2 Combinational Design: Binary-to-BCD conversion

This experiment introduces combinational circuit design using the Xilinx software for implementation and testing. This includes:

- **Schematic Capture**
- **Combinational Verilog**
- **Modular design**

By the end of this lab you should be able to…

...design a combinational circuit which coverts from one base to another

…use both schematic capture and Verilog to implement a combinational circuit

This lab assumes that you have purchased a Spartan 3 development board and have your own laptop computer to install the software.

You will need to fill out the lab DATA SHEET located at the end of this lab assignment during the performance of the lab.

I. Shift-Add-3 formula

……

Binary Coded Decimal (BCD) is used to represent binary numbers using the decimal system. Often calculations are performed in binary and then converted to BCD to display on LCDs. Each decimal digit has a 4-bit binary representation. Since there are ten decimal digits (0-9), there are ten different binary representations. Since four bits can represent sixteen individual codes, six codes are not used. The typical BCD values are shown in Table 1.

Decimal Digit	Binary Representation
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Table 1. BCD representation

Decimal numbers with multiple bits require a 4-bit BCD number for each bit. For numbers 10-99, we need 8 bits. For numbers 100-999, we need 12 bits. Table 2 illustrates this concept.

	Hundreds	Tens	Ones
Decimal	9	5	2
BCD	1001	0101	0010

Table 2. Multiple decimal digits

Note that if we convert a binary number to BCD, sometimes more than one digit location is required. The binary number `1100` represents the decimal number `12`. To convert a four bit binary number to a decimal number, any number greater than `1001` must be converted to two 4-bit BCD numbers. This is done by adding 6 to the binary number. For example, given the binary number `1100`, if six is added to this, the result is `0001 0010`. Converting each of these individual BCD numbers results in the decimal number `12`.

A similar computation could be done using shifts. Recall that a left shift moves all bit values one position to the left (toward the MSB). A left shift has the same function as multiplying the binary number by two. If we have the number `0101`, and we shift it left, we know the result will be `1010` (assuming we shift in a zero). If we then want to convert this number to a BCD, we will have to add six to it. Knowing that we will convert the number to BCD, we can check the original number before shifting it. If it is five or greater, then we can add three to that number. Then when we shift it, it will automatically be in BCD form. Table 3 illustrates this concept using the number `0110`.

Operation	**Binary**
Original number	0110
Add 3	1001
Shift Left	0001 0010
Decimal	**1 2**

Table 3. Preemptive add, then shift example

This concept can be used on a binary number with any number of bits. For each bit in the binary number, the shift-add-3 algorithm must be implemented. For a four bit number, four shifts must be performed. Before each shift is performed, if the number is greater than four, then it must be added to three. Once all four bits have been shifted, the final BCD number will be in the upper-most bits of the new number. See the example in Table 4.

Operation	**Tens**	**Ones**	**Binary**
Start			1111
Shift		1	111
Shift		11	11
Shift		111	1
Add-3		1010	1
Shift	1	0101	
Decimal	**1**	**5**	

Table 4. Four-bit Shift-Add-3 example

To perform a 4-bit binary conversion, 12 bits must be used. The upper four bits will be the tens decimal digit and the middle four will be the ones decimal digit. The lower four bits will be ignored. In the previous example, only the relevant bits were shown. In reality, the missing bit locations would contain zeros. This process can be extended for any size of binary number. Another example for an eight bit binary number is shown in Table 5.

Operation	Hundreds	Tens	Ones	Binary	
Start				1111	1111
Shift			1	1111	111
Shift			11	1111	11
Shift			111	1111	1
Add-3			1010	1111	1
Shift		1	0101	1111	
Add-3		1	1000	1111	
Shift		11	0001	111	
Shift		110	0011	11	
Add-3		1001	0011	11	
Shift	1	0010	0111	1	
Add-3	1	0010	1010	1	
Shift	11	0101	0101		
Decimal	**2**	**5**	**5**		

Table 5. Eight-bit Shift-Add-3 example

II. Combinational Implementation

……

The Shift-Add-3 algorithm can be implemented using all combinational logic. The truth table for the individual Add-3 module is shown in Table 6. It has a 4-bit input and a 4-bit output. When the set of inputs is less than 5, it outputs the same number. When the set of inputs is greater than four, it adds three to the input. Given the implementation, the inputs should never be greater than the number nine, so all inputs above nine are don't care.

A3A2A1A0	S3S2S1S0
0000	0000
0001	0001
0010	0010
0011	0011
0100	0100
0101	1000
0110	1001
0111	1010
1000	1011
1001	1100
1010	XXXX
1011	XXXX
1100	XXXX
1101	XXXX
1110	XXXX
1111	XXXX

Table 6. Add-3 module truth table

To shift a set of numbers in a combinational circuit, the inputs are connected to the outputs, but shifted by one position. So the least significant bit would route to the second least significant bit. This can be repeated for each stage the Shift-Add-3 operation must occur. Note in the example in Table 5, only when there are three or more bits in a column does it need to be checked for Add-3. The circuit for the Shift-Add-3 is shown in Figure 1.

Your objective for the this lab is to implement the Shift-Add-3 circuit in the Xilinx design software and download the circuit to your Spartan 3 development board. The input to the circuit will be the eight switches and the output will be the four-digit seven-segment LED array.

