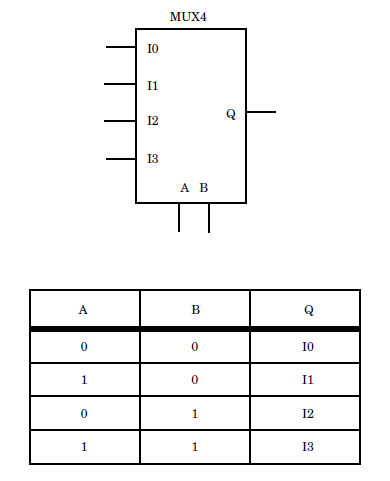
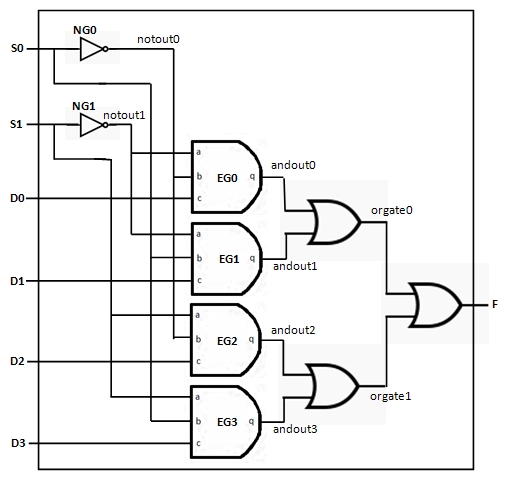
**Tutorial 6**

**Question 1:**

****

**Figure 1:** Top leveldesign and truth table for 4 to 1 MUX4

****

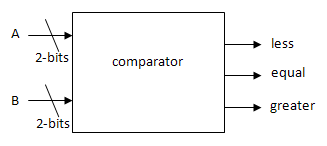
**Figure 2:** 4 to 1 Multiplexer RTL circuit

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S0** | **S1** | **D0** | **D1** | **D2** | **D3** | **F** |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 |

**Figure 3:** Partial Example output of a 4 to 1 multiplexer

**Figure 1** shows the top level design and the truth table for 4:1 Mux. The multiplexer can be represented by circuit shown in **Figure 2**. Write a VHDL code using the circuit given in **Figure 2**. Then, write another VHDL code which use sequential statement ‘if’ to describe the same circuit. Your output should be similar to the **Figure 3**. Write an assert statement to proof that both circuit has the same behavior. Try also the statement ‘case’, ‘with’ and ‘when’ to represent the 4:1 Mux circuit.

**Question 2:** Comparator

****

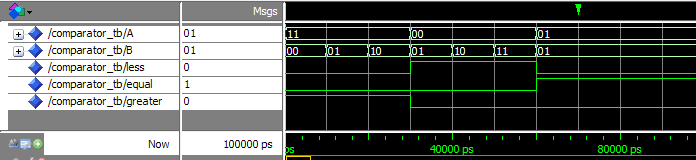
**Figure:** Comparator module

Create a VHDL code to represent the comparator shown above using the ‘if’ statement and create another one using ‘case’ statement. The table and the simulation result are given below:

**Table:** Behavioral of comparator module

|  |  |
| --- | --- |
| **Input** | **Behavioral** |
| If A < B | Less = ‘1’ |
| If A > B | Greater = ‘1’ |
| If A = B | Equal = ‘1’ |

**Result:**



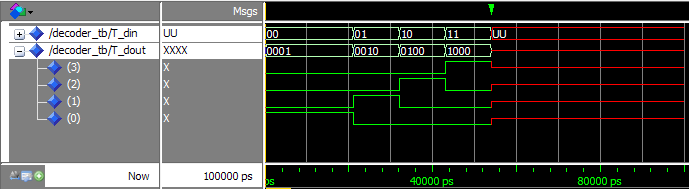
**Question 3:** Decoder

Table: The Decoder Behavioral

|  |  |
| --- | --- |
| din | dout |
| 00 | 0001 |
| 01 | 0010 |
| 10 | 0100 |
| 11 | 1000 |

Use VHDL ‘Case’ statement to describe the table above. Then, write a testbench with 100ns period. Your waveform result should look as below. In your testbench, write a code that detect and counts if there is any output error when the input is changed (e.g. if the input din=”01” and the output dout produce value other than “0010”, then the counter will increase). At the end of the code, write a program which report "Testbench of Adder completed successfully!" if there is no error ((i.e. counter=0)) and "Something wrong, try again" when there is an error (i.e. counter>0). Use the Assert statement to perform error checking.

**Result:**



**Question 4:** Variable versus Signal

**Table:** Example of variable and signal declaration

|  |  |
| --- | --- |
| Variable | Signal |
| proc1: process(d1,d2)  variable var\_s1: std\_logic;  begin  var\_s1 := d1 and d2;  out1 <= not var\_s1  end process; | signal sig\_s1: std\_logic;  proc2: process(d1,d2)  begin  sig\_s1 <= d1 and d2;  out2 <= not sig\_s1;  end process; |

The table above describes how variable and signal assignments can be programmed. Write a VHDL code and its test bench. From the simulation output, state their differences (if any).