EE25266 – ASIC/FPGA Chip Design

Mahdi Shabany
Electrical Engineering Department
Sharif University of Technology

Assignment #1

Introduction to Computer-Aided Design Software, the Board and Simple Logic, and Control Logic

1. Introduction

The purpose of this assignment is to introduce you to the software tools and hardware that are used in the labs for this course. The main software tool is the Altera Quartus II Computer Aided Design (CAD) system. You will need to install that software on your computer before you can start this lab. The software will be distributed to you in class.

You will be learning to design hardware that will go into a kind of programmable logic chip call a Field-Programmable Gate Array, or FPGA. The FPGA chip is mounted on a board called the Altera DE2 Development and Education board, pictured below (Fig. 1). This board will be used for the first six assignments and possibly the final project of this course.

The board contains many useful features for learning about logic circuits, including simple input and output mechanisms like switches and lights, and more complicated features like audio and video devices. This assignment will use only the switches and lights that are provided on the bottom edge of the board, as illustrated below, but other assignments will utilize more advanced features. A detailed description of the board can be found on the course website at: http://ee.sharif.edu/~asic

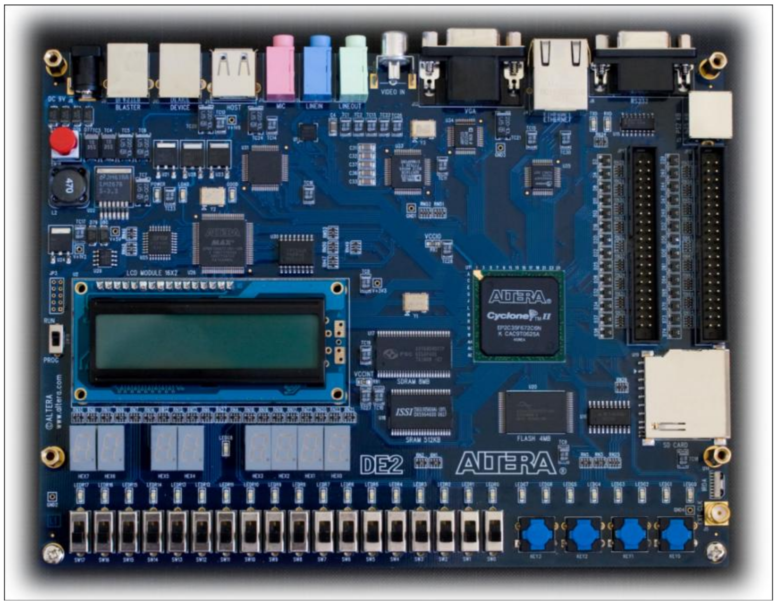


Fig. 1. The Altera DE2 Development and Education board.

