|  |  |  |
| --- | --- | --- |
| **American University of Sharjah**  **College of Engineering**  Dept of Computer Science & Engg  P. O. Box 26666  Sharjah, UAE | A picture containing drawing  Description automatically generated | **Lab Instructor:** Mr. Mohmmed Elnawawy  **Office:** EB2-001  **Phone**: 971-6-5152975  **e-mail**: [melnawawy@aus.edu](mailto:melnawawy@aus.edu)  **Semester**: Fall 2020 |

Lab 2

Input/Output: Using Switches and 7-Segment Display on the FPGA Board

Objectives:

The purpose of this exercise is to learn how to use the basic Input and output devices on DE2-115, and implement a circuit with several heriarchical modules that uses these devices. We will use the switches SW17-0 on the DE2-115 board as inputs to the circuit. We will use light emitting diodes (LEDs) and 7-segment displays as output devices.

Required Equipment and Tools:

* Quartus® Prime Lite Edition

INTRODUCTION:

Driving LEDs by Switches

DE2-115 is provided with 18 switches, labeled *SW*[17] to *SW*[0], 18 red LEDs, labeled LEDR[17] to LEDR[0], and 8 Green LEDs, named LEDG[7] to LEDG[0].

The switches can be used for input, and the LEDs can be used as output devices. **Code 1** is an example to show you how to connect the switches to the LEDs using Verilog **assign** statement.

**Code 1**

// Simple module for DE2-115 that connects the SW switches to the LEDR lights

**module** code\_1 **(**SW**,** LEDR**);**

**input** **wire** **[**17**:**0**]** SW**;** // slide switches

**output** **wire** **[**17**:**0**]** LEDR**;** // red LEDs

// Concurrent assignment, the LEDs [17:0] will always take the value of

// Switches [17:0] accordingly

**assign** LEDR **=** SW**;**

//This is equivalent to the following Vector bit select

//assign LEDR[17]= SW[17];

//...

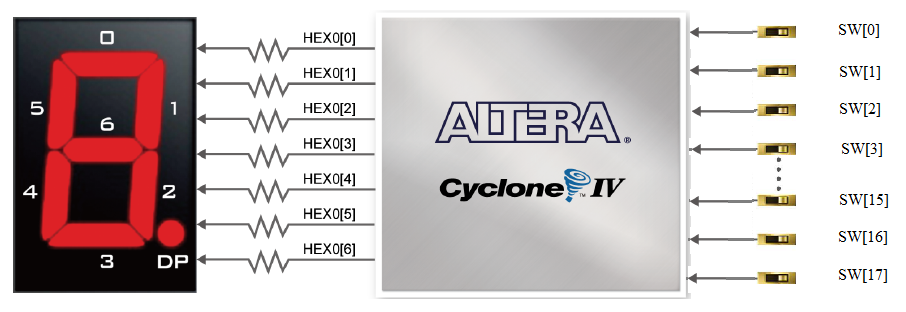
//assign LEDR[1] = SW[1];

//assign LEDR[0] = SW[0];

**endmodule**

Displaying numbers in hexadecimal

The DE2-115 Board has eight 7-segment displays HEX0 to HEX7. These displays are arranged into two pairs and a group of four. As indicated in the schematic in Figure 1, the seven segments (common anode) are connected to pins on Cyclone IV E FPGA. Applying a **LOW** logic level to a segment will light it up and applying a **HIGH** logic level turns it OFF.

 Figure 1 Connections between the 7-segment display HEX0 and Cyclone IV E FPGA

Each segment in a display is identified by an index from 0 to 6, with the positions given in the figure below. i.e. HEX1[0] controls segment 0 in display 1. The name of each display is printed beneath it on the board.

