National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering and Computer Science

Faculty Member: Dr. Rehan Ahmed Dated: 8/02/2023

Semester: 6th Section: BEE 12C

EE-421: Digital System Design

Lab 1: Introduction to Quartus Tool
Writing and Testing Simple Verilog Code

Group Members

		PLO4-CLO3		PLO5 - CLO4	PLO8 - CLO5	PLO9 - CLO6
Name	**Reg. No**	**Viva / Quiz / Lab Performance**	**Analysis of data in Lab Report**	**Modern Tool Usage**	**Ethics and Safety**	**Individual and Teamwork**
		5 Marks	**5 Marks**	**5 Marks**	**5 Marks**	**5 Marks**
Danial Ahmad	331388					
Muhammad Umer	345834					
Tariq Umar	334943					

1 Table of Contents

2 Switches, Lights, and Multiplexers

2.1 Objectives

The purpose of this exercise is to learn how to connect simple input and output devices to an FPGA chip and implement a circuit that uses these devices. We will use the switches on the DE-series boards as inputs to the circuit. We will use light emitting diodes (LEDs) and 7-segment displays as output devices.

2.2 Introduction

The field of digital design has been revolutionized by the advent of Field-Programmable Gate Arrays (FPGAs), which allow designers to create complex digital circuits using programmable logic. In this lab exercise, we will be using Intel's Quartus Prime software to design and implement a circuit that utilizes switches, lights, and multiplexers. The objective of this exercise is to gain hands-on experience in designing and implementing digital circuits using FPGAs, and to understand how switches, lights, and multiplexers can be used in a digital circuit. By the end of this lab, we will have a deeper understanding of the basic components of digital circuits and how they can be used to create complex systems. Additionally, we will have gained valuable experience in using Intel's Quartus Prime software, which is a leading platform for digital design.

2.3 Software

Quartus Prime is a comprehensive design software developed by Intel Corporation for designing digital circuits using Field-Programmable Gate Arrays (FPGAs). It is a leading software platform in the field of digital design, offering a range of advanced tools and features that enable users to easily create, debug, and verify complex digital circuits. With Quartus Prime, users can benefit from a streamlined design flow that facilitates the creation of digital circuits from concept to implementation. It provides an intuitive graphical user interface that allows users to easily design, test, and debug their circuits. Additionally, Quartus Prime supports a variety of popular programming languages, making it a versatile platform for digital designers of all levels.

3 Lab Procedure

3.1 Part 1

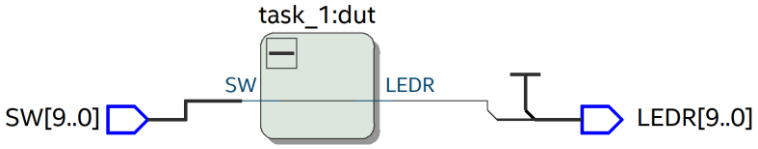
The DE-series boards have hardwired connections between its FPGA chip and the switches and lights. Perform the following steps to implement a circuit corresponding to the code in Figure 1 on the DE-series boards.

1. Create a new Quartus project for your circuit. Select the target chip that corresponds to your DE-series board.
2. Create a Verilog module for the code in Figure 1 and include it in your project.
3. Include in your project the required pin assignments for your DE-series board, as discussed above. Compile the project.
4. Download the compiled circuit into the FPGA chip by using the Quartus Programmer tool (the procedure for using the Programmer tool is described in the tutorial Quartus Introduction). Test the functionality of the circuit by toggling the switches and observing the LEDs.

```
// Module that connects ten switches and lights
module  part1 (SW, LEDR);
    input [9:0] SW;          // slide switches
    output [9:0] LEDR;       // red LEDs

    assign LEDR = SW;
endmodule
```