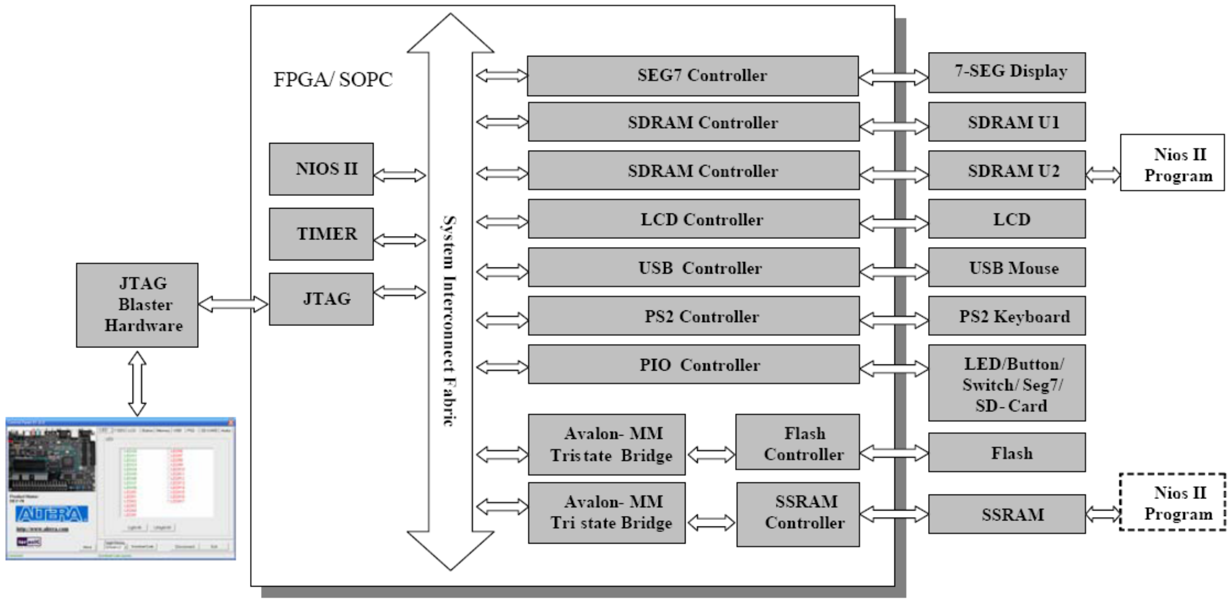


Fig. 2. The block diagram of the DE2-70 Control Panel.

Note:

The blue boxes identified by the word "DE2" are the parts that should be done on-line on the board. The rest can be done at home.

Important Notes:

Note that there are two types of board in the lab (DE2 and DE2-70). So the FPGA and the corresponding pin assignment will be different for them. Please consider the following notes in all of the assignments:

1- **Use 'DE2_pin_assignments.csv' for the pin assignment of the DE2 board and use 'DE2_70_pin_assignments.csv' for the pin assignment of the DE2-70 board.**
2- **A '*starter kit*' is provided for each computer assignment, which can be used for both of the boards without any change. Because the names of the useful pins are changed in the above '.csv' files for both of the boards. Note that always the port names of your design should be the same with the content of the corresponding .csv file, which you added to your design.**
3- **Set the part name in your design as follows:**
   - **For the DE2 board: EP2C35F672C6**
   - **For the DE2-70 board: EP2C70F896C6**

Part I:

Control Panel

The DE2-70 board comes with a Control Panel facility that allows users to access various components on the board from a host computer. The host computer communicates with the board through an USB connection. The facility can be used to verify the functionality of components on the board or be used as a debug tool while developing RTL code. The block diagram of the DE2-70 control panel is shown in Fig. 2. In this section some basic functions of the Control Panel, its structure in block diagram form and its capabilities, will be described.

The DE2-70 Control Panel is based on a NIOS II system running in the Cyclone II FPGA with the SDRAM-U2 or SSRAM (NIOS II is a predefined processor, which you will design it in the other assignments. The details of the concept of NIOS II are not needed in this assignment.). The software part is implemented in C code;

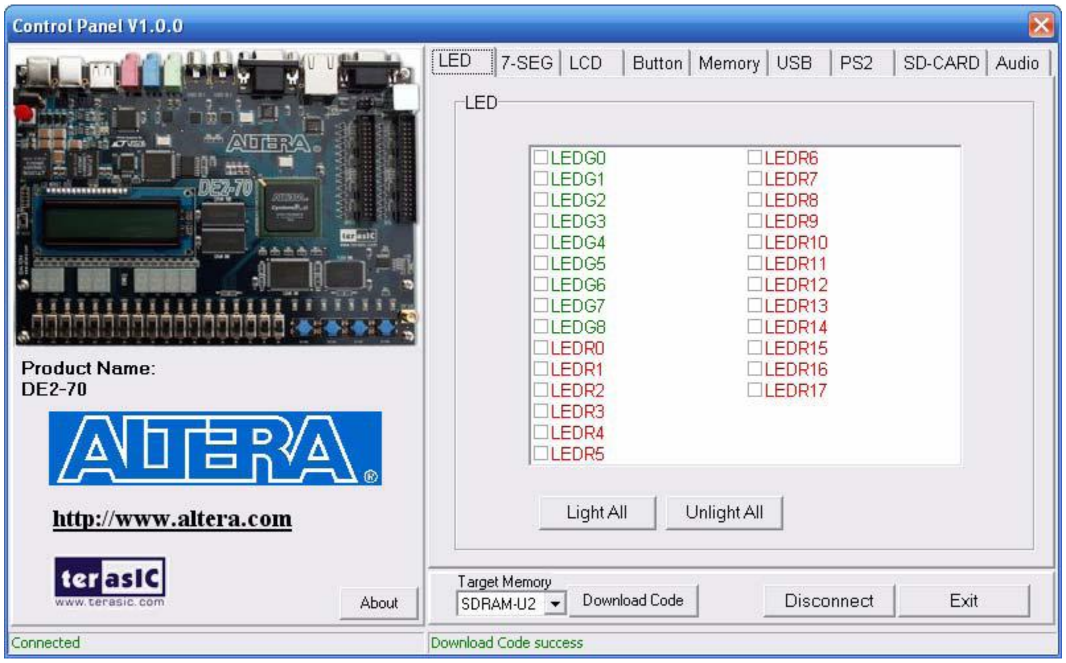


Fig. 3. The DE2-70 Control Panel.

the hardware part is implemented in Verilog code with SOPC builder, which makes it possible for a knowledgeable user to change the functionality of the Control Panel.