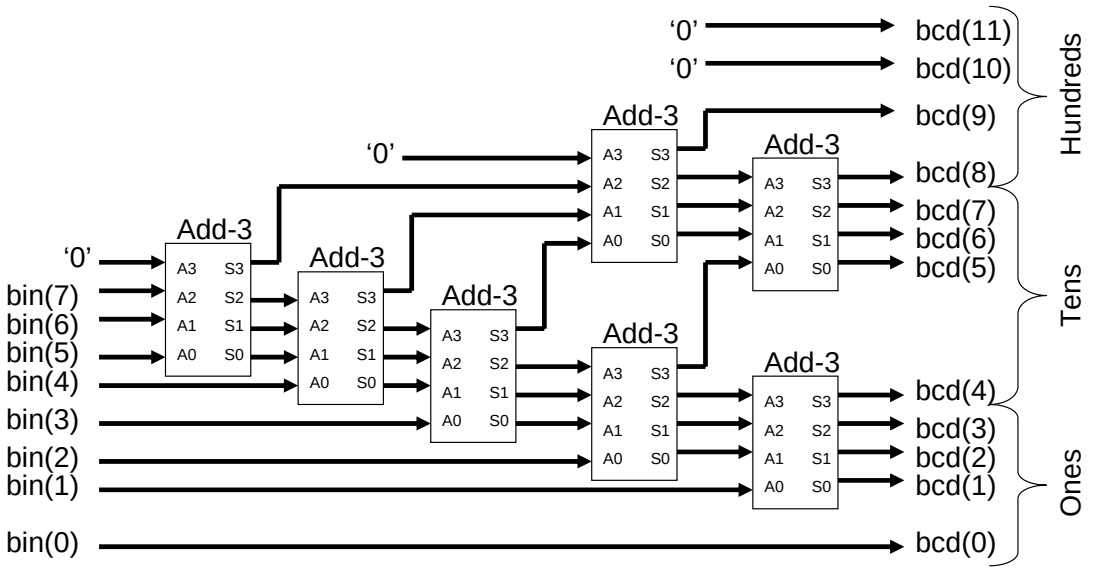

Figure 1. Shift-Add-3 circuit using the Add-3 module

III. Laboratory Work

……

Setting up the project file

1. First extract the compressed files into a project directory called "Lab2". It would probably be convenient to put it in the same place as Lab1.
2. Load Project Navigator and open the lap2top.sch schematic. Notice that the eight switch inputs are connected through buffers to the BCD outputs. This will make the LCDs display the binary values in hexadecimal. In this configuration only the right-most characters are used, so the left two characters are connected to ground (so they will display zeros).
3. The BCD outputs are not directly connected to the LCD. A special interface is needed to properly format the outputs to the LCD. You won't need to know how the interface works to complete the lab.
4. Compile the program and download the design to see your board displaying the hexadecimal outputs.

Grade B: Using schematic capture for the Add-3 module

1. Reference the truth table for the Add-3 module.
2. Write a boolean equation for S3, S2, S1, S0. You can use Karnaugh maps (K-maps) to minimize the logic required. You can also use minterm notation. Reference chapter 2 in your textbook if you do not remember how. The Xilinx software will minimize your logic if you don't want to; however, simpler equations mean fewer mistakes.
3. Implement each of the four equations in the schematic file "add3.sch"using the Xilinx design software package. You will need to use input/output markers, gates (found under Logic symbols), and connect them all with wires.
4. Once the design is complete, synthesize the circuit by executing the **Synthesize – XST** process.
5. Create a schematic symbol with this circuit by executing **Create Schematic Symbol** under the **Design Entry Utilities** process.
6. Open the top level schematic "lab2top.sch".

   Note: To check the functionality of your Add3 module, refer to the class notes on how to build a 4-bit binary-to-BCD converter. Implement this circuit first and download it to your Spartan 3 board. Check all 4-bit combinations on the numeric display before continuing.

7. Follow Figure 1 to implement your circuit. Use the Add-3 module you created to place into the design. In the categories window, select the second line which is the directory name for your project. The symbol name "add3" will appear in the bottom left window.
8. Note that a single wire cannot have two names. If you want to connect a wire to ground, it must first go through a buffer.
9. Compile your source and download the design to your Spartan 3 board.
10. Have the TA evaluate your assignment.

Grade A: Using Verilog for the Add-3 module

1. Create a new Verilog source module by right-clicking in the sources window and selecting "New source".
2. Select Verilog module and type "add3veri.v" in the file name box. Click "Next".
3. Enter the four inputs and four outputs under the heading "Port Name". You should have 8 lines when finished. Be sure to select under "Direction" whether they are inputs or outputs. Click "Next" and then "Finish".
4. A Verilog template will open which has the entity completed. You will only need to complete the architecture portion.

5. Implement the four equations using concurrent statements. This is more efficient than using if-else statements with an adder/comparator.
6. Once you have completed the Verilog code, create a new symbol, and change the top-level schematic to use your Add3 Verilog module instead of the schematic Add3 module.
7. Compile your source and download the design to your Spartan 3 board.
8. Have the TA evaluate your assignment.

LAB DATA PAGE **Name: _____**

Grade B. Using schematic capture for the Add-3 module

1. Program compiles (Yes/No) _____
2. Using schematic capture for output equations (Yes/No) _____
3. All eight switches change the display (Yes/No) _____
4. All numbers correctly displayed (Yes/No) _____

Total hours reported (from work log) for this lab portion: _____

TA CHECKOFF SIGNATURE: _____(must be legible!)

Grade A: Using Verilog for the Add-3 module

1. Program compiles (Yes/No) _____
2. Using Verilog for outputs (Yes/No) _____
3. All eight switches change the display (Yes/No) _____
4. All numbers correctly displayed (Yes/No) _____

Total hours reported (from work log) for this lab portion: _____

TA CHECKOFF SIGNATURE: _____(must be legible!)

Student Evaluation:

What did you like most about this lab?

What would you change about this lab to make it better (not necessarily easier)?

WORK LOG PAGE **Name: _____**

Lab portion worked on / completed	**Start Time & Date**	**Stop Time & Date**	**# Hours:**
Total Hours:		Grade C:	
		Grade B:	
		Grade A:	