Hexadecimal (base 16) is a common way to represent numbers in digital circuits. Figure 2 shows the relationship between a value in decimal and its hexadecimal counterpart.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Value | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Representation in hex | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |

**Code 2** is an example of using the 7-seg display with **assign** statement, it shows how to display the character F on display HEX0. F

**Code 2**

// Simple module for DE2-115 that displays F on display HEX0

**module** code\_2 **(**HEX0**);**

**output** **wire** **[**6**:**0**]** HEX0**;** // 7 segments display HEX0

**assign** HEX0 **=** 7'b0001110**;** // 0 turns ON the segment

//| |

//MSB...LSB 1 turns OFF the segment

**endmodule**

**Code3** shows a part of a hexadecimal to 7 segments decoder, using **case** statement and behavioural modelling. It converts a 4 bit hexadecimal input into its corresponding digit to be displayed on a 7-seg display.

**Code 3**

//Hexadecimal to 7 segments decoder module. (Incomplete)

**module** hex7seg **(**in\_binary**,** out\_7\_seg**);**

**input** **wire** **[**3**:**0**]** in\_binary**;** //

**output** **reg** **[**6**:**0**]** out\_7\_seg**;** //

**always** **@\***

//always @\* means "continuously

//drive the output out based on

//the value computed in this block".

**case** **(**in\_binary**)** //

4'h0**:** **begin** out\_7\_seg **=** 7'b1000000**;** **end** //

4'h1**:** **begin** out\_7\_seg **=** 7'b1111001**;** **end** //

/\* Put your code here\*/

**default:** out\_7\_seg **=** 7'b1111111**;**

//Important: Do not forget default case; Bad thing will happen!

**endcase**

**endmodule** //

Combining Multiple Modules in Verilog

Modules can be instantiated from within other modules. When a module is instantiated, connections to the ports of the module must be specified. There are two ways to make port connections. One is connection by name, in which variables connected to each of module inputs or outputs are specified in a set of parenthesis following the name of the ports. In this method order of connections is not significant. See **Code 4**.

**Code 4**

//Simple module for DE2-115 that displays hexadecimal numbers on display HEX0

**module** code\_4 **(**SW**,** HEX0**);**

**input** **wire** **[**17**:**0**]**SW**;**

**output** **wire** **[**6**:**0**]** HEX0**;** // 7 segments display HEX0

// connection by name

hex7seg u1 **(.**in\_binary **(**SW**[**3**:**0**]),.** out\_7\_seg **(**HEX0**));**

// module\_name Instance\_name (.port\_name(signal),.port\_name(signal));

**endmodule**

**Code 5** is an example of the second method, called ordered connection. In this method the order of the ports must match the order appearing in the instantiated module.

**Code 5**

//Simple module for DE2-115 that displays hexadecimal numbers on display HEX0

**module** code\_5 **(**HEX0**);**

**input** **wire** **[**17**:**0**]**SW**;**

**output** **wire** **[**6**:**0**]** HEX0**;** // 7 segments display HEX0

// ordered connection

hex7seg u1 **(**SW**[**3**:**0**],** HEX0**);**

// module\_name instance\_name (signal, signal);

**endmodule**

In-lab Activity

Complete the following tasks, make sure to heavily comment your code.:

1. Modify *hex7seg.v* from Code 3, to accomplish the following: the module will read any 4-bit input word and produce the 7-bit output word that represents the binary input on the 7-segments display. All values from 0000 to 1111 should be discussed in the **case** statement. Simulate your module using the waveform editor.
2. Create a *top.v* file that contains the top-level module *top*. This module should read the values of SW[3:0] and SW[7:4] then display their binary values on LEDs LEDR[3:0] and LEDR[7:4]. The hexadecimal value of the switches should be displayed on HEX0 and HEX1 respectively. Use the module *hex7seg.v* created earlier as a hexadecimal to 7-segments decoder. Use the Connection by name method in Code 4 to connect your modules in the top level module. Make sure to heavily comment your code. Figure 3 shows the functional diagram of your circuit.

