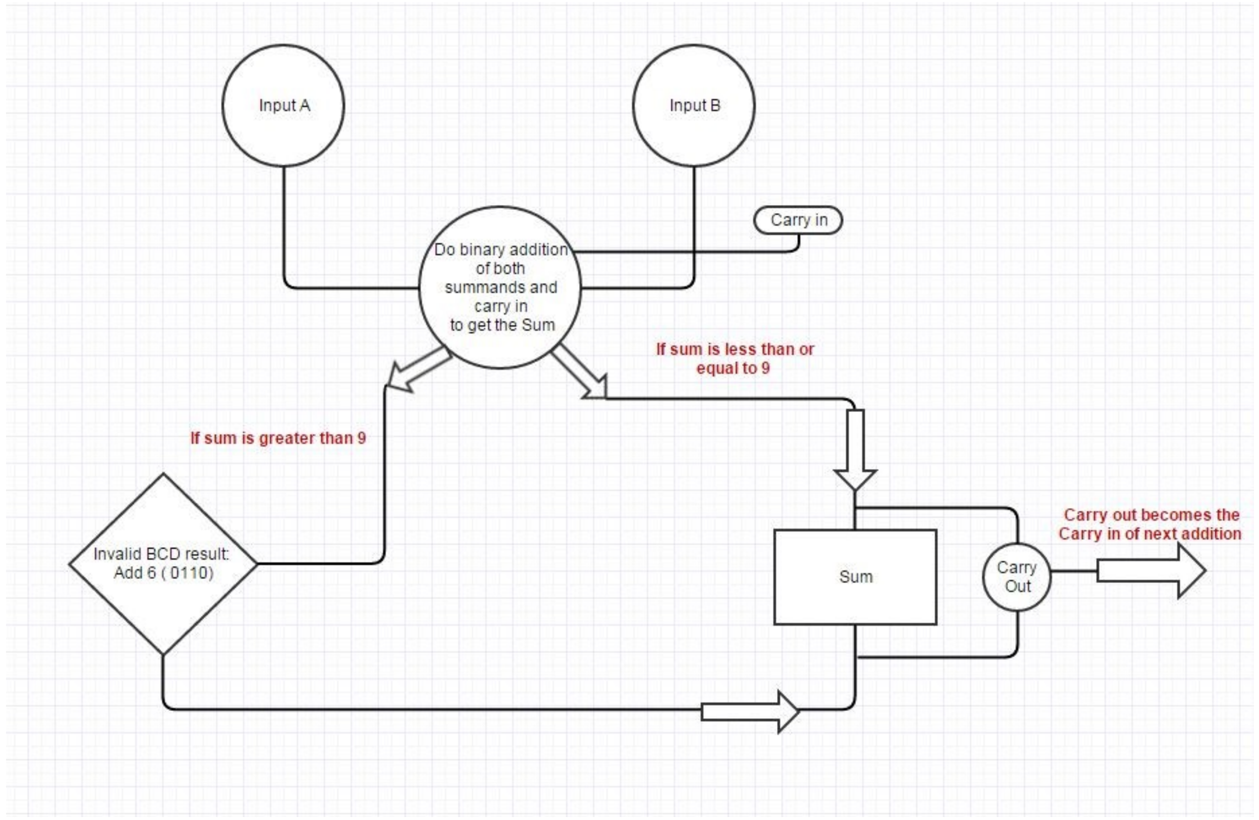
I. Objectives

Build a binary-coded decimal digit adder using the results of past labs on multiplexers, comparators, adders, etc.

II. Introduction

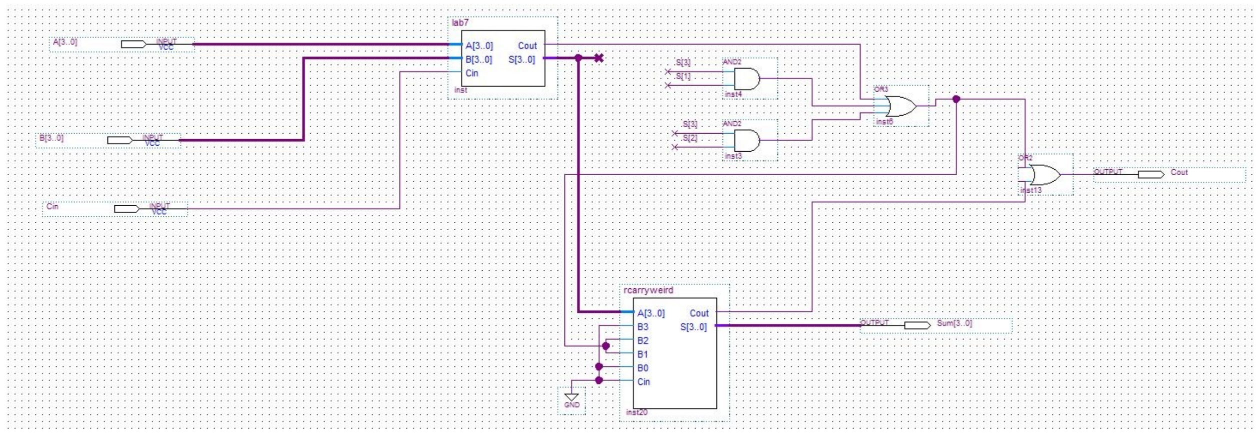
In this lab we build a binary- coded decimal. We used the flow chart from our pre lab to help better understand the process (Flow Chart 1). The BCD adder takes in 2 inputs and performs addition. If the sum is greater than 9 then it adds 6 to the result, if not then that result is the sum. The sum and carry out are the output, and the carry out is used as the carry in, in the next addition.