Note:

Perform this application only on the **DE2-70** board.

Part I.1. Control Panel Setup

The Control Panel Software Utility is located in the "*DE2_70_control_panel/SW*" folder in the **DE2-70 System CD-ROM**. To install it, just copy the whole folder to your host computer. Launch the control panel by executing the "*DE2_70_Control_Panel.exe*". After the execution you will see the Control Panel in your PC (See Fig. 3).

Specific control codes should be downloaded to your FPGA board before the control panel can request it to perform required tasks. The control codes include one *.sof* file and one *.elf* file. To download the codes, just click the "**Download Code**" button on the program. The program will call Quartus II and Nios II tools to download the control codes to the FPGA board through USB-Blaster[USB-0] connection. The *.sof* file is downloaded to FPGA. The *.elf* file is downloaded to either SDRAM-U2 or SSRAM, according to the user option.

To activate the Control Panel, perform the following steps:

1. Make sure Quartus II and NIOS II are installed successfully on your PC.
2. Connect the supplied USB cable to the USB Blaster port, connect the 12V power supply, and turn the power switch ON.
3. Set the RUN/PROG switch to the RUN position
4. Start the executable *DE2_70_control_panel.exe* on the host computer. The Control Panel user interface shown in Fig. 3 will appear.
5. Select the target memory, **SDRAM-U2** or **SSRAM**, on the control panel. Note. The *.elf* file will be downloaded to the target memory and the memory will be read-only in later memory access operation.

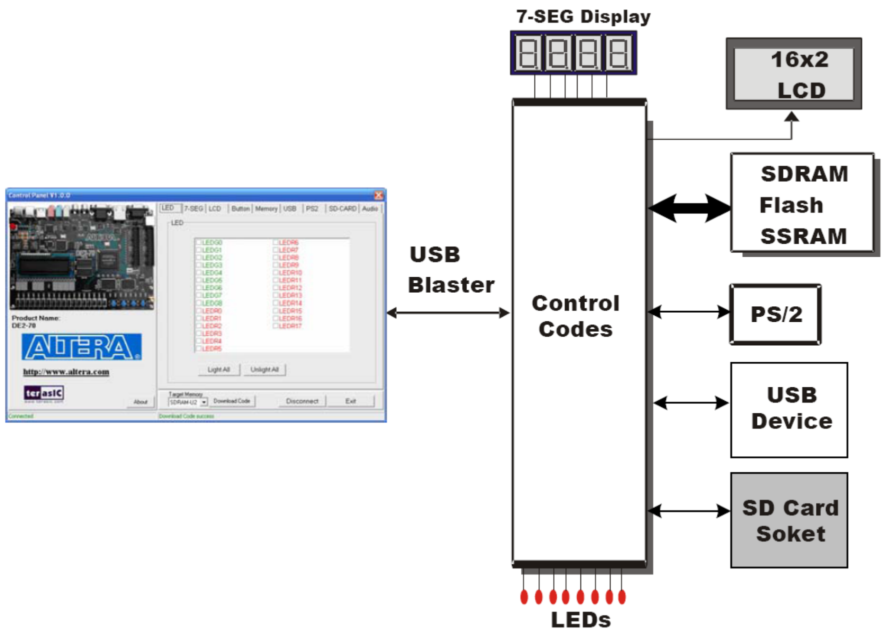


Fig. 4. The concept of the DE2-70 Control Panel.

6. Click **Download Code** button.

Note:

The Control Panel will occupy the USB port until you close that port; you cannot use Quartus II to download a configuration file into the FPGA until you close the USB port.

7. The Control Panel is now ready for use; experiment by setting the value of some LEDs display and observing the result on the DE2-70 board.

