Lab #6 Student ID scroll using Seven Segment Display
Revised 2_21_11

Overview

LED sign is now very common in the street. It is an effective method for providing information so it is well suit to advertisements because the scrolling message draws the attention better than the normal advertisement panel. LED sign is made up of many small LEDs (light emitting diodes), and small digital device chip control what message to display. In this lab, we are going to display the student ID on the seven segment displays (SSDs) of the BaSYS board which is also made up of seven bar type LEDs. With 4 SSDs, we will display 9 numbers (including (-) dash signs). You will learn how to control the seven segment displays on the BaSYS board. Also, you will learn to use the multiplexer to choose which of the seven-segments to turn on at one time.

Before beginning this module, you should...

- Be familiar with reading and constructing basic logic circuits;
- Understand logic equations and how to implement a logic circuit from a logic equation given in minterm notation, decimal format;

After completing this module, you should...

- Be able to input a combinational circuit using the Xilinx ISE schematic editor;
- Be able to examine the output of a logic simulator to verify whether a given circuit has been designed correctly.
- Be able to program the BaSYS board using the USB programmer cable, given a constraints file.

This module requires:

- A Windows PC
- The Xilinx ISE/WebPack software
- A Digilent BaSYS board
Lab #6 Hexadecimal to Seven-Segment Display Decoder

Background

Light Emitting Diode (LED)
An LED emits light when current flows through it in the positive direction. Typically, if the voltage difference between the positive end of the LED (the anode) and the negative end of the LED (the cathode) is greater than ~1.7 volts, the LED will turn on. The actual amount of voltage needed will depend on the specific model of LED used. A current limiting resistor is usually used with LEDs to prevent too much current from destroying the LED.

When using an LED with a digital device, there are two types of connections that are typically used. The negative end of the LED (the cathode) can be connected to ground and the positive end of the LED is connected through a resistor to the output of the device. When a positive output occurs at the output of the digital device, the LED is turned on. The LEDs on the BaSYS board are high true logic - LED anodes are driven from the FPGA via current-limiting resistors, so they will illuminate when a logic ‘1’ is written to the corresponding FPGA pin. Alternatively, the LED anodes can be connected to VDD (the supply voltage) via a current limiting resistor, and the cathode of the LED can be connected to the output of the digital device. When the digital device provides a low signal, the LED is turned on. This is commonly called a low true output or the LED is said to be floating high.

Seven Segment Displays of BaSYS board

With seven-segment displays, there are seven LEDs and seven resistors in one package. Note on the diagram below (from the BaSYS datasheet) that each of the seven segment displays also has a decimal point LED available.
There are two types of seven-segment displays that are typically available (shown below). One type has a common cathode (all of the cathodes of the seven LEDs are all tied together), the other type has a common anode (all of the anode ends of the LEDs are tied together). The type on the BaSYS board is a common anode which means that all of the anodes of the LEDs are all connected together. Each of the cathodes is available to apply a ground voltage to turn on. The seven-segment displays on the BaSYS board require are low true logic to drive them.

The BaSYS board has four common-anode seven-segment displays. All of the anodes are tied together and connected through a pnp transistor to +3.3V. There is not an I/O pin to this connection – the value is fixed. An FPGA I/O pin is dedicated to each of the cathodes, a-g, plus the decimal point. Each of the LEDs also is connected through a 100Ω resistor as part of the connection. If we look at the diagram above for the common anode connection, since the anode portion of the circuit is always connected to V_DD, the cathode side needs to be connected to V_SS to light up the LED. The bottom line is an output of 0 from the FPGA pin will turn on a segment, and an output of 1 will turn the device off since the cathode value will not be less than the anode value across the LED.

Another important fact is that only 1 set of cathodes are used for 4 SSDs so data is passed to all SSD at the same time. It only shows the same number on 4 SSDs. In order to put different value on each SSD, the common anode signals are available as four “digit enable” input signals to the 4-digit display. The cathodes of similar segments on all four displays are connected into seven circuit nodes labeled CA through CG (so, for example, the four “D” cathodes from the four digits are grouped together into a single circuit node called “CD”). These seven cathode signals are available as inputs to the 4-digit display. This signal connection scheme creates a multiplexed display, where the cathode signals are common to all digits but they can only illuminate the segments of the digit whose corresponding anode signal is asserted. Therefore, the different data can be shown on each SSD by ‘scanning’ SSD with anode control signal which is sampled from the machine clock.