Flow Chart 1: BCD


Input A
Input B
Carry in
Do binary addition of both summands and carry in to get the Sum
If sum is less than or equal to 9
If sum is greater than 9
Invalid BCD result: Add 6 (0110)
Sum
Carry Out
Carry out becomes the Carry in of next addition

We took the insight that flow chart gave us to create a rough diagram of a 1-digit BCD adder (Schematic 1). The Schematic below features 2 ripple carry adders and AND and OR logic gates.

Schematic 1: Rough BCD Adder



During our lab we used the rough diagram to build our final schematic. Our BCD adder outputs to the FPGA board's 7-segment display(displaying both summands and sum), it utilizes the switches to change the number of each input, and the LED to alert of a carry out.

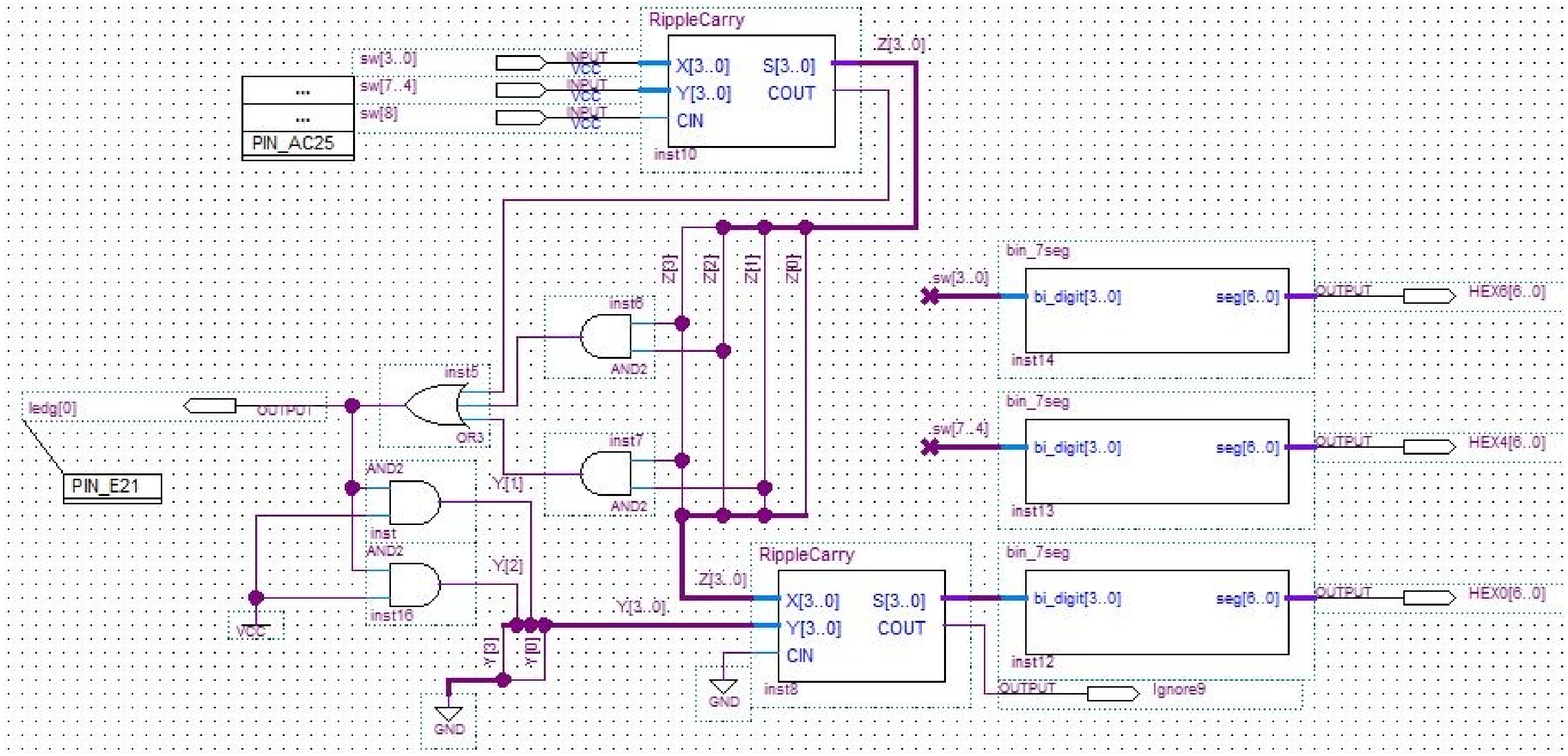
III. Procedure

In order to create our BCD adder, we had to refer to our flow chart (Flow Chart 1) and rough diagram (Schematic 1) made for pre-lab. We know that BCD has 6 unused binary and the one's that are used have a decimal value of 0 to 9. If the decimal value sum of two BCD code numbers adding together goes above 9 (10 to 15), then we need to add 0110 (decimal value 6) to the 4 bit binary sum so we get the desired output and have the sum fall back into a valid code. With this knowledge we know we must include two 4-bit ripple carry adders, one with inputs of the numbers adding and one with the sum of inputs adding to 0110. We do not want 0110 to be added to every case however, only when sum is greater than decimal value 9. This happens when both the leftmost digits([3] and [2]) are 1, when digits [3] and [1] are 1, or when there is a COUT in the two input addition. Therefore, we must connect all of these possibilities into an OR gate. That OR gate will decide the COUT of the BCD adder and if 0110 will be added to the sum or merely 0000 (which 0000 will have no effect on the sum). Once that addition is done, we then get our final sum output. Also, when 0110 is added, this means a 1 or (0001) is carried out to the next BCD adder if there was more than one. Once the BCD Adder was created, we made sw[3..0] input A, sw[7..4] input B, and then connected each one to it's own 7-seg display. The final sum was also connected a 7-seg display. COUT was connected to a Green LED.

We decided to test our circuit by programming the FPGA Board and testing it there. We had minor trouble shoot since wires were not connected correctly with the first design, but we fixed it and got the desired results. With the desired results our BCD Adder was a success and we then proceeded to the extra credit.

You can see our schematic (Schematic 2) follows the correct BCD Adder logic. An example case would be is if sw[0,3,4,5] are on (1) and the rest of the switches were off (0), this means 1001 and 0011 are being added or BCD numbers 9 and 3. The binary sum is 1100 with 0 carry which is greater than decimal value 9, thus an AND gate is activated (1) by Z[3] and Z[2] being on (1) and this activation activates the OR gate. The OR gate gives off a COUT (ledg[0] in this case) and activates 2 AND gates which give off on values (1) to Y[2] and Y[1]. This means 0110 (decimal 6) is added to 1100 and we get the final output of 0010 which is BCD code for 2. 2 Is then displayed on 7-seg display which connected to the final sum S[3..0]. Input A(sw[3..0]) show up as 9 on one 7 seg display and input B(sw[7..4]) show up as 3 on another 7-seg display.