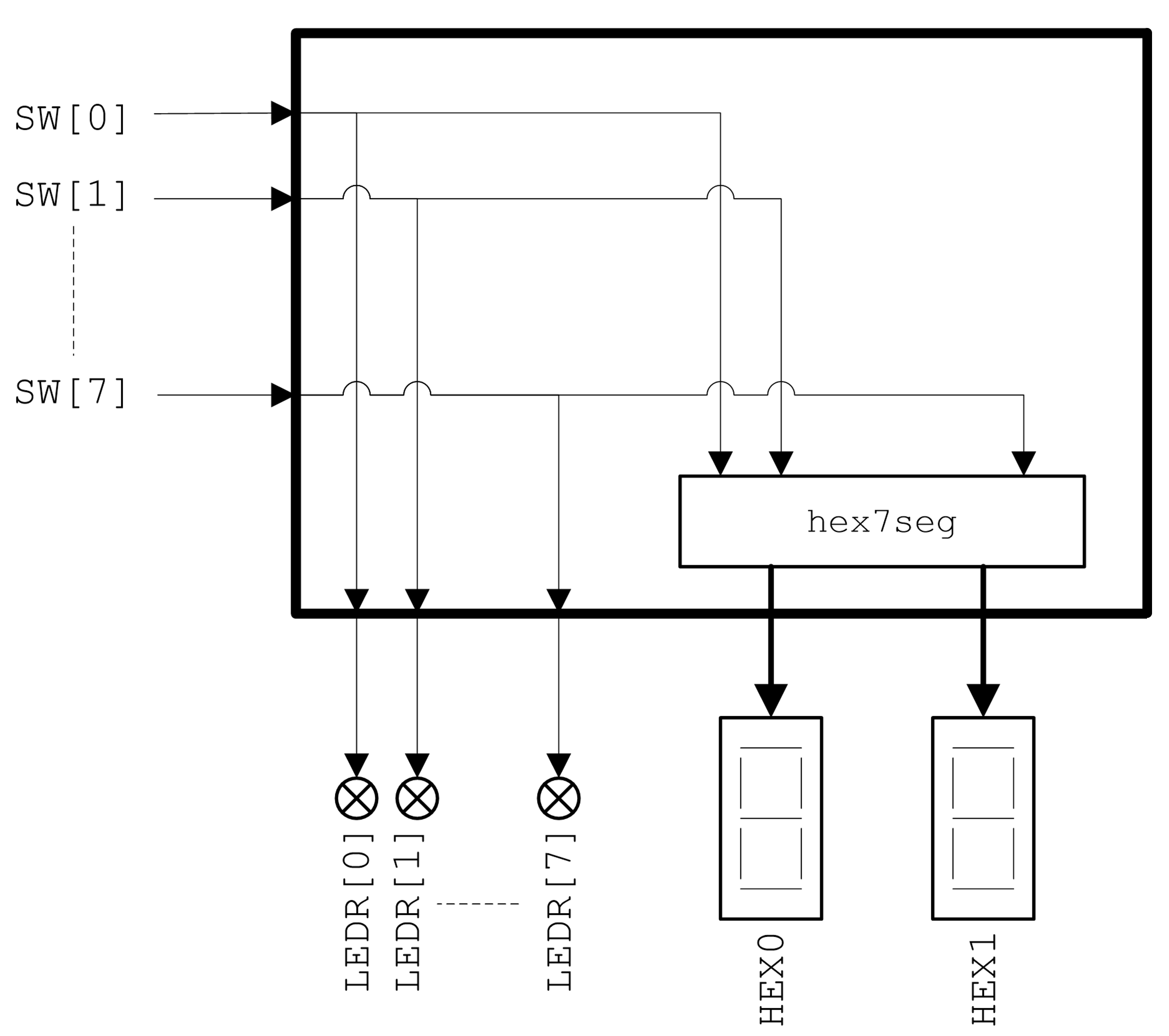


Figure 3. Functional Block Diagram of the final circuit