Note:

In some cases you will get the error message “**Fail to query board name**”. This problem is that all the directory paths of Quartus and NIOS have changed. Consequently, the software doesn't know where are the basics softwares. In order to use the Control Panel, you must download the .sof and the .elf manually.

So please perform the following steps before running the Control Panel:

1) load the .sof manually using programmer tool (Refer to the tutorial 3).

2) use nios2 shell command :"…:\altera\90\nios2eds\Nios II Command Shell.bat"

3) in this console window, go to the Control panel directory (the path has the form …/cygdrive/<nios drive>/.....

4) Type and execute the following command:

```
nios2-download DE2_70_Control_Panel_ssram.elf -c USB-Blaster[USB-0] -r -g
```

5) The execution tell you that the .elf is loaded

6) Now, execute the control panel software.

8) Enjoy!

The concept of the DE2-**70** Control Panel is illustrated in Fig. 4. The “Control Codes” that performs the control functions is implemented in the FPGA board. It communicates with the Control Panel window, which is active on the host computer, via the USB Blaster link. The graphical interface is used to issue commands to the control codes. It handles all requests and performs data transfers between the computer and the DE2-**70** board.

The DE2**-70** Control Panel can be used to light up LEDs, change the values displayed on 7-segment and LCD displays, monitor buttons/switches status, read/write the SDRAM, SSRAM and Flash Memory, monitor the status of an USB mouse, read data from a PS/2 keyboard, and read SD-CARD specification information. The feature of reading/writing a word or an entire file from/to the Flash Memory allows the user to develop multimedia applications (Flash Audio Player, Flash Picture Viewer) without worrying about how to build a Memory Programmer

Part I.2. Controlling the LEDs, 7-Segment Displays and LCD Display

- A simple function of the Control Panel is to allow setting the values displayed on LEDs, 7-segment displays, and the LCD character display. Choosing the **LED** tab leads to the window in Fig. 3. Here, you can directly turn the individual LEDs on or off by selecting them or click “Light All” or “Unlight All”.

- Choosing the **7-SEG** tab leads to the window in Fig. 3. In the tab sheet, directly use the **Up-Down** control and **Dot** Check box to specified desired patterns, the 7-SEG patterns on the board will be updated immediately.

- Choosing the **LCD** tab leads to the window in Fig. 3. Text can be written to the LCD display by typing it in the LCD box and pressing the **Set** button.

The ability to set arbitrary values into simple display devices is not needed in typical design activities. However, it gives the user a simple mechanism for verifying that these devices are functioning correctly in case a malfunction is suspected. Thus, it can be used for troubleshooting purposes.

Part I.3. Switches and Buttons

Choosing the **Button** tab leads to the window in Fig. 3. The function is designed to monitor the status of switches and buttons in real time and show the status in a graphical user interface. It can be used to verify the functionality of the switches and buttons.
Press the **Start** button to start button/switch status monitoring process, and button caption is changed from **Start** to **Stop**. In the monitoring process, the status of buttons and switches on the board is shown in the GUI window and updated in real time. Press **Stop** to end the monitoring process.
The ability to check the status of button and switch is not needed in typical design activities. However, it provides users a simple mechanism for verifying if the buttons and switches are functioning correctly. Thus, it can be used for troubleshooting purposes.

Part II:

For this part you need to do three important tutorials indicated in the following. Make sure you perform every single step specified in them as they are the foundation of all you will do in the rest of this course. Save them in three subfolders in a folder called Part I to deliver their soft copy to the TA.

Tutorial 1. Do the tutorial called *Using Quartus II CAD Software*, which is in fact the Appendix B of the book Fundamentals of Digital Logic with Verilog Design, 2nd Edition. This tutorial is available on-line on the course website under Assignment #1. This tutorial describes the basics of how Quartus II helps a designer describe circuits and check them for correctness. It shows two ways of creating circuits: either using schematic capture, in which you “draw” a picture of the circuit, or describing a circuit in language form. Please do the same circuit in both ways so that you understand the schematic form and the textual form describing exactly the same thing. We will use the textual form often because it is far more powerful and quicker.

Tutorial 2. Do the first part of the tutorial called *Implementing Circuits in Altera Devices* (Appendix C.1), available on the course website under Assignment #1.

Tutorial 3. Do the tutorial *Physical Implementation in an FPGA* (Appendix D), available on the course website under Assignment #1.

DE2: Download the circuits you designed in the tutorial preparation onto the Altera board and test to see that they work. Remember to set the device to EP2C35F672C6 (the FPGA on the Altera board).