Schematic 2: BCD Adder



IV. Results

After we finished building the BCD adder we ran it on the FPGA board to test it. We had one summand on the 7 seg display on the far left and the other in the middle. The sum displayed on the far right. Switches 0,1,2,3 controlled the number input for one of the summand while switches 4-7 controlled the other. When there is a carry out LED 0 will turn on. To test we started off with easy numbers like 2 (0010) + 1 (0001) which displayed the sum of 3. Then we tried numbers were we expected an overflow like 9 (1001) + 7 (0111) and it displayed the sum of 6 and the LED was on, meaning there is a carry out. We test other numbers and each time we got the output we were expecting.

V. Conclusion

For the most part this lab went smoothly. The pre lab really laid the groundwork for the BCD adder. When we went to test our schematic for the first time all of our sums were off by 1. For example when we tried 4 (0100) + 3 (0011) it would display 6 instead of the correct output of 7. When we first attempted the schematic we connected the sum output(S[3..0]) of the first ripple carry adder to an output to make it easier to access when we were using it in the second ripple carry adder. However we think this caused our design to act funny, it also added an additional output to our symbol which we didn't want. When we took out the output and changed the wiring the problem went away. When we tested our schematic a second time we noticed that it would add correctly as long as the sum was less than or equal to 9. However when the sum was greater than 9 it would not display the sum correctly (using only digits 0-9), it would display the hexadecimal value of the numbers for 10-15 (A-F). But the carry out LED was functioning properly. For example when we would enter 9 (1001) + 3 (0011) it would output C with the carry out LED on. It should of outputted 2 with the carry out LED on. We went back to look at our schematic and we noticed that the problem was the way we split the wires to do the add 6 operation (by using AND gates). It was ANDing all the wires together instead of doing them separately. So we fixed that but adding extra AND gates to make it cleaner and rewiring it. When we tested our schematic a third time it ran perfectly. After we processed to work on a 2 digit BCD extra credit assignment. This lab let us look in depth on Binary Coded Decimals and help us better understand how they work. We also became more familiar with quartus and learned a few tricks for wiring schematics.

VI. References

- Krishnan, S. *Laboratory #7: BCD Adder*. 2016. Print.
- Quartus II Tutorial
  - ftp.altera.com/up/pub/Altera_Material/12.1/Tutorials/Schematic/Quartus_II_Introduction.pdf
- Waveform Simulator Tutorial
  - ftp.altera.com/up/pub/Altera_Material/13.1/Tutorials/Verilog/Quartus_II_Simulation.pdf
- Pin Assignment Chart
  - ftp.altera.com/up/pub/Altera_Material/12.1/Boards/DE2-115/DE2_115_User_Manual.pdf
- 7-Segment Display Tutorial
- bin_7seg_temp.txt