Laboratory 3 Design Multi-bit Multiplexer and Programming a FPGA

For your report:
- The problem written in English
- The flowchart to solve the problem if it is necessary
- The design entry included (VHDL and Schematic)
- The RTL Viewer has to be included
- The simulation result for designed digital component
- The analysis for the simulation
- The pin assignment - the table for assigning the circuit inputs and outputs to specific pins on the FPGA
- The configuration for the FPGA Device (JTAG)
- The test table you designed to record and verify the designed circuit on hardware
- Photos can be included if any
- The conclusion

PART I Implement the eight-bit wide 2-to-1 multiplexer

In Figure 1(a) shows a sum-of-products circuit that implements a 2-to-1 multiplexer with a select input \( s \). If \( s = 0 \) the multiplexer’s output \( m \) is equal to the input \( x \), and if \( s = 1 \) the output is equal to \( y \). Figure 1(b) gives a truth table for this multiplexer, and Figure 1(c) shows its circuit symbol.

The single-bit multiplexer can be described by the VHDL statement:

\[
\begin{align*}
    m &\leftarrow (\text{NOT } s \text{ AND } x) \text{ OR } (s \text{ AND } y);
\end{align*}
\]
In your lab, you need to write a VHDL entity that includes eight assignment statements like the one shown above to describe the circuit given in Figure 2. This circuit has **two eight-bit inputs X and Y**, and produces the **eight-bit output M**.

If $s = 0$ then $M = X$, while if $s = 1$ then $M = Y$. We refer to this circuit as an eight-bit wide 2-to-1 multiplexer. It has the circuit symbol shown in Figure 2b, in which $X$, $Y$, and $M$ are depicted as eight-bit wires. Perform the steps shown below.

**Figure 2** An eight-bit 2-to-1 multiplexer.

1. Create a new Quartus II project `Lab3_1` for your circuit.
2. Include your VHDL file for the eight-bit wide 2-to-1 multiplexer in your project (follow the approach in previous lab and generate a digital component to be reused). Use switch SW17 on the DE2 board as the $s$ input, switches SW7–0 as the $X$ input and SW15–8 as the $Y$ input. Connect the $SW$ switches to the red lights LEDR and connect the output $M$ to the green lights LEDG7–0.
3. Include in your project the required pin assignments for the DE2 board. These assignments ensure that the input ports of your VHDL code will use the pins on the Cyclone II FPGA that are connected to the $SW$ switches, and the output ports of your VHDL code will use the FPGA pins connected to the LEDR and LEDG lights.
4. Compile the project.
5. Build a vector waveform called `Lab3_1.vwf`, and then perform a simulation to observe and verify the function.
6. View RTL from Netlist viewer
7. Download the compiled circuit into the FPGA chip. Design your test table and test the functionality of the eight-bit wide 2-to-1 multiplexer by toggling the switches and observing the LEDs.
-- Implements a 8-bit wide 2-to-1 multiplexer
-- inputs: SW7-0 represent the 8-bit input X, and SW15-8 represent Y
-- SW17 selects either X or Y to drive the output LEDs
-- outputs: LEDR17-0 show the states of the switches
-- outputs: LEDG7-0 shows the outputs of the multiplexers

-- IMPLEMENTATION I
-- LIBRARY NEEDED
ENTITY mux_8bit_2to1 IS
PORT  ( s : IN STD_LOGIC;
        X, Y : IN STD_LOGIC_VECTOR(7 DOWNTO 0); -- switches
        LEDR: OUT STD_LOGIC VECTOR (15 DOWNTO 0); red lights
        M : OUT STD_LOGIC VECTOR(7 DOWNTO 0)); -- green lights
END mux_8bit_2to1;
ARCHITECTURE Behavior OF mux_8bit_2to1 IS
BEGIN
        M(0) <= (NOT (s) AND x(0)) OR (s AND y(0));
        --your code is here
END Behavior;

---- IMPLEMENTATION II
-- LIBRARY NEEDED
ENTITY mux_8bit_2to1 IS
PORT  ( s : IN STD_LOGIC;
        X, Y : IN STD_LOGIC_VECTOR(7 DOWNTO 0); -- switches
        LEDR: OUT STD_LOGIC VECTOR (15 DOWNTO 0); red lights
        M : OUT STD_LOGIC VECTOR(7 DOWNTO 0)); -- green lights
END mux_8bit_2to1;
ARCHITECTURE Behavior OF mux_8bit_2to1 IS
BEGIN
        --use when... else statement
        M <= X when s = ‘0’ else
            Y when s= ‘1’ else
                NULL; --default case
END Behavior;

