**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Dr. Rehan Ahmed  **Dated:** 15/02/2023

**Semester:**  6th **Section:** BEE 12C

**EE-421:** **Digital System Design**

**Lab 2: Introduction to Altera DE-10 Board**

Learn to connect simple I/O devices to an FPGA chip

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 |  |  |  |  |  |

**Table of Contents**

[2 Switches, Lights, and Multiplexers 3](#_Toc127568538)

[2.1 Objectives 3](#_Toc127568539)

[2.2 Introduction 3](#_Toc127568540)

[2.3 Software 3](#_Toc127568541)

[3 Lab Procedure 4](#_Toc127568542)

[3.1 Part I 4](#_Toc127568543)

[3.2 Part II 6](#_Toc127568544)

[3.3 Part III 10](#_Toc127568545)

[4 Conclusion 14](#_Toc127568546)

# Switches, Lights, and Multiplexers

## 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.

## 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.

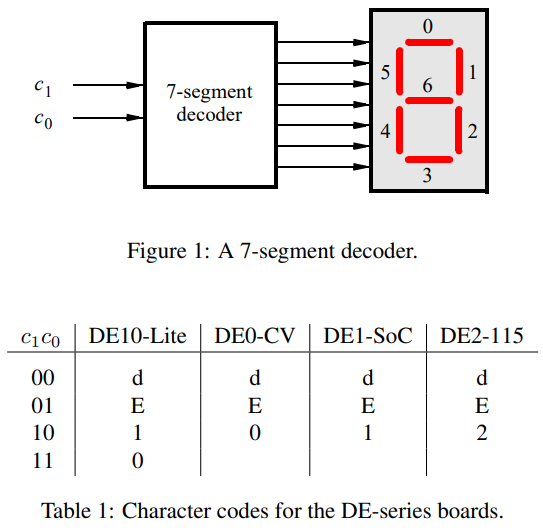
## Software

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.

# Lab Procedure

## Part I

The objective of this part is to display a character on a 7-segment display. The specific character displayed depends on a two-bit input. Figure 1 shows a 7-segment decoder module that has the two-bit input . This decoder produces seven outputs that are used to display a character on a 7-segment display. Table 1 lists the characters that should be displayed for each valuation of for your DE-series board. Note that in some cases the ‘blank’ character is selected for code 11.



1. Create a new Quartus project for your circuit.
2. Create a Verilog module for the 7-segment decoder. Connect the c1c0 inputs to switches SW1−0 and connect the outputs of the decoder to the HEX0 display on your DE-series board. The segments in this display are called HEX00, HEX01, . . ., HEX06, corresponding to Figure 1. You should declare the 7-bit port 1 in your Verilog code so that the names of these outputs match the corresponding names in your board’s user manual and pin assignment file.
3. After making the required pin assignments, compile the project.
4. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by toggling the SW1−0 switches and observing the 7-segment display.

module *standard* (

*input*  [9:0] SW,

*output* [6:0] HEX0

);

*task\_1* dut (

       .c1 (SW[1]),

       .c0 (SW[0]),

       .hex(HEX0)

   );

endmodule

module *task\_1* (

*input* c1,

    c0,

*output* [6:0] hex

);

   assign hex[0] = (~c1 & ~c0) | (c1 & ~c0);

   assign hex[1] = (~c1 & c0);

   assign hex[2] = (~c1 & c0);