In this lab, we will learn how to represent hexadecimal number on SSD, and how to determine the position of SSD to be lit.
Lab Introduction

As described in the overview, we are going to build the student ID scroll with 4 SSDs. We start with the basic but essential blocks, **hexadecimal to seven segment converter** and **SSD selector**. Hex number to seven segment convertor is a function which turns on the LEDs of SSD. SSD selector is a function to choose which SSD to be turned on. There are three signals to control 4 SSDs: A_TO_G, AN, and DP in BaSYS UCF file. A_TO_G determines what LEDs (a to g in diagram below). AN signal determines which SSD (we have four), or how many SSDs, should be turned on. Finally, DP determine to turn on/off decimal point (normally used in calculator, but we don’t use here). All three control signals must be specified in the module properly. Here, two modules, hexadecimal to seven segment converter and SSD selector, helps to determine the control signal, A_TO_G and AN.

**Hexadecimal to Seven Segment converter**

The table below shows the logic level values needed for each cathode pin to turn on the appropriate LEDs to create the hex values 0-F. See the diagrams below to determine which LEDs must be turned on for each value. A_TO_G [6:0] is the output names for the seven segment displays for this module. Make sure to turn on the LEDs, the value for A_TO_G must be 0, and to turn off, 1. For example, to make 0(zero) on SSD, a, b, c, d, e, f should be turned on. So the values of A_TO_G for Hex number 0 is 0000001. Complete the worksheet below before proceeding to Verilog implementation.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>0</td>
<td>0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lab #6 Hexadecimal to Seven-Segment Display Decoder

SSD selector
As seen in Seven Segment Displays of BaSYS board section, all SSDs have common cathodes, which means converted signal is applied on all SSDs. To

To control the 7-segment display, we can use ‘an’ signal. One simple way to control is using four other switches to control on/off on the each 7-segment display, for example, assign an[0]= ¬sw[5] in Verilog code. However, if we want to select which 7-segment display is turned on, we can use multiplexer with two other switches.

<table>
<thead>
<tr>
<th>BTN[1]</th>
<th>BTN[0]</th>
<th>an[3]-1st SSD</th>
<th>an[2]-2nd SSD</th>
<th>an[1]-3rd SSD</th>
<th>an[0]-4th SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
</tbody>
</table>

Corresponding signals for 7-segment display in Basys board is shown on the following table.

<table>
<thead>
<tr>
<th>BTN[1]</th>
<th>BTN[0]</th>
<th>an[3]-1st SSD</th>
<th>an[2]-2nd SSD</th>
<th>an[1]-3rd SSD</th>
<th>an[0]-4th SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

We can make circuit by using K-map for an[3], an[2], an[1], and an[0]. Complete the K-map in the lab report.
Lab #6 Hexadecimal to Seven-Segment Display Decoder

Circuit Design (Verilog)

**Specification (Three modules)**

- **hex7segment**: generates control signal to turn on/off 7 segment LEDs
  - Inputs: HEX [3:0] – 4 bit width wire (can represent one hexadecimal number)
  - Output: SEG[6:0] – 7 bit width register
- **SSD_selector**: sets which SSD, or how many SSD, should be lit
  - Inputs: BTN [1:0] – 2 Buttons on BaSYS board will be used
  - Output: AN [3:0] – 4 bit width wire (to choose which SSD to be lit)
    - Ex) 0000 – turn all SSDs on, 1110 – turn the fourth SSD on
- **Hex7segment_top**: TOP module that calls two functions above inside the module. This module tests the SSD using 4 switches for the hexadecimal value and 2 buttons for SSD control.
  - Inputs: SW [3:0], BTN [1:0]
  - Outputs: A_TO_G [6:0], AN[3:0], DP (Decimal Point)

**hex7Segment**

1. Create a new project, i.e., Lab6 and select the top level source as HDL
2. Create a new source file, i.e. hex7segment, and choose Verilog module.
3. Define port name like following. Be careful on the direction.
   - a. Inputs: HEX [3:0] (This is a bus where MSB is 3, and LSB is 0)
   - b. Output: SEG[6:0]
4. Add ‘BaSYS2_ECE3714.ucf’ file at the ‘Add Existing Sources’ step.
5. Write a Verilog functions
   - a. Write functions inside module and endmodule.
   - b. Do not write module, endmodule, input, and output again.

