|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Hardware Description Language Lab (2UXL401)** | **Semester:** | **IV** |
| **Date of Performance:** | **02/03/2021** | **Batch No:** | **B2** |
| **Faculty Name:** | **Bhargavi maam** | **Roll No:** | **1912052** |
| **Faculty Sign & Date:** |  | **Grade/Marks:** |  |

**Experiment No: 3**

**Title:** Use of concurrent statements: Priority Encoders

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| Write a VHDL code for implementing a 8:3 priority encoder with enable input  Write a testbench to verify your results.  Also, generate a programming file and download the code on CPLD kit and verify the results.  To study various types of concurrent statements in VHDL code 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.  **CO 3**: Synthesize and Implement the designed circuits on CPLD/ FPGA. |

|  |
| --- |
| **Work to be uploaded** |
| VHDL code for priority encoder.  library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  use ieee.std\_logic\_arith.all;  entity encoder\_Vedant is  port(  x: in std\_logic\_vector(7 downto 0);  enable: in std\_logic;  y: out std\_logic\_vector(2 downto 0)    );  end encoder\_Vedant;  architecture encoder\_Vedant\_arch of encoder\_Vedant is  signal y\_s: std\_logic\_vector(2 downto 0);  begin  y\_s<= "111" when x(7)='1' else  "110" when x(6)='1' else  "101" when x(5)='1' else  "100" when x(4)='1' else  "011" when x(3)='1' else  "010" when x(2)='1' else  "001" when x(1)='1' else  "000";  y <= y\_s when enable='1' else  "UUU";  end encoder\_Vedant\_arch;  Test bench for priority encoder and simulation waveform of the same.  library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  use ieee.std\_logic\_arith.all;  entity encoder\_Vedant\_tb is  end encoder\_Vedant\_tb;  architecture encoder\_Vedant\_tb\_arch of encoder\_Vedant\_tb is  component encoder\_Vedant is  port(  x: in std\_logic\_vector(7 downto 0);  enable: in std\_logic;  y: out std\_logic\_vector(2 downto 0)  );  end component;    signal x\_in: std\_logic\_vector(7 downto 0);  signal en: std\_logic;  signal y\_out: std\_logic\_vector(2 downto 0);  begin  PE : encoder\_Vedant port map(x\_in,en, y\_out);    process  begin  en <= '1';  x\_in<="00000000";  wait for 10ns;    x\_in<="01010100";  wait for 10ns;    x\_in<="10011100";  wait for 10ns;    x\_in<="00010101";  wait for 10ns;    x\_in<="00000001";  wait for 10ns;  x\_in<="00111011";  wait for 10ns;    x\_in<="01111111";  wait for 10ns;    en<='0';    x\_in<="00001010";  wait for 10ns;    x\_in<="01100011";  wait for 10ns;  end process;  end encoder\_Vedant\_tb\_arch; |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:** |
| Upload Answer of following question before coming to next laboratory.  **Q1. What are the Shift Operators? Explain them with the help of examples**   |  |  |  | | --- | --- | --- | | **Operator** | **Use** | **Description** | | << | op1 << op2 | Shift bits of op1 left by distance op2; fills with zero bits on the right-hand side | | >> | op1 >> op2 | Shift bits of op1 right by distance op2; fills with highest (sign) bit on the left-hand side | | >>> | op1 >>> op2 | Shift bits of op1 right by distance op2; fills with zero bits on the left-hand side |   **Example**  13 >> 1;  The binary representation of the number 13 is 1101. The result of the shift operation is 1101 shifted to the right by one position — 110, or 6 in decimal. The left-hand bits are filled with 0s as needed.    **Q2. What is std\_logic\_arith package in library ieee? Which data conversion functions are available in the same?**  Functions defined in the std\_logic\_arith package provide conversion to and from the predefined VHDL data type INTEGER and arithmetic, comparison, and Boolean operations. With this package, you can perform arithmetic operations and numeric comparisons on array data types. The package defines some arithmetic operators (+, -, \*, and abs) and the relational operators (<, >, <=, >=, =, and /=).  The package also defines two major data types of its own; UNSIGNED and SIGNED.  Using this package, you can make the vector type (for example, std\_logic\_vector) synonymous with either SIGNED or UNSIGNED. This way, if you plan to use mostly UNSIGNED numbers, you do not need to convert your vector type to call UNSIGNED functions. |

|  |
| --- |
| **Conclusion:**  We wrote VHDL code for implementing a 8:3 priority encoder with enable input using Quartus. |

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