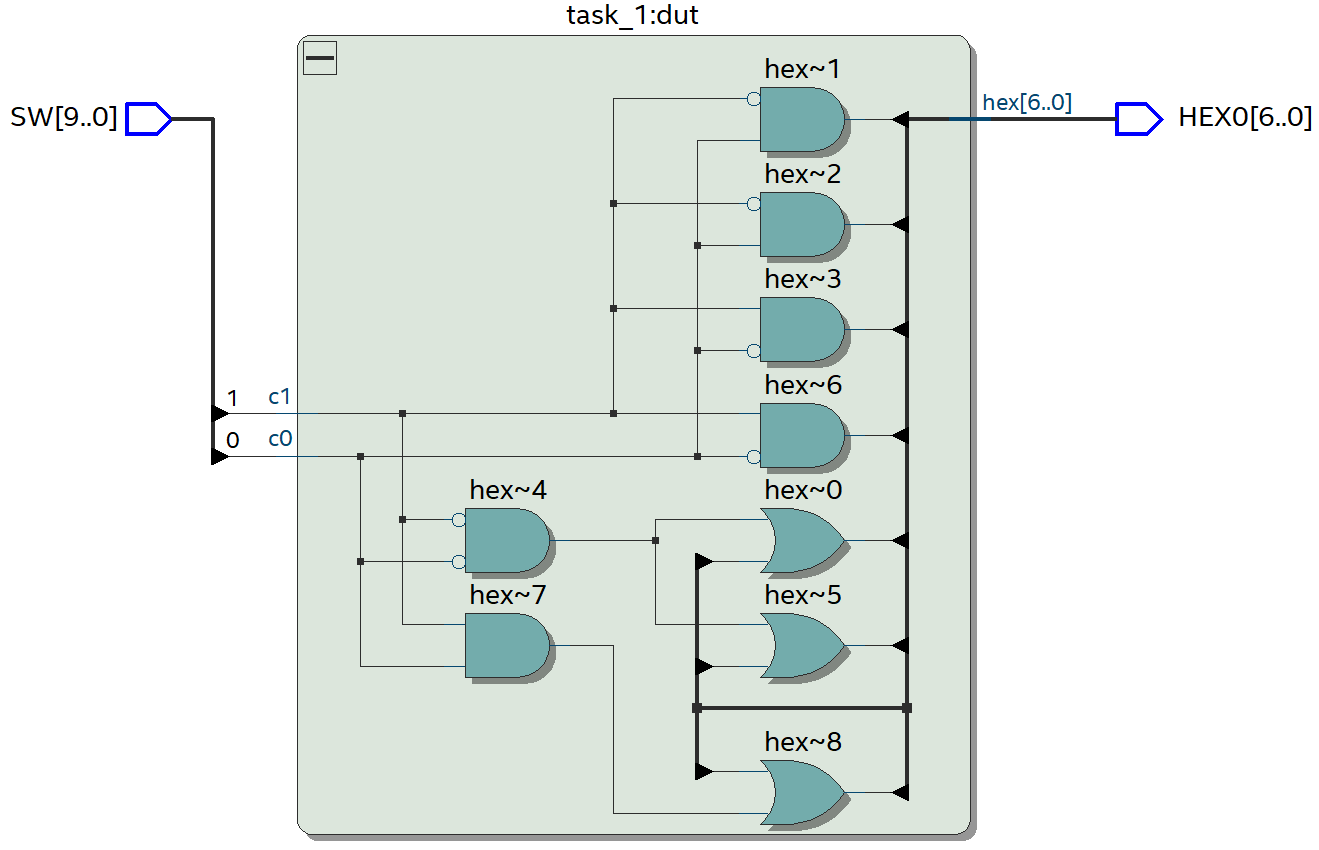
   assign hex[3] = (c1 & ~c0);

   assign hex[4] = (c1 & ~c0);

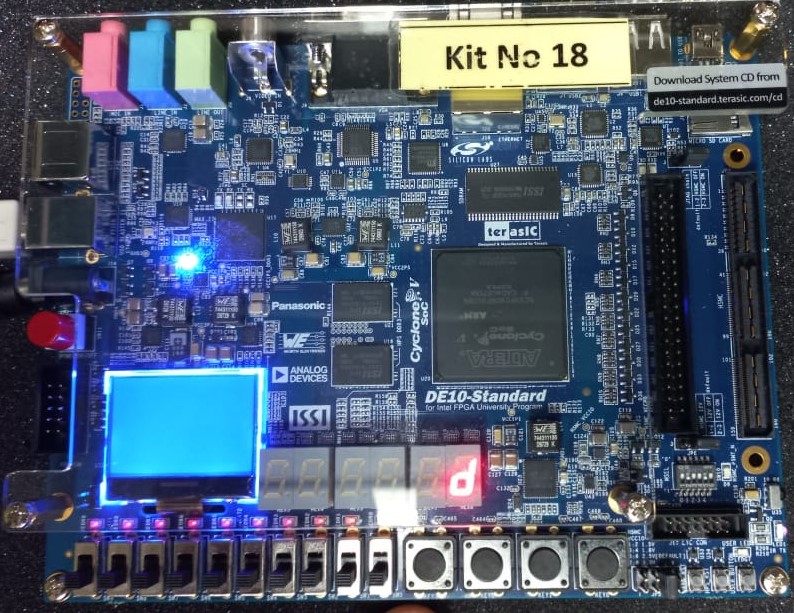
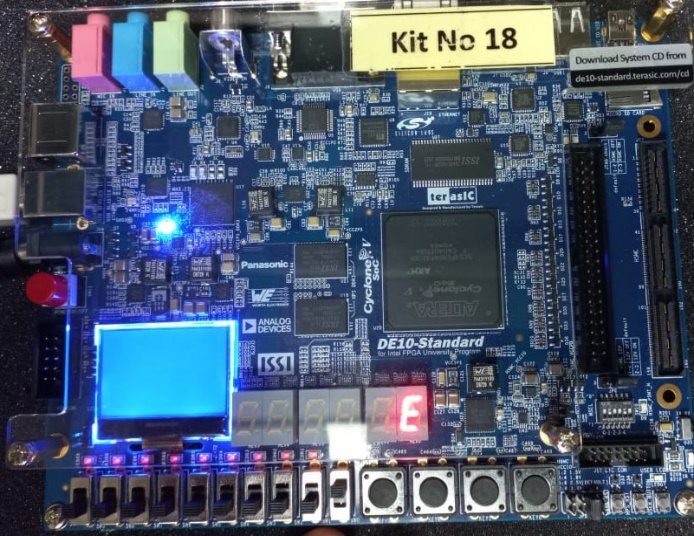
   assign hex[5] = (~c1 & ~c0) | (c1 & ~c0);