Example code – hex7segment>

```
module hex7segment(HEX,SEG);
  input [3:0] HEX;
  output [6:0] SEG;

  reg SEG;           // declare SEG as register (required in case statement)
  always @(*)
    case (HEX)       // make a case statement from 0 to F
      0: SEG = 7'b0000001;
      1: SEG =
      // ... write for 2~9
      'hA: SEG =
      'hB: SEG =
      'hC: SEG =
      'hD: SEG =
      'hE: SEG =
      'hF: SEG =
      default: SEG = 7'b0000001; // 0
    endcase
  endmodule
```
1. Click ‘Create New Source’ on Process tab.
2. Create a new source file, i.e. SS_selector, and choose Verilog module.
3. Define port name like following. Be careful on the direction.
   a. Inputs: BTN [1:0]
   b. Output: AN [3:0]
4. Write a Verilog
   a. Write functions inside module and endmodule.
   b. Use gate primitive function. Don’t use continuous statement. (That’s not been covered yet)
   c. Hint: it should have simple 4 OR function.
   d. Ex) or(AN[3], ~BTN[1], ~BTN[0])
   e. ~ is simplified inverting operator ( same as not(BTN1_not, BTN[1])

hex7segment_top (Hierarchical Design: Top module)

1. Click ‘Create New Source’ on Process tab.
2. Create ‘hex7segment_top’ Verilog file.
3. Define input and output port list like following
   a. Inputs: SW [3:0], BTN [1:0]
   b. Outputs: A_TO_G [6:0], AN[3:0], DP (Decimal Point)
4. Right click the ‘hex7segment_top’ on the Source tab, and choose the ‘Set As Top Module’ like following.
   a. (NOTE‼) In this way, you can designate which module is the major module which calls other sub modules.
5. Write a Verilog
   a. Call two module above inside the module
   b. Form: <module_name> <instance_name> (variable1, variable2,...)
   c. Ex) my_function M1 (SW, BTN, CLK)
   d. Add one statement like the following figure

Contains material © Digilent, Inc.
Lab #6 Hexadecimal to Seven-Segment Display Decoder

Example code – hex7segment_top

```verbatim
module hex7segment_top(SW,BTN, A_TO_G, AN, DP);
  input [3:0] SW;
  input [1:0] BTN;
  output [6:0] A_TO_G;
  output [3:0] AN;
  output DP;

  assign DP = 1;  // off decimal point

endmodule
```

// call two functions here

endmodule

Circuit Simulation for Verification

You don’t have to make a test bench waveform for this project. Actual board testing is enough.

Configuring the FPGA and Test

1. Make sure two thing before compiling the modules
   a. Make sure you have a right constraint file (BaSYS2_ECE3714.ucf or BaSYS_ECE3714.ucf).
   b. Right Click on ‘Implementation’ on Process tab, and go to property menu. You will see the
      figure like following. Check ‘Allow Unmatched LOC constraints’ option. This option tells the
      hardware that there are some pins that we do not use so as to ignore it.

2. Compile the program by double clicking on ‘Synthesize-XST’, ‘Implement Design’, and ‘Generate Programming File’
   menu on the Process tab.
3. You can check Errors on Transcript window of the Xilinx ISE program. If you have error, check your Schematic design
   and two steps above.
4. If there is no major error, you can find the ‘bit’ file on your working folder. The name follows you project name, i.e.,
   hex7segment_top.bit. Check it out.
5. Check the number yourself from 0°F by using switches.
6. Check the position of the SSD yourself by using buttons.
7. If there are some error, check the appropriate module and fix the error.
Lab #6 Hexadecimal to Seven-Segment Display Decoder

Homework (required): Putting All Together for Student ID scroll

Now we can make the student ID scroll. Here is the diagram for Student ID scroll module. Suppose we have a student ID, 947-012-123. We want to make it scroll from right to left. The student ID scroll module consists of ‘clock_divider’, ‘shift_array’, and ‘x7segment.’ First, this module passes 4 characters (Hex value) from the stored the student ID to x7segment module which converts 4 characters to SSD signals. In this module, an important signal is CLK (clock). We need a 2 bit width CLK signal to select the data (out of 4 characters) and the SSD out of 4 SSDs. Red colored ‘SELECT’ signal is sampled from the CLK (machine clock, 50 MHz), and it is passed to data selector modules and SSD selector module. For example, if the Select is 00 or 0, 4th data and 4th SSD is chosen, and if the Select is 01 or 1, 3rd data and 3rd SSD is chosen.

1. Download ‘lab6.zip’ from the lab webpage (Lab files folder)
2. Unzip files and add all files to the project.
3. Set ‘scroll_top’ module as a top module
4. Change only three files
   a. Change the default ID with your student ID in ‘scroll_top’ module
   b. Insert two module calls (Hex7segment, SSD selector) like we’ve done above in ‘x7segment’ module. Make sure you put right inputs and outputs in module call
   c. Modify the ‘hex7segment’ to process ‘-’ (dash) and ‘ ‘ (blank). We can use the hexadecimal number ‘D’ as a signal for dash and the number ‘F’ as blank.
      • ‘hD: SEG = // - dash
      • ‘hF: SEG = // blank
5. Bring the result to the next lab, we TA will check the results (10 points out of 35 points)
   a. You can talk about the problem each other, but do not show your results or copy others’ results.
   b. Try to complete yourself. If it seems too hard, ask TA.