---- IMPLEMENTATION III
-- LIBRARY NEEDED
ENTITY mux_8bit_2to1 IS
PORT  ( s : IN STD_LOGIC;
        X, Y : IN STD_LOGIC_VECTOR(7 DOWNTO 0); -- switches
        LEDR: OUT STD_LOGIC VECTOR (15 DOWNTO 0); red lights
        M : OUT STD_LOGIC VECTOR(7 DOWNTO 0)); -- green lights
END mux_8bit_2to1;
ARCHITECTURE Behavior OF mux_8bit_2to1 IS
BEGIN
process (s)  --use case statement
begin
  case s is
    when '0' =>  M <= X;
    when '1' =>  M <= Y;
    when others => M <= "00000000";
  end case;
end process;
END Behavior;

PART II Implement the three-bit wide 5-to-1 multiplexer

Figure 1 showed a 2-to-1 multiplexer that selects between the two inputs $x$ and $y$. For this part consider a circuit in which the output $m$ has to be selected from five inputs $u$, $v$, $w$, $x$, and $y$. Figure 3a shows how we can build the required 5-to-1 multiplexer by using four 2-to-1 multiplexers. The circuit uses a 3-bit select input $s_2s_1s_0$ and implements the truth table shown in Figure 3b. A circuit symbol for this multiplexer is given in the figure 3c.

![Circuit Diagram](image)

**a) Circuit**

<table>
<thead>
<tr>
<th>$s_2$</th>
<th>$s_1$</th>
<th>$s_0$</th>
<th>$m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$u$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$v$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$w$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$x$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$y$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$y$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$y$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$y$</td>
</tr>
</tbody>
</table>

**b) Truth table**

![Truth Table](image)

**c) Symbol**

Figure 3 A 5-to-1 multiplexer.
Recall Figure 1 in PART I, an eight-bit wide 2-to-1 multiplexer can be built by using eight instances of a 2-to-1 multiplexer. Figure 4 applies this concept to define a three-bit wide 5-to-1 multiplexer. It contains three instances of the circuit in Figure 3a

![Figure 4. A three-bit wide 5-to-1 multiplexer.](image)

Perform the following steps to implement the three-bit wide 5-to-1 multiplexer.

1. Create a new Quartus II Lab3_2 project for your circuit.
2. Create a VHDL entity for the three-bit wide 5-to-1 multiplexer (follow the approach in previous lab and generate a digital component to be reused). Connect its select inputs to switches $SW_{17-15}$, and use the remaining 15 switches $SW_{14-0}$ to provide the five 3-bit inputs $U$, $V$, $W$, $X$, and $Y$. Connect the $SW$ switches to the red lights $LEDR$ and connect the output $M$ to the green lights $LEDG_{2-0}$.
3. Include in your project the required pin assignments for the DE2 board.
4. Compile the project.
5. Build a vector waveform called $Lab3_2.vwf$, and then perform a simulation to observe and verify the function.
6. View RTL from Netlist viewer
7. Download the compiled circuit into the FPGA chip. Design your test table and test the functionality of the three-bit wide 5-to-1 multiplexer by toggling the switches and observing the LEDs. Ensure that each of the inputs of $U$, $V$, $W$, $X$, and $Y$ can be properly selected as the output $M$.

---

**implements a 3-bit wide 5-to-1 multiplexer**

```vhdl
ENTITY mux_3bit_5to1 IS
  PORT ( S, U, V, W, X, Y : IN STD LOGIC VECTOR(2 DOWNTO 0);
         LEDR: OUT STD LOGIC VECTOR (14 DOWNTO 0);
         M : OUT STD LOGIC VECTOR(2 DOWNTO 0));
END mux_3bit_5to1;
```
ARCHITECTURE Behavior OF mux_3bit_5to1 IS
  -- your code is here
END Behavior;

PART III Implement the three-bit wide 6-to-1 multiplexer
Perform the following steps to implement the three-bit wide 6-to-1 multiplexer. You need to wire your switches and connect them to GPIO pins by using your breadboard.

1. Create a new Quartus II project Lab3_3 for your circuit.
2. Create a VHDL entity for the three-bit wide 6-to-1 multiplexer (follow the approach in previous parts and generate a digital component to be reused). Connect its select inputs to GPIO, and use the 18 switches SW17−0 to provide the six 3-bit inputs T, U, V, W, X, and Y. Connect the selectors to GPIO input power by using your switches and breadboard (VCC33 on DE2 board provides 3.3V), the on board SW switches to the red lights LEDR17−0 and connect the output M to the green lights LEDG2−0.
3. Include in your project the required pin assignments for the DE2 board.
4. Compile the project.
5. View RTL from Netlist viewer
6. Download the compiled circuit into the FPGA chip. Design your test table and test the functionality of the three-bit wide 6-to-1 multiplexer by toggling the GPIO input signals (selectors) and the on board switches and observing the LEDs. Ensure that each of the inputs of T, U, V, W, X, and Y can be properly selected as the output M.

---
- implements a 3-bit wide 6-to-1 multiplexer
- implement your vhdl code