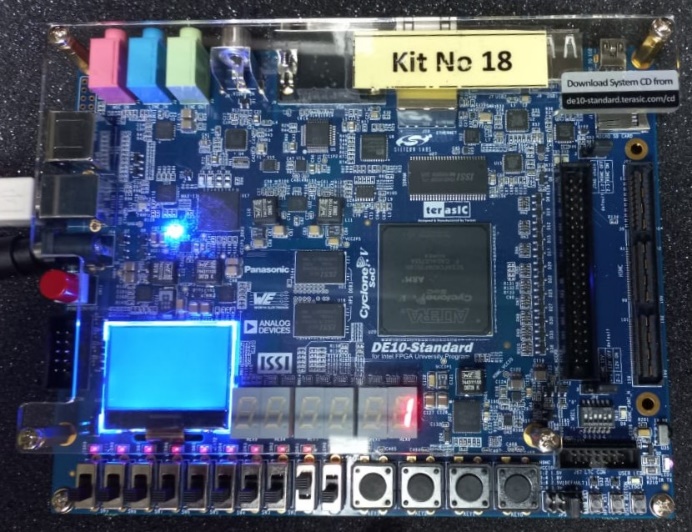
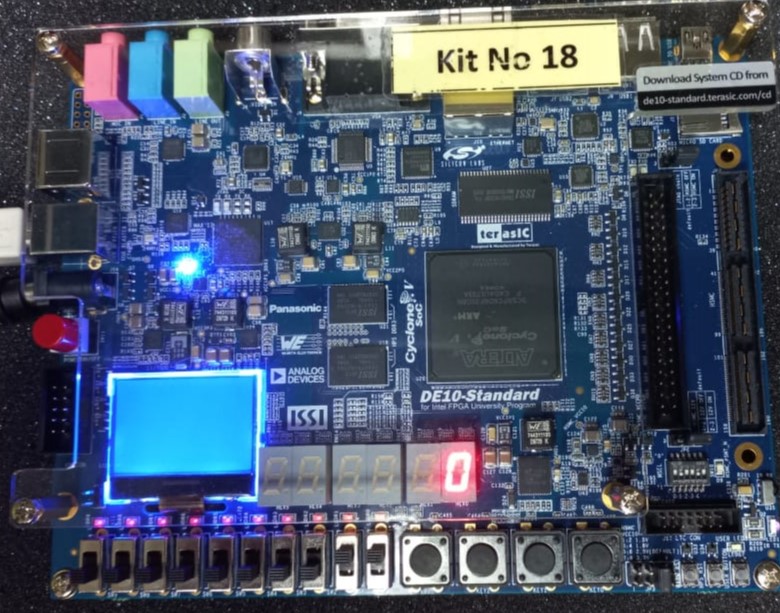
   assign hex[6] = (c1 & ~c0) | (c1 & c0);

endmodule



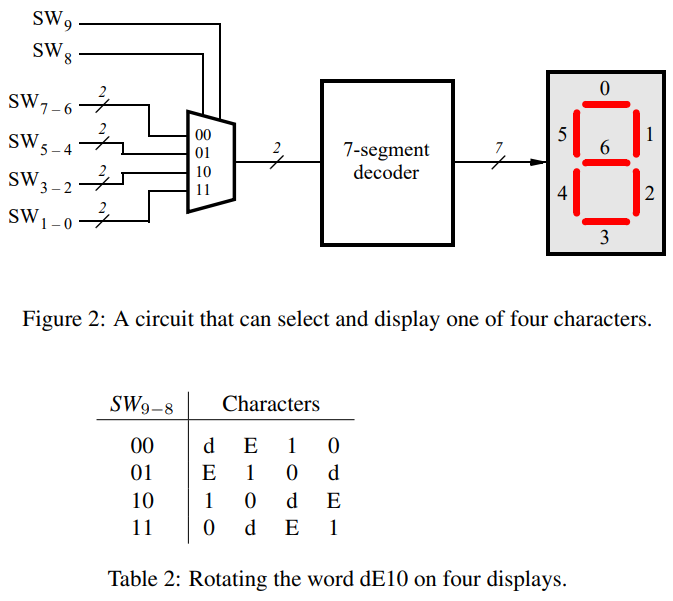
Following are sample hardware demonstrations of Part I on the DE-10 board:

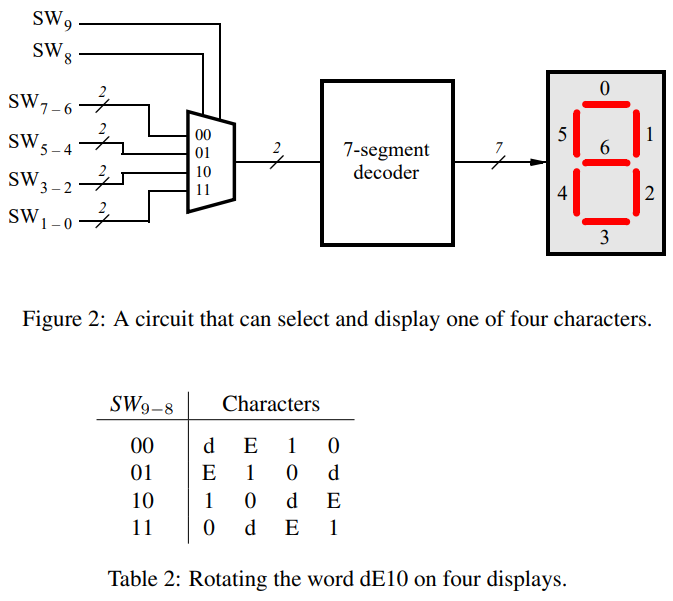
 

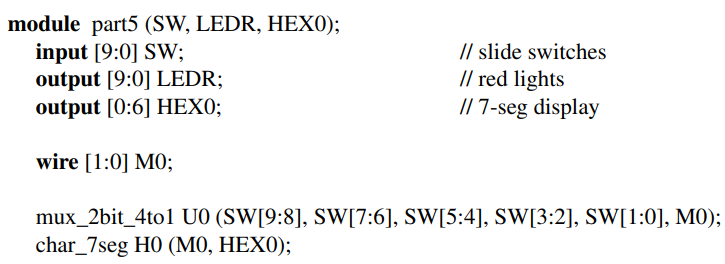
## Part II

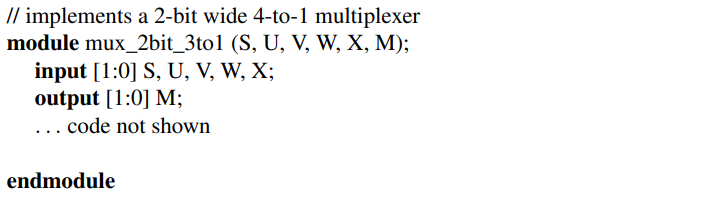
Consider the circuit shown in Figure 2. It uses a two-bit wide 4-to-1 multiplexer to enable the selection of four characters that are displayed on a 7-segment display. Using the 7-segment decoder from Part IV this circuit can display the characters d, E, 0, 1, 2, or ‘blank’ depending on your DE-series board. The character codes are set according to Table 1 by using the switches SW7−0, and a specific character is selected for display by setting the switches SW9−8.

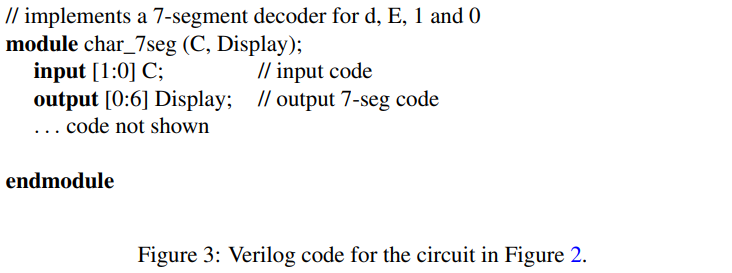




1. Create a new Quartus project for your circuit.
2. Include your Verilog module in the Quartus project. Connect the switches SW9−8 to the select inputs of each of the four instances of the two-bit wide 4-to-1 multiplexers. Also connect SW7−0 to each instance of the multiplexers as required to produce the patterns of characters shown in Table 1. Connect the SW switches to the red lights LEDR and connect the outputs of the four multiplexers to the 7-segment displays HEX3, HEX2, HEX1, and HEX0.
3. Include the required pin assignments for your DE-series board for all switches, LEDs, and 7-segment displays. Compile the project.
4. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by setting the proper character codes on the switches SW7−0 and then toggling SW9−8 to observe the rotation of the characters.







module *standard* (

*input*  [9:0] SW,

*output* [6:0] HEX0,

*output* [6:0] HEX1,

*output* [6:0] HEX2,

*output* [6:0] HEX3

);

   wire [27:0] HEX;

*task\_2* dut (

       .switch(SW),

       .hex(HEX)

   );

   assign HEX0 = HEX[6:0], HEX1 = HEX[13:7], HEX2 = HEX[20:14], HEX3 = HEX[27:21];

endmodule

module *task\_2* (

*input*  [ 9:0] switch,

*output* [27:0] hex

);

   wire [1:0] s1, s2, s3, s4;

   assign s1 = switch[9:8], s2 = switch[9:8] + 1, s3 = switch[9:8] + 2, s4 = switch[9:8] + 3;

   wire [7:0] s;

   wire [7:0] m;

   wire [1:0] u = switch[1:0], v = switch[3:2], w = switch[5:4], x = switch[7:6];

   assign s = {s1, s2, s3, s4};

   genvar i;

   generate

      for (i = 1; i < 8; i = i + 2) begin : mux

*mux\_2b\_4to1* muxstage (

             .s(s[i:i-1]),

             .u(u),

             .v(v),

             .w(w),

             .x(x),

             .m(m[i:i-1])

         );

      end

   endgenerate

   generate

      for (i = 1; i < 8; i = i + 2) begin : seg\_decoder

*task\_1* decoderstage (

             .c1 (m[i]),

             .c0 (m[i-1]),

             .hex(hex[7\*((i+1)/2)-1:7\*((i-1)/2)])

         );

      end

   endgenerate

endmodule

module *mux\_2b\_4to1* (

*input* [1:0] s,

*input* [1:0] u,

*input* [1:0] v,

*input* [1:0] w,

*input* [1:0] x,

*output* reg [1:0] m

);

   always @(\*) begin

      case (s)

