Lab Exercise #1

ENGR-433

Objectives

1) Begin learning to use the Xilinx design tools
2) Design a digital circuit, implement it in an FPGA, and make it work.

References

Tutorial handout showing how to create a schematic based design using Xilinx tools.
Xilinx manuals (See class web page); Digital Logic textbook.
Pinout for the WWU FPGA3 digital logic board (also found on the class webpage)

Problem Statement

There are two tasks (two parts) for this lab:
Task 1) Implement the circuit in figure 1 to learn the Xilinx tool flow.
Task 2) Design a circuit that adds two numbers and displays the result.

The task 2 circuit is not exotic and you may even consider it mundane. However it was chosen as a straight forward circuit to illustrate an approach to designing digital circuits. The basic steps to follow for doing design are:

a) Create (or have) a clear definition of the problem. This includes defining inputs, outputs, function to be performed, and possibly specifications about how fast, how much, etc.
b) Decompose the big problem into multiple smaller problems, i.e. circuit blocks. Document with a block diagram. Note that this may lead to what is known as a hierarchical design.
c) As needed, decompose the prior blocks into multiple smaller blocks. Again document with a diagram. If the smaller blocks are not fundamental circuit units (for this lab those are basic logic gates) then decompose again. Repeat as needed.
d) Once steps b and c above are completed construct from the bottom up. Assemble components to create the lower blocks. Sometimes testing the lower blocks before combining to make a higher level block is worthwhile, particularly if their function is complex. Repeat as needed until the system is complete.
e) Test overall function of the assembled circuit (system) to confirm that it functions as defined and that performance meets specifications.

Circuits designed in this lab will be implemented in a Xilinx Spartan6 FPGA using schematic driven design capture

Procedures for this lab

.

Task 1

Follow the handout that details schematic driven design using the Xilinx ISE program to implement the circuit in Figure 1. Use switch 0 (SW0) as input, invert the signal, and display it on LED 0. Flipping SW0 should turn LED 0 on/off. The purpose is to learn to use the Xilinx tools. The schematic will look something like this (next page):

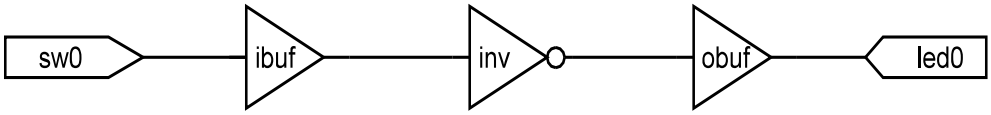

Figure 1

Task 2

Given problem: Design a circuit that will add two numbers and display the result. (A general statement of function but inadequate for design)

Expanded problem;

Improved problem description: Design a circuit that will add two three-bit numbers together and display the result in hexadecimal notation using a 7-segment display.

Specifications: 3-bit numbers as input, unsigned, result is one hexadecimal digit.

Input: One number, in binary, will be specified using switches sw2,sw1,sw0 (sw0 is the least significant digit). The second number will also be specified in binary using sw5, sw4, sw3. Numbers are assumed to be unsigned and positive.

Output: Use the right hand digit of the four digit 7-segment display.

Approach: First draw a block diagram showing the principle circuit functions. These likely will include the data source, 3-bit adder, code converter (binary to 7-segment), and display (see handout on block diagram drawing for expected format). Then design logic circuits for the respective blocks. (Note that the approach here is to decompose, i.e. break, the overall problem into smaller problems).

Adder circuit: For this lab you are to create the adder using fundamental logic components, i.e. AND, OR, NAND, NOR, XOR, NOT gates. And do it yourself. No using mister Google or his cousins. Note how we add two binary numbers together using long hand. The right most digits of the two binary numbers are added together to create a sum that is the right hand digit of the binary result. There may be a carry out into the middle digit. The middle two digits and carry in are added to create the middle digit of the result and possible a carry out, etc.

Below is a truth table for a 1-bit binary full adder (full adder means there is carry-in as well as data). Signals A and B represent two binary digits that are added together. Use it to create logic for a full adder. Replicate this circuit as needed. Document the circuit by drawing the logic gates needed to implement it.

