Lab 7 – 7-Segment Display

CS 2052 Computer Architecture

Dept. of Computer Science and Engineering, University of Moratuwa

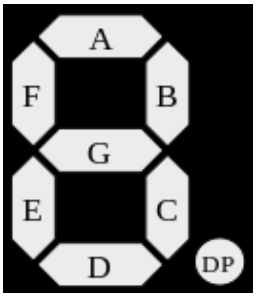
Learning Outcomes

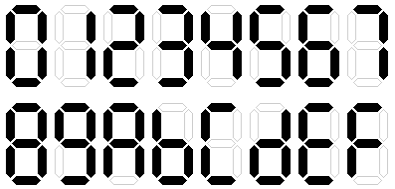
In this lab, we will design a 7-segment display for our 4-bit Arithmetic Unit developed in previous lab. After completing the lab, you will be able to:

- design and develop a lookup table using Read Only Memory (ROM)
- design and develop a 7-segment display using the output from the lookup table
- verify their functionality via simulation and on the development board

Introduction

7-segment display is a form of electronic device for displaying decimal numerals. It is an alternative to the more complex displays like dot matrix displays in LCD and LED screens. BASYS2 has 4, 7-segment displays. Following figure shows the 7 segments, the decimal point on a 7-segment display, and a possible representation of hexagonal numbers (numbers between 0 and 15).





Depending on the number we want to display, we can switch on or off the desired segment(s). Which segment you may want to switch on or off also depends on the circuit where it may light up depending on an active high (common cathode circuit) or active low input (common anode circuit).

In this lab, we will display the output of our 4-bit arithmetic unit (developed in Lab 6) as a hexadecimal number using a 7-segment display. Depending on the 2 input numbers, the arithmetic unit will produce a 4-bit sum, a carry, and zero flag. While Lab 3 and 6 indicated the resulting sum using a set of LEDs, in this lab we will use a 7-segment display to show the output of 4-bit sum from the RCA as a hexadecimal number. Therefore, we need to decide which segments to light up depending on the sum produced by the RCA. One way to do this is to write logic equations for inputs to each of the segments (using k-maps). Instead, we will use a lookup

table to map the 4-bit sum to the 7 segments on the display. Such a lookup table can be built using a ROM.

Building the Circuits

Step 1: Using the BASYS2 Reference Manual find out whether the BASYS2 board uses a common cathode circuit or a common anode circuit.

Complete the following table to find out the mapping between the 4-bit sum and the corresponding 7-segment code. Some lines are filled as examples.

Output from RCA					**Segments to Switch On**						
S_3	**S_2**	**S_1**	**S_0**	**Hex. Value**	**A**	**B**	**C**	**D**	**E**	**F**	**G**
0	0	0	0	0	*0*	*0*	*0*	*0*	*0*	*0*	*1*
0	0	0	1	1	*1*	*0*	*0*	*1*	*1*	*1*	*1*
0	0	1	0	2							
0	0	1	1	3							
0	1	0	0	4							
0	1	0	1	5							
0	1	1	0	6							
0	1	1	1	7							
1	0	0	0	8							
1	0	0	1	9							
1	0	1	0	A	*0*	*0*	*0*	*1*	*0*	*0*	*0*
1	0	1	1	B							
1	1	0	0	C							
1	1	0	1	D							
1	1	1	0	E							
1	1	1	1	F							
Hexadecimal value of each column Note: 0000 refers to LSB and 1111 refers to MSB					2812						

Step 2: Building 4-bit arithmetic unit symbol.

Create a new project in Xilinx ISE Design Suite and name it as **Lab 7**.

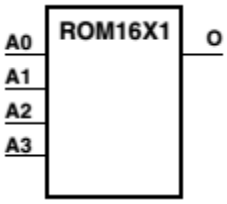
Import relevant schematics from previous labs. You may either copy .sch files manually to the new project or use **Add Copy of Source...** option from the pop-up menu on Sources window.

Create a new symbol and name it as **AU**.

Step 3: Building Lookup Table.