         2'b00: begin

            m[0] = u[0];

            m[1] = u[1];

         end

         2'b01: begin

            m[0] = v[0];

            m[1] = v[1];

         end

         2'b10: begin

            m[0] = w[0];

            m[1] = w[1];

         end

         2'b11: begin

            m[0] = x[0];

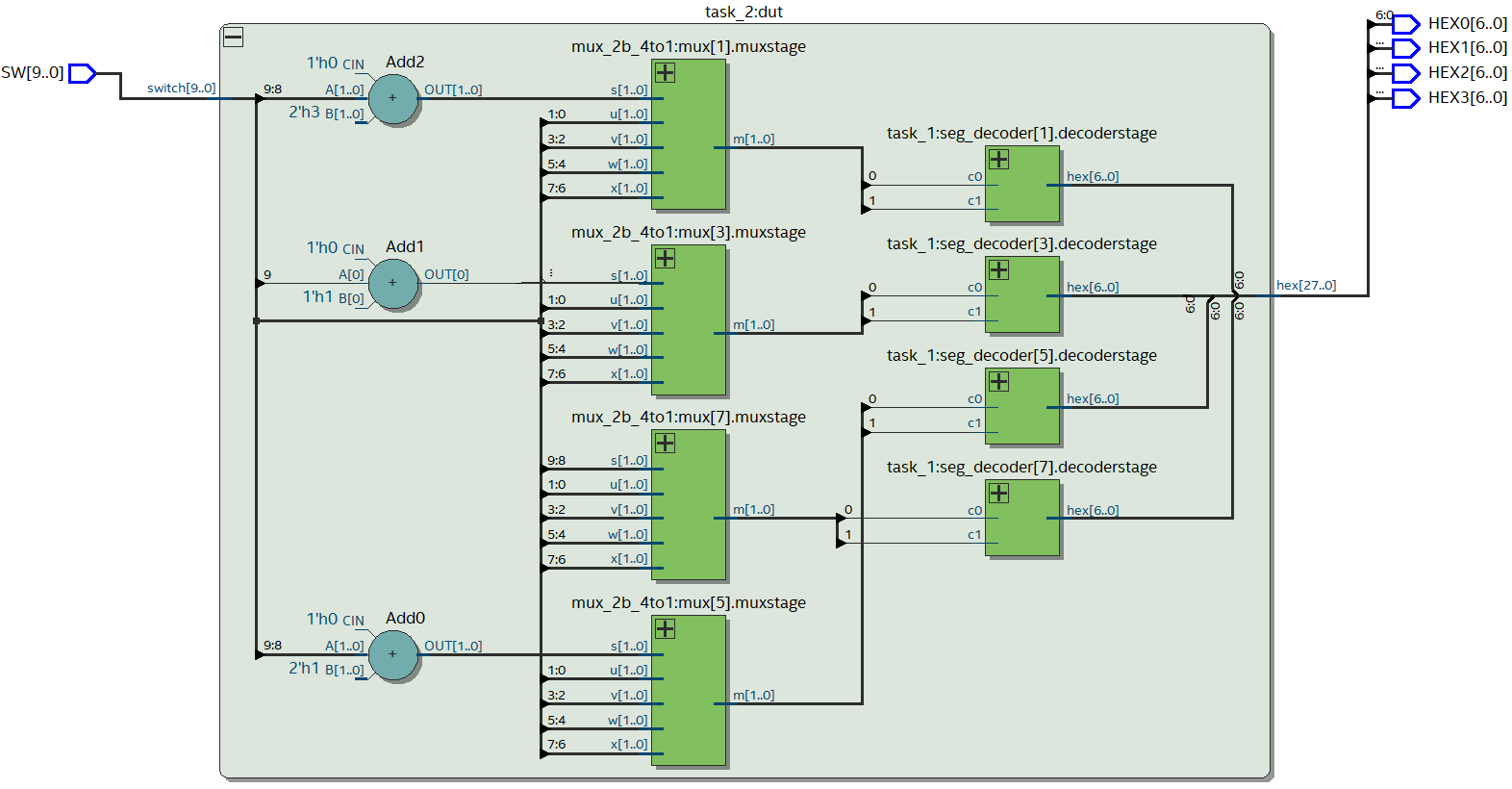
            m[1] = x[1];

         end

      endcase

   end

endmodule



To test the functional relationship between the switches and the 7-segment displays, we implement the following testbench:

module *t2\_testbench* ();

   reg  [ 9:0] switch;

   wire [27:0] hex;

*task\_2* dut (

       .switch(switch),

       .hex(hex)

   );

   integer i;

   initial begin

      for (i = 0; i < 2 \*\* 10; i = i + 1) begin

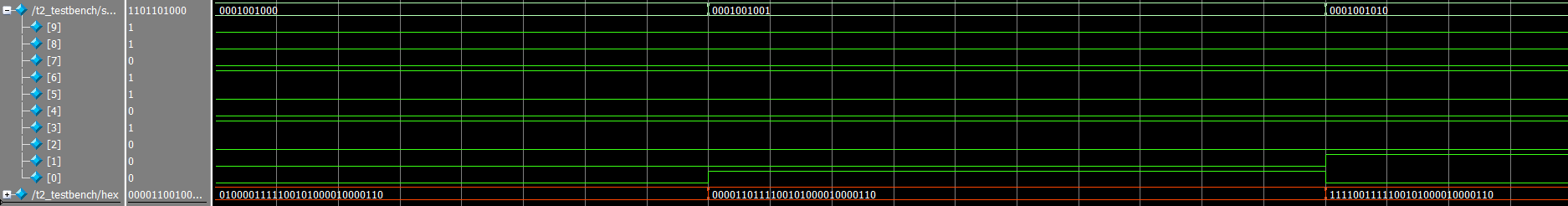
         {switch} = i;