Figure 1: Verilog code that uses ten switches and lights.

```
module standard (
    input [9:0] SW,      // 10 switches
    output [9:0] LEDR    // 10 LEDs

);

   task_1 dut (      // instantiate task_1 module
       .SW  (SW),
       .LEDR(LEDR)
   );

endmodule

module task_1 (      // task_1 module
    input SW,
    output LEDR
);

   assign LEDR = SW;

endmodule
```



3.2 Part 2

You are to write a Verilog module that includes four assignment statements to describe the circuit given in Figure 2a. This circuit has two four-bit inputs, X and Y, and produces the four-bit output M. If s = 0 then M = X, while if s = 1 then M = Y. We refer to this circuit as a four-bit wide 2-to-1 multiplexer. It has the circuit symbol shown in Figure 2b, in which X, Y, and M are depicted as four-bit wires.

1. Create a new Quartus project for your circuit.
2. Include your Verilog file for the four-bit wide 2-to-1 multiplexer in your project. Use switch SW_9 as the (s) input, switches $SW_{3–0}$ as the X input and $SW_{7–4}$ as the Y input. Display the value of the input (s) on $LEDR_9$, connect the output M to $LEDR_{3–0}$, and connect the unused LEDR lights to the constant value 0.
3. Include in your project the required pin assignments for your DE-series board. As discussed in Part I, these assignments ensure that the ports of your Verilog code will use the pins on the FPGA chip that are connected to the SW switches and LEDR lights.
4. Compile the project, and then download the resulting circuit into the FPGA chip. Test the functionality of the four-bit wide 2-to-1 multiplexer by toggling the switches and observing the LEDs.

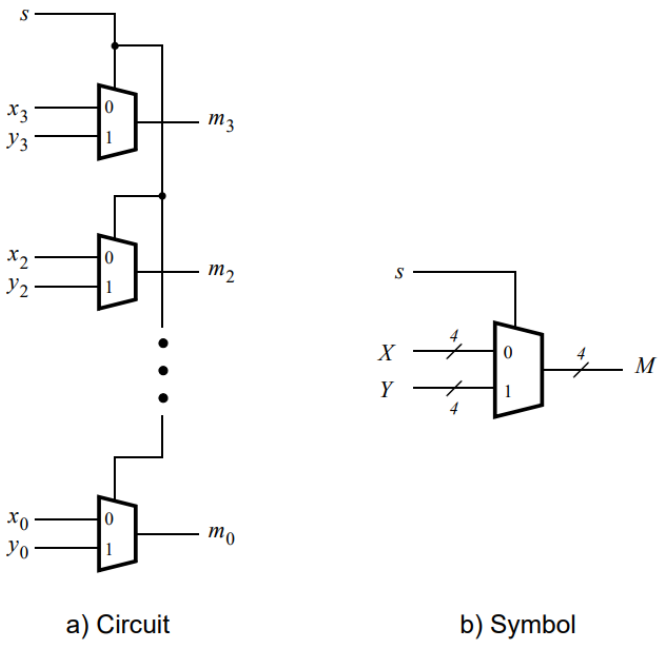
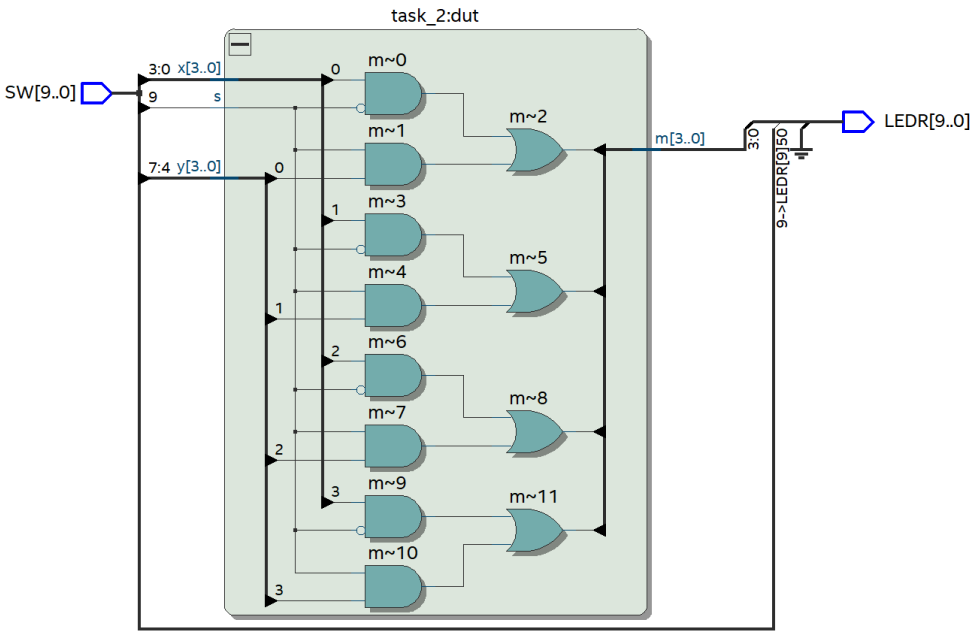