A lookup table is used to find the mapping between inputs S_3 - S_0 and outputs A - G in the above table. As the mapping will not change with time, we can save it in a ROM. However, being a simple board, BASYS2-based schematic development only supports ROMs with 1-bit output. As we have 16 possible input combinations and 7 outputs (i.e., one for each segment), we need 7, 16×1 ROMs (look for ROM16×1 symbol, see the datasheet on Moodle for more details).

Following is the schematic representation of ROM16×1 in ISE.



A 16x1 ROM stores a 16-bit number. It takes a 4-bit address as an input. Depending on the address value, it then produces an output based on which bit is referred by the address. For example, suppose we stored the value $2A1B_{16}$ (i.e., 0x 2A1B) in the ROM:

2A1B = 0010 1010 0001 1011

Then, if we give the address 0000, it will give 1 as the output as the LSB is 1. If we set 0001 as the address, then the output is also 1. If the address is 1000, then the output is 0. If the address is 1111, then the output is 0.

To build a ROM that is capable of keeping track of the mapping between the 4-bit sum from RCA and the desired inputs to be given to the 7-segment display, we need 7 ROM16×1s. For each ROM16×1, we also need to set its initialization value (recall that ROMs are typically programmed during implementation). Hence, find out the hexadecimal representation for each of the columns in the above table. Each value should be represented as a 4-digit hexadecimal number. For example, once you complete the column for segment A, you should get the value 2812_{16}.

Create a new schematic file and name it as **LUT** (which stands for lookup table).

Add 7 **ROM16×1**s to the schematic. Label the address lines as **X0-X3**. Label the outputs from ROMs as **A-G**.

Select the first **ROM16×1** module (one attached to segment **A**) and select **Object Properties** from the pop-up menu. Set the initialization value of the ROM **INIT** to **2812**. Make sure to enter a 4-digit hexadecimal number. Click **OK** to close the dialogue box.

Repeat the above step for the remaining 6 ROMs. While this is tedious, unfortunately, no other efficient solution is available for a schematic design. This configuration is much simpler for VHDL or Verilog designs, which is not covered in this class.

Test the functionality of the LUT by simulating the circuit. At least try four distinct input cases based on the binary representation of your index no.

Create a symbol and name it as **LUT_16_7**.

Step 4: Build High-Level Circuit.

Create a new schematic file and save it as **AU_7_seg**.

Now add the 4-bit **AU** (from Lab 6) and **LUT_16_7** to the schematic. Connect their pins such that the output from RCA in AU is fed to the LUT.

Label the inputs to AU as **X0-X3**, **Clock**, and **RegSel**. Label the outputs as **A-G**, **C_out**, and **Z**.

Step 5: Connecting inputs and outputs.

Input and output pins are connected only to the top-level design **AU_7_seg** (so make sure to set the top-level design). Connect switches **SW0**-**SW3** as **X0**-**X3** inputs, **SW7** as **RegSel** input, and **BTN0** as **Clock** input. Connect outputs **A**-**G** to **CA**-**CG** (on 7-segment) and **C_out** to LED **LD6**, and **Z** to **LD7**.

Step 6: Test on BASYS2.

Generate the programming file (i.e., bitstream) and load it to the BASYS2 board.

Change the switches on the BASYS2 and verify the functionality of your circuit (check the output of 7-segment display and LED).

You may realize that though you plan to show the output using only a single 7-segment, all 4 7-segments will light up. Using the BASYS2 manual, find out how to show your results only on the right most 7-segment display.

Demonstrate the circuit to the instructor and get the Lab Completion Log singed.

Step 10: Lab Report

You need to submit a report for this lab. Your report should include the following:

- Student name and index number. Do not attach a separate front page
- State the assigned lab task in a few sentences
- Filled up table with hexadecimal codes
- All schematic circuits (label figures as Annex 1, Annex 2, ...)
- Timing diagram
- Briefly describe how can you show your results using only a single 7-segment
- Conclusions from the lab

Submit the lab report at the beginning of the next lab.

Prepared By

- Dilum Bandara, PhD – Mar 06, 2014.
- Updated on Oct 26, 2017.