         #10;

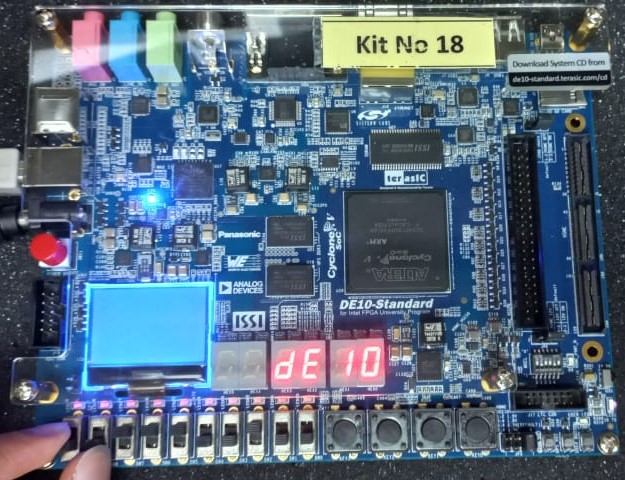
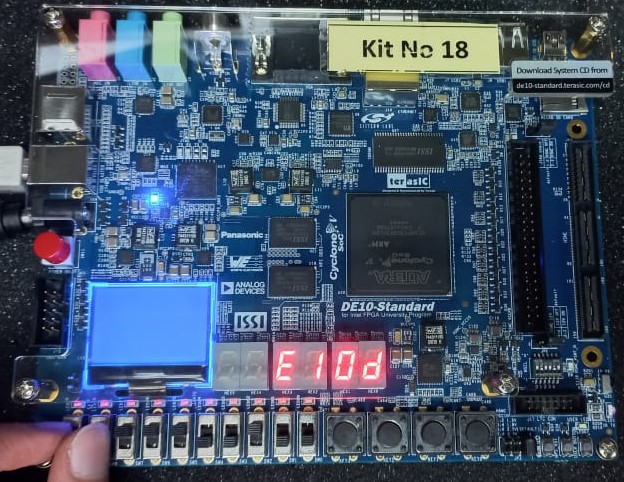
      end

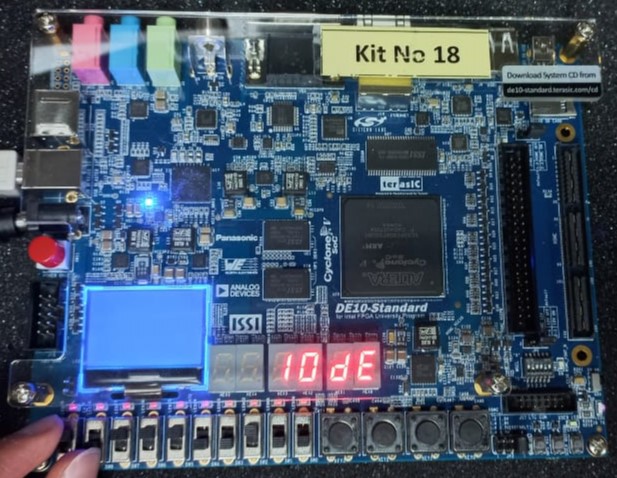
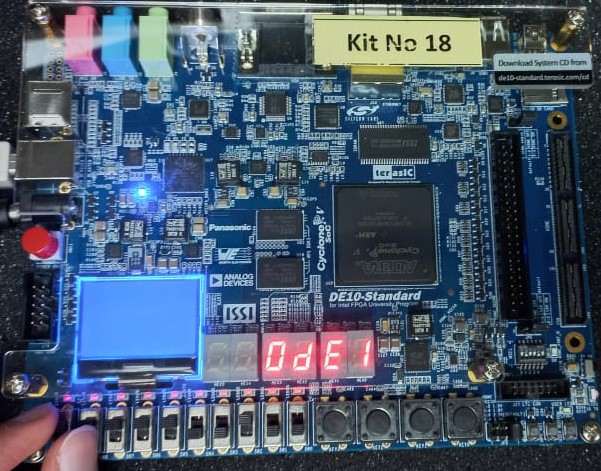
   end