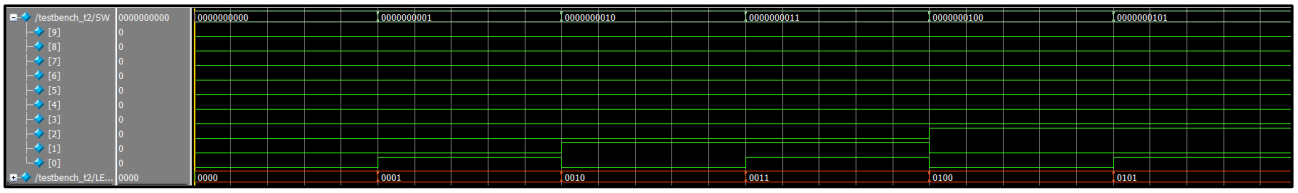
Figure 2: A four-bit wide 2-to-1 multiplexer.

```
module standard (
    input [9:0] SW,      // 10 switches
    output [9:0] LEDR    // 10 LEDs

);

   wire [3:0] x = SW[3:0];
   wire [3:0] y = SW[7:4];
   wire s = SW[9];
   wire [3:0] m;
   assign LEDR[3:0] = m;
   assign LEDR[9] = s;

   task_2 dut (      // instantiate task_2 module
       .x(x),
       .y(y),
       .m(m),
       .s(s)
   );

endmodule

module task_2 (      // 4-bit mux
    input [3:0] x,
    y,
    input s,
    output [3:0] m
);

   assign m[0] = (~s & x[0]) | (s & y[0]);
   assign m[1] = (~s & x[1]) | (s & y[1]);
   assign m[2] = (~s & x[2]) | (s & y[2]);
   assign m[3] = (~s & x[3]) | (s & y[3]);

endmodule
```



To test the 2-bit wide, 4:1 multiplexer, we implement the following testbench:

```
module testbench_t2 ();
   reg  [9:0] SW;       // For input [3:] x, y
   wire [3:0] LEDR; // For output [3:0] m

   standard dut (
       .SW  (SW),
       .LEDR(LEDR)
   );

   integer i;

   initial begin
      for (i = 0; i < 2 ** 10; i = i + 1) begin
         {SW} = i;
         #10;
      end
   end

endmodule
```



3.3 Part 3

Perform the following steps to implement the two-bit wide 4-to-1 multiplexer, as shown in Figure 3.

1. Create a new Quartus project for your circuit.
2. Create a Verilog module for the two-bit wide 4-to-1 multiplexer. Connect the select inputs to switches SW_{9-8} and use switches SW_{7-0} to provide the four 2-bit inputs U to X. Connect the output M to the red lights LEDR1−0.
3. Include in your project the required pin assignments for your DE-series board. Compile the project.