Part III:

Note:
To simplify your work for the other parts, a *starter kit* is provided on course website, located under Assignment #1. The starter kit is a ZIP archive containing a Quartus II project that you will need for each part of the lab. Unzip the archive into a working directory called Assignment_1. If you use Windows Explorer to view the contents of the Assignment #1 directory, you should see the folders shown in Fig. 5

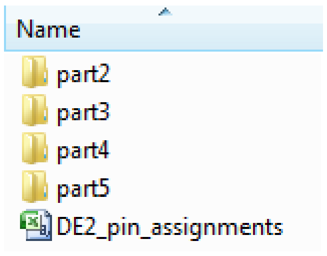

Name
part2
part3
part4
part5
DE2_pin_assignments

Fig. 5. Inside the starter kit for the assignment #1

Create a simple circuit to connect four switches to four lights on the Altera board, by extending the following Verilog code:

```
// Simple module that connects the SW switches to the LEDR lights
module part3 (Switch 1, Switch 2, Switch 3, Switch 4, Light 1, Light 2, Light 3, Light 4);
input Switch 1, Switch 2, Switch 3, Switch 4; // toggle switches
output Light 1, Light 2, Light 3, Light 4; // lights

// Your code goes here

endmodule
```

Fig. 6. Verilog code for Part III.

On the board the FPGA chip that your designs will be programmed into has hardwired connections between the pins of the FPGA chip and the switches and lights on the board. Note that these lights and switches are connected to circuits that allow them to generate 1s and 0s as inputs to your FPGA circuit (for the switches) and to turn on and off in response to digital 1s and 0s that are outputs from your circuit. To use switches and lights you have to tell Quartus II which of the input/output signals in your Verilog code should be connected to which pins on the FPGA chip and which are connected to the switches or lights. The procedure for doing this is called *pin assignment* and was covered in the tutorial 3 of Part I above. Table 1 below indicates which pins (which are referred to by names such as PIN N25) of the FPGA are connected to which switches and lights on the board. There are 18 total switches and lights on the board, but the table only lists 4 of each needed for this part.

NOTE:

Please only connect the SW0…SW3 to LEDR0…LEDR3 on the Altera board .Do NOT try to connect the other switches and lights.

Table 1. Pin assignment table for lights and switches in Part II.

Function	Altera FPGA Cyclone II Pin on the DE2
Switch_1	PIN_N25
Switch_2	PIN_N26
Switch_3	PIN_P25
Switch_4	PIN_AE14
Light_1	PIN_AE23
Light_2	PIN_AF23
Light_3	PIN_AB21
Light_4	PIN_AC22

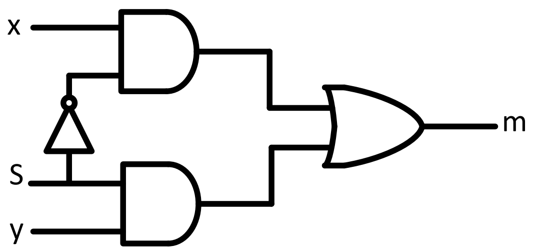
Do the following steps to download and test the circuit in Part II:

1. Open in Quartus II (using the command File > Open Project) the project named *part3.qpf* in the *part3* subdirectory to begin your work.
2. Create a Verilog module named *part3* for the code in Fig. 3 and include it in your project. Make sure to complete the code by adding the assignment statements for the lights.
3. Use Quartus II to make the pin assignments shown in Table 1 as described in tutorial 3. Compile the project.

DE2: Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by flipping the switches and observing the lights.

Part IV:

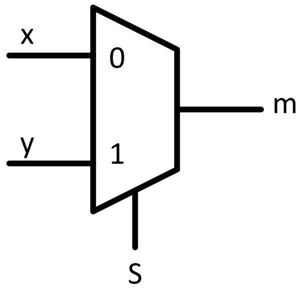
Fig. 7a shows a sum-of-products circuit that implements a 2-to-1 multiplexer (MUX) of inputs x and y select input s and output m. If s = 0 the multiplexer's output m is equal to the input x, and if s = 1 the output is equal to y. Part b of the figure gives a truth table for this multiplexer, and part c shows the schematic circuit symbol.



a) Circuit

S	m
0	x
1	y

b) Truth Table



c) Symbol

Fig. 7. A 2-to-1 Multiplexer.