endmodule



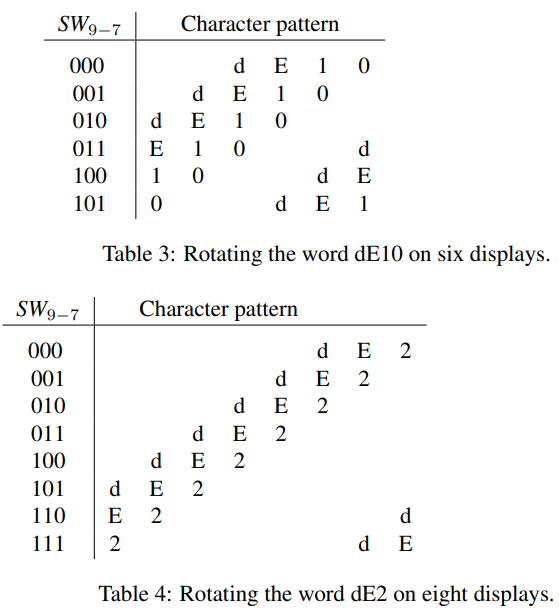
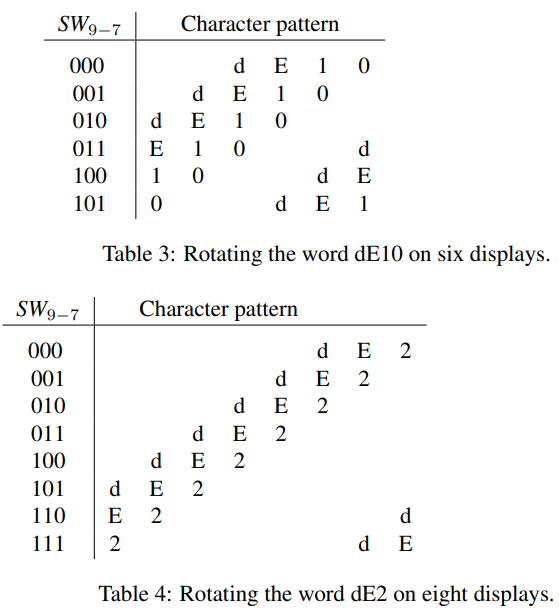
Following are sample hardware demonstrations of Part II on the DE-10 board:

## Part III

Extend your design from Part V so that it uses all 7-segment displays on your DE-series board. Your circuit needs to display a three- or four-letter word, corresponding to Table 2, using ’blank’ characters for unused displays. Implement rotation of this word from right-to-left as indicated in Table 3 and Table 4. To do this, you will need to connect 6-to-1 multiplexers to each of six 7-segment display decoders for the DE10-Lite, DE0-CV and DE1-SoC. Note that for the DE10-Lite you will need to use 3-bit codes for your characters, because five characters are needed when including the ’blank’ character (your 7-segment decoder will have to use 3-bit codes, and you will need to use 3-bit wide 6-to-1 multiplexers).



1. Create a new Quartus project for your circuit.
2. Include your Verilog module in the Quartus project. Connect the switches SW9−7 to the select inputs of each instance of the multiplexers in your circuit. Connect the outputs of your multiplexers to the 7-segment displays HEX5, . . ., HEX0 of the DE10- Lite, DE0-CV and DE1-SoC or HEX7, . . ., HEX0 for the DE2-115.
3. Include the required pin assignments for your DE-series board for all switches, LEDs, and 7-segment displays. Compile the project.
4. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by toggling SW9−7 to observe the rotation of the characters.

module *standard* (

*input*  [9:0] SW,

*output* [6:0] HEX0,

*output* [6:0] HEX1,

*output* [6:0] HEX2,

*output* [6:0] HEX3,

*output* [6:0] HEX4,

*output* [6:0] HEX5

);

   wire [41:0] HEX;

*task\_3* dut (

       .switch(SW),

       .hex(HEX)

   );

   assign HEX0 = HEX[6:0],

       HEX1 = HEX[13:7],

       HEX2 = HEX[20:14],

       HEX3 = HEX[27:21],

       HEX4 = HEX[34:28],

       HEX5 = HEX[41:35];

endmodule

module *task\_3* (

*input*  [ 9:0] switch,

*output* [41:0] hex

);

   wire [17:0] m;

   wire [17:0] s;

   wire [2:0] s1, s2, s3, s4, s5, s6;

   assign s1 = switch[9:7] + 5,

       s2 = switch[9:7] + 4,

       s3 = switch[9:7],

       s4 = switch[9:7] + 1,

       s5 = switch[9:7] + 2,

       s6 = switch[9:7] + 3;

   wire [2:0] u = 3'b011, v = 3'b010, w = 3'b001, x = 3'b000, y = 3'b111, z = 3'b111;

   assign s = {s1, s2, s3, s4, s5, s6};

   genvar i;

   generate

      for (i = 2; i < 18; i = i + 3) begin : mux

*mux\_3b\_6to1* muxstage (

             .s(s[i:i-2]),

             .u(u), .v(v),

             .w(w), .x(x),

             .y(y), .z(z),

             .m(m[i:i-2])

         );

      end

   endgenerate

   generate

      for (i = 2; i < 18; i = i + 3) begin : seg\_decoder

*t3\_decoder* decoderstage (

             .c2 (m[i]),

             .c1 (m[i-1]),

             .c0 (m[i-2]),

             .hex(hex[7\*((i+1)/3)-1:7\*((i-1)/3)])

         );

      end

   endgenerate

endmodule

module *mux\_3b\_6to1* (

*input* [2:0] s,

*input* [2:0] u,

*input* [2:0] v,

*input* [2:0] w,

*input* [2:0] x,

*input* [2:0] y,

*input* [2:0] z,

*output* reg [2:0] m

);

   always @(\*) begin

      case (s)