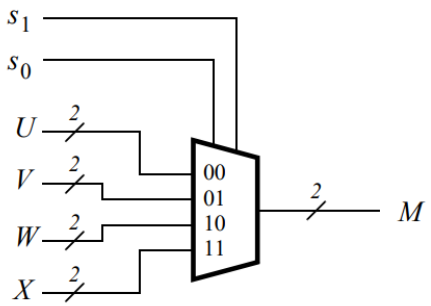
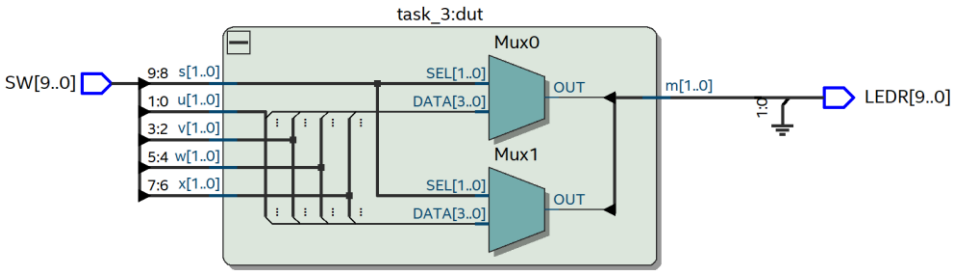


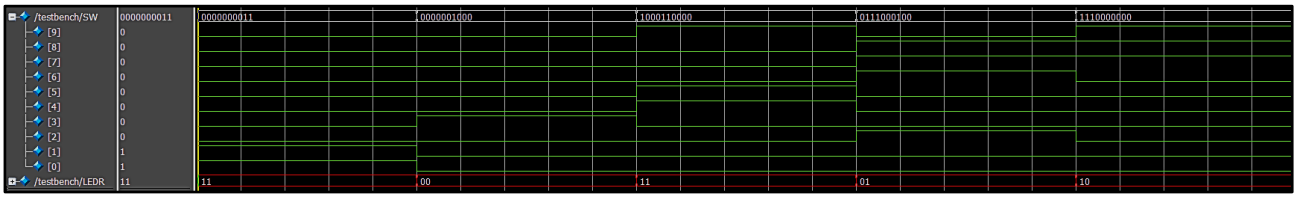
Figure 3: A two-bit wide 4-to-1 multiplexer.

```
module standard (

    input [9:0] SW,      // 10 switches
    output [9:0] LEDR    // 10 LEDs

);

   wire [1:0] u = SW[1:0];
   wire [1:0] v = SW[3:2];
   wire [1:0] w = SW[5:4];
   wire [1:0] x = SW[7:6];
   wire [1:0] s = SW[9:8];
   wire [1:0] m;
   assign LEDR[1:0] = m;

   task_3 dut (      // instantiate task_3 module
       .s(s),
       .u(u),
       .v(v),
       .w(w),
       .x(x),
       .m(m)
   );


endmodule

module task_3 (      // 2-bit wide 4:1 multiplexer
    input [1:0] s,
    input [1:0] u,
    v,
    w,
    x,

    output reg [1:0] m
);

   always @(*) begin     // combinational logic
      case (s)
         2'b00: m = u;
         2'b01: m = v;
         2'b10: m = w;
         2'b11: m = x;
      endcase
   end

endmodule
```



To test the 2-bit wide, 4:1 multiplexer, we implement the following testbench:

```
module testbench_t3 ();
   reg  [9:0] SW;
   wire [1:0] LEDR;

   standard dut (
       .SW  (SW),
       .LEDR(LEDR)
   );

   integer i;

   initial begin
      for (i = 0; i < 2 ** 10; i = i + 1) begin
         {SW} = i;
         #10;
      end
   end

endmodule
```



4 Conclusion

In conclusion, this lab exercise provided an opportunity to gain hands-on experience in designing and implementing digital circuits using Field-Programmable Gate Arrays (FPGAs) and Intel's Quartus Prime software. Through the integration of switches, lights, and multiplexers, we were able to understand how these basic components can be used to create complex digital circuits. The exercise also allowed us to explore the design flow of Quartus Prime, from creating the circuit design to programming the FPGA.