Cin	A	B	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

3-bit adder circuit: Three 1-bit full adders can be combined to form a 3-bit adder where the carry out of the right hand adder connects to the carry-in of the adder to its left and the carry out of that digit to the carry-in of the digit to its left. Carry out of the leftmost 1-bit adder is the most significant digit of the result. The sum outputs of the three full adders create the least significant 3 bits of the result. We can label the most significant result digit as A and the least significant result digit as D.

Code converter: The binary result from the 3-bit adder needs to turn on appropriate segments of the 7-segment display. Below is a truth table showing logically the needed mapping (1 means the segment is on and 0 off). You can complete the column showing the 8-bit hex value associated with each character.

	A	B	C	D	a	b	c	d	e	f	g	8-bit hex
0	0	0	0	0	1	1	1	1	1	1	0	7E
1	0	0	0	1	0	1	1	0	0	0	0	30
2	0	0	1	0	1	1	0	1	1	0	1	6D
3	0	0	1	1	1	1	1	1	0	0	1	79
4	0	1	0	0	0	1	1	0	0	1	1	33
5	0	1	0	1	1	0	1	1	0	1	1	5B
6	0	1	1	0	1	0	1	1	1	1	1	5F
7	0	1	1	1	1	1	1	0	0	0	0	70
8	1	0	0	0	1	1	1	1	1	1	1	7F
9	1	0	0	1	1	1	1	0	0	1	1	73
A	1	0	1	0	1	1	1	0	1	1	1	77
b	1	0	1	1	0	0	1	1	1	1	1	1F
c	1	1	0	0	0	0	0	1	1	0	1	0D
d	1	1	0	1	0	1	1	1	1	0	1	3D
E	1	1	1	0	1	0	0	1	1	1	1	4F
F	1	1	1	1	1	0	0	0	1	1	1	47

There are multiple ways to implement logic for this. A 4x4 cell K-map could be created for each of the 7 segments a-g and a minimized function obtained for each. These functions could be implemented with AND/NAND/OR/NOR gates. Or, seven 4-bit selectors, one for each segment a-g, could be used. Or a ROM could be used as a look-up table.

A byte wide, 16 byte long, ROM can be used where signals A, B, C, D of the ROM truth table represent the input address of the ROM coming from the output of the 3-bit adder (A is the MSB and D the LSB of the address). There are 8 output bits of the ROM. Use the seven right hand bits of the output (g is the D0 bit and a is D6) as the signals going to segments a to g of the 7-segment display to turn them on. Turning on a segment requires the signal to be logic zero.

I recommend using a ROM, specifically a Xilinx component named RAM16x8S. While this is a RAM and we need a ROM, initial data values can be loaded into the RAM when the FPGA is programmed and if the WE (write enable) and WCLK (write clock) are never asserted then the data will not change in the RAM making it look like a ROM. As addresses change the output data will change.

Components used in a schematic can have properties. Properties have a name identifying them and then an associated value. For the ROM, after placing its symbol in the schematic properties named INIT_00, INIT_01, INIT_02, INIT_03, INIT_04, INIT_05, INIT_6, and INIT_07 can be added. The value (data to be placed in the RAM) for each of these will be 4-digit hexadecimal numbers that specify a column of data shown in the code convertor table above.

Testing: When implementing a circuit in an FPGA and wishing to confirm correct operation, it is not possible to randomly place a scope probe on circuit nodes after the circuit is installed in the FPGA. A useful strategy is to pick circuit nodes that may be useful for debugging and connect them to an output pin or an LED. For this problem, connect LEDs to the adder sum nodes, to observe for debugging (for this lab don't spend time putting your circuit into a simulation program. Test using LEDs or route signals to the EXTout pins where signals can be observed with a meter or oscilloscope).

To Turn In

For this lab each person is to create their own report, due next Tuesday. While this report will be quite short, there should be a title to the project and an abstract (a person reading the abstract should be able to learn what the problem was, the general approach taken to solve it, the solution, and the results or outcome). The body of the report for this lab is minimal. It should include a schematic and possibly comments about things that didn't work or things you learned.