         3'b000:  m = u;

         3'b001:  m = v;

         3'b010:  m = w;

         3'b011:  m = x;

         3'b100:  m = y;

         3'b101:  m = z;

         default: m = 3'b111;

      endcase

   end

endmodule

module *t3\_decoder* (

*input* c2,

    c1,

    c0,

*output* reg [6:0] hex

);

   wire [2:0] c = {c2, c1, c2};

   always @(\*) begin

      if (c == 3'b111) hex = 7'b1111111;

      else hex[0] = (~c1 & ~c0) | (c1 & ~c0);

      hex[1] = (~c1 & c0);

      hex[2] = (~c1 & c0);

      hex[3] = (c1 & ~c0);

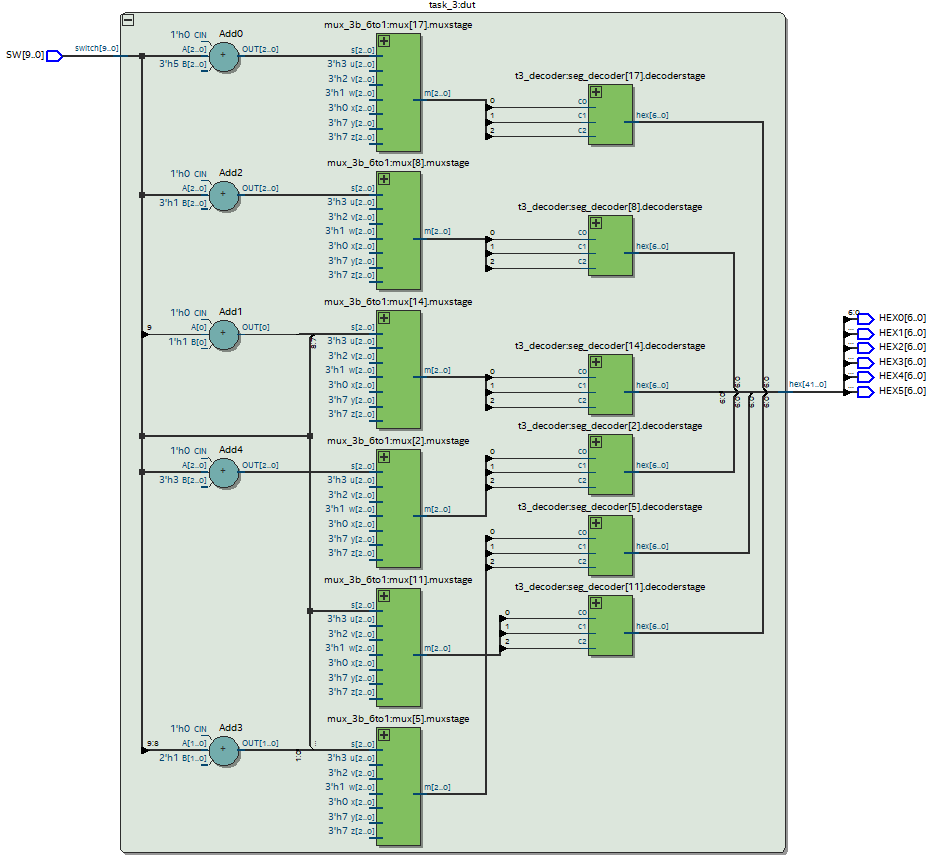
      hex[4] = (c1 & ~c0);

      hex[5] = (~c1 & ~c0) | (c1 & ~c0);

      hex[6] = (c1 & ~c0) | (c1 & c0);

   end

endmodule



To test the functional relationship between the switches and the 7-segment displays, we implement the following testbench:

module *t3\_testbench* ();

   reg  [ 9:0] switch;

   wire [41:0] hex;

*task\_3* dut (

       .switch(switch),

       .hex(hex)

   );

   integer i;

   initial begin

      for (i = 0; i < 2 \*\* 3; i = i + 1) begin

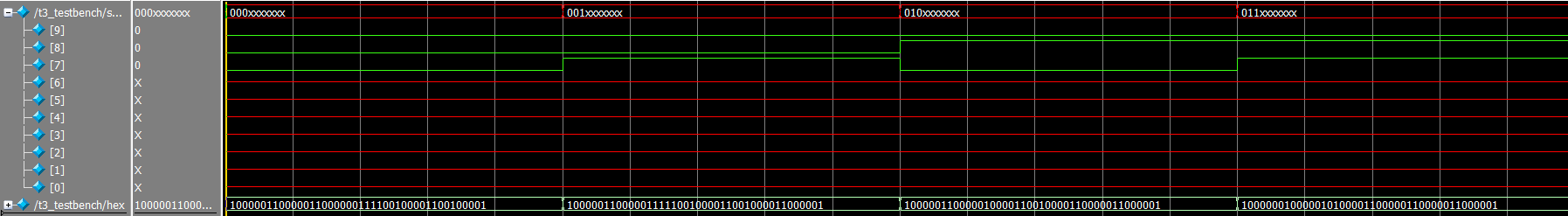
         {switch[9:7]} = i;

         #10;

      end

   end

endmodule



# 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.