The multiplexer can be described by the following Verilog statement:

```
assign m = (~s & x) | (s & y);
```

You are to design a circuit, using Verilog, which is a more complex version of a multiplexer. Rather than select between two signals, your circuit is to select between two sets of *eight* signals, as illustrated in Fig. 8a. 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 Fig. 6b, in which X, Y , and M are depicted as eight-bit wires.

Do the following steps to download and test the circuit in Part IV:

1. Open the project named *part4.qpf* in the *part4* subdirectory to begin your work.
2. Include your Verilog file for the eight-bit wide 2-to-1 multiplexer in your project. Use switch SW17 on the board as the s input, switches SW7−0 as the X input and SW15−8 as the Y input. Connect the output M to the green lights on the board, called LEDG7−0.

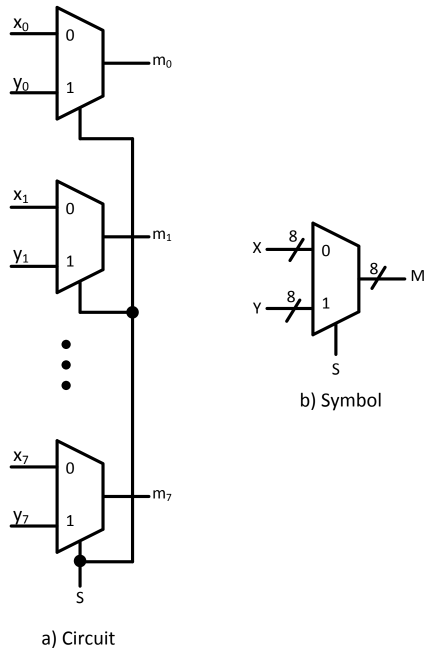


Fig. 8. An eight-bit wide 2-to-1 multiplexer.

3. Include in your project the required pin assignments for the board using the pin assignment file as described above. As discussed in Parts II and III, these assignments ensure that the inputs declared in your Verilog code will use the pins on the Cyclone II FPGA that are connected to the *SW* switches and LEDRs, and the outputs of your Verilog code will use the FPGA pins connected to the LEDG lights. Compile the project.

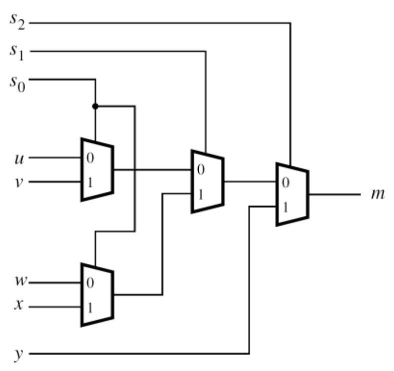
NOTE:

Always Connect the inputs to the LEDRs in Part IV and Part V.

DE2: Download the compiled circuit into the FPGA chip. Test the functionality of the eight-bit wide 2-to-1 multiplexer by toggling the switches and observing the LEDs.

Part V:

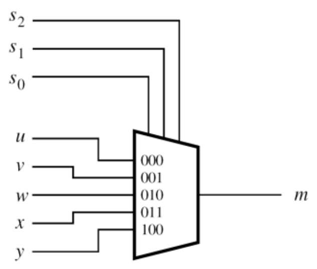
In Fig. 7 we showed a 2-to-1 multiplexer that selects between the two inputs *x* and *y*. For this part consider a circuit in which the output *m* has to be selected from *five* inputs u, v, w, x, and y. Part a of Fig. 9 shows how we can build the required 5-to-1 multiplexer by using four 2-to-1 multiplexers. The circuit uses a 3-bit select input {s2,s1,s0} and implements the truth table shown in Fig. 9b. A circuit symbol for this multiplexer is given in Fig. 9c. Recall from Fig. 8 that an eight-bit wide 2-to-1 multiplexer can be built by using eight 2-to-1 multiplexers. Fig. 10 applies this concept to define a three-bit wide 5-to-1 multiplexer. It contains three instances of the 5-to-1multiplexer circuit in Fig. 9.



a) Circuit

s_2 s_1 s_0	m
0 0 0	u
0 0 1	v
0 1 0	w
0 1 1	x
1 0 0	y
1 0 1	y
1 1 0	y
1 1 1	y

b) Truth table



c) Symbol

Fig. 9. A 5-to-1 MUX.

