns;  i<='0';  wait for 10ns;  i<='0';  wait for 10ns;  i<='1';  wait for 10ns;  i<='0';  wait for 10ns;  i<='1';  wait for 10ns;  i<='0';  wait for 10ns;  end process;  end postlab\_vedant\_tb\_arch;  State Machine:    Output:   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:**  **Thus, in this experiment we have implemented a word problem (Vehicle Tail Light) by using finite state machine in VHDL.** |

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