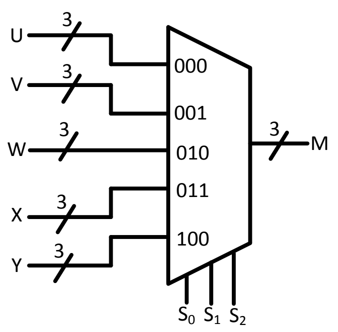


Fig. 10. A three-bit wide 5-to-1 multiplexer.

Do the following steps to download and test the circuit in Part IV:

1. Open the project named *part5.qpf* in the *part5* subdirectory to begin your work.
2. Create a Verilog module for the three-bit wide 5-to-1 multiplexer. Connect its select inputs to switches SW17−15, and use the remaining 15 switches on the Altera board (SW14−0) to provide the five 3-bit inputs U through Y. Connect the output M to the green lights *LEDG*2−0.
3. Include in your project the required pin assignments for the board. Compile the project.

DE2: Download the compiled circuit into the FPGA chip. 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 U to Y can be properly selected as the output M.

Part VI - Design of a 7-Segment Decoder Circuit

Fig. 11 shows a 7-segment character display controlled by a logic circuit. These displays are common on digital watches and various electrical devices. This circuit is called a 7-segment *decoder* and it has a three-bit input {c_2, c_1, c_0} (which you will connect to switches on the board), and 7 outputs that turn on or off the 7 different segments in the character display. The three inputs present a code that are translated by the circuit into 7 outputs to create a particular character by lighting up some of the lights on the display.

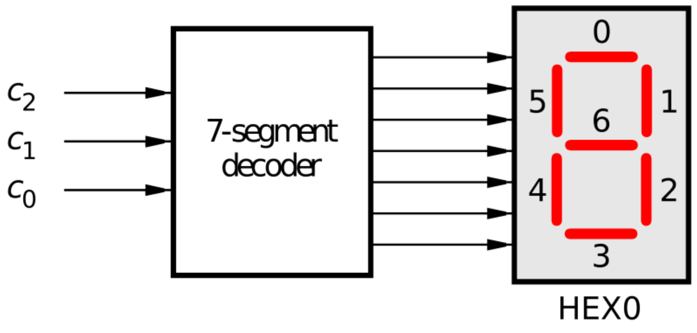


Fig. 11. A 7-segment decoder.

You are to design a circuit that is able to decode and display *the last 5 digits of your student number*. An example mapping is given in Table 2, which lists the characters that should be displayed for each value of {c_2, c_1, c_0}. To keep the design simple, you only need to decode five numbers, represented using the codes {c_2, c_1, c_0} = 000, 001, 010, 011, and 100. The fifth code should produce a blank character with all of the lights off. Since we don't care what is displayed for the remaining code values (110 and 111) you can be treat these as don't care values. The seven segments in the display are identified by the numbers 0 to 6 shown in Fig. 11. Each segment is lit up by driving it to the logic value 0 (which is a little opposite of what you might expect). You must implement a separate logic function that controls each segment in the display. Use a Karnaugh map to determine the minimal (optimal) sum-of-products expressions for each of these 7 outputs.

Create a Verilog module for your 7-segment decoder circuit. In your Verilog code, use only simple assign statements to specify each of the sum-of-products expressions generated from your Karnaugh maps. In your Verilog code assign the {c_2, c_1, c_0} inputs to switches SW2–0 on the board, and assign the outputs of the decoder to the HEX0 display on the board. The segments in this display are called HEX00, HEX01, . . ., HEX06, corresponding to Fig. 11 using the pin assignment file provided in the previous assignment.

Note that in the pin assignments file the HEX0 segments are declared as an array. To use the same names in your Verilog code, your should declare the seven-bit output port as

```
output [0:6] HEX0;
```

By using this 7-bit array, the names of the seven segments in your code will match the names that are used for these segments in the pin assignment file.

Table 2. Character codes (for the case where your student number ends in ’12345’)

$c_2c_1c_0$	Character
000	1 (replace with last digit of your student number)
001	2 (replace with 2nd last digit of your student number)
010	3 (replace with 3rd last digit of your student number)
011	4 (replace with 4th last digit of your student number)
100	5 (replace with 5th last digit of your student number)
101	‘blank’
110	‘Don’t Care’
111	‘Don’t Care’

DE2: Compile and download your circuit onto the board. Test the functionality of the circuit by toggling the SW2−0 switches and observing the 7-segment display HEX0.