ECE2277A Lab 4: Excess-3 Adder/Subtractors

Laboratory Report

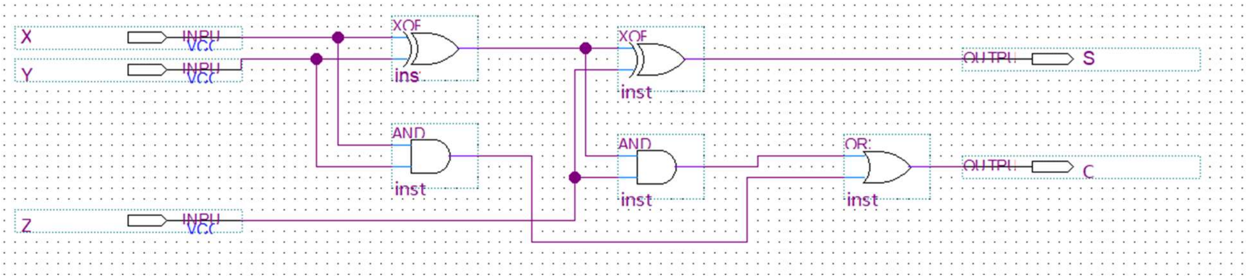
Name: Waleed Sawan

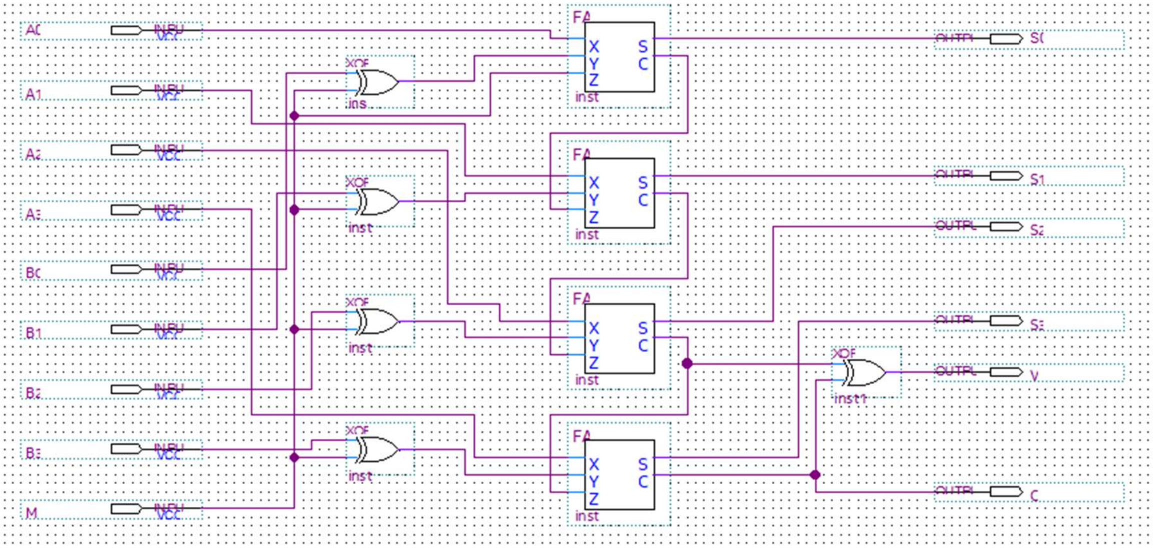
Student Number: 251085963

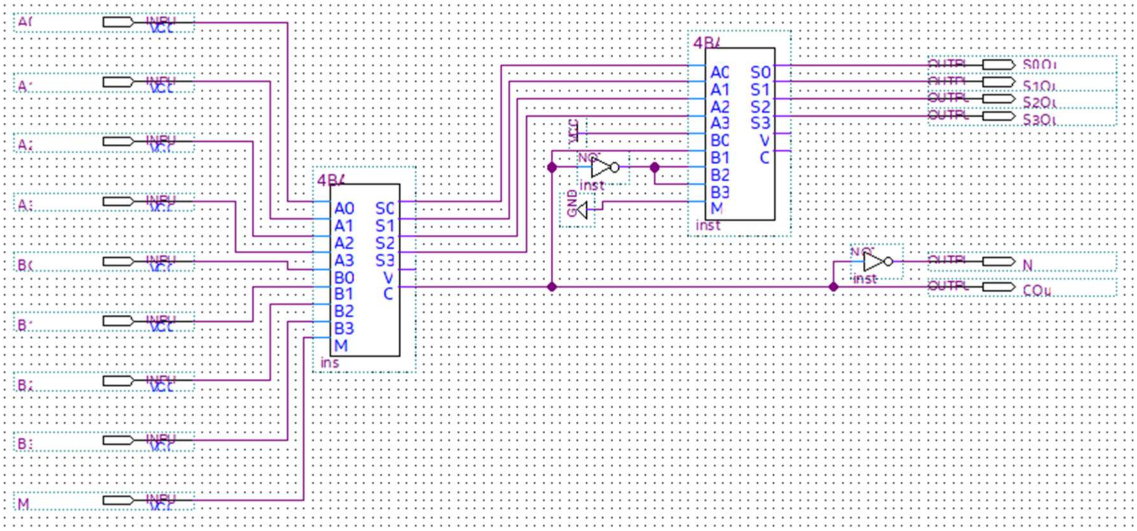
Submission Date: 2020-12-01

4-Bit Excess-3 Adder/Subtractor Circuit

Please include image(s) of your circuit from Quartus. If you generated your own custom symbols, include an image of that circuit as well. Please write a label or caption below each image to help identify what each circuit block is. Add extra space as necessary.





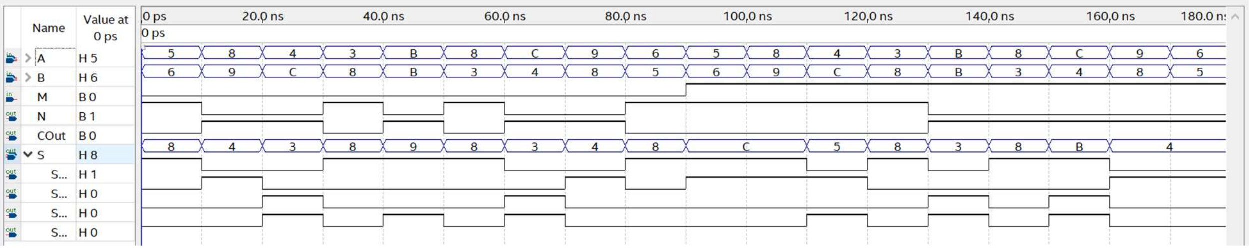


Simulation Output

Please complete the table below. Please write all values in binary, exactly as used for input or output by the circuit.

Time Step	$A_3A_2A_1A_0$	$B_3B_2B_1B_0$	M	$S_3S_2S_1S_0$	C	N
1	0101	0110	0	1000	0	1
2	1000	1001	0	0100	1	0
3	0100	1100	0	0011	1	0
4	0011	1000	0	1000	0	1
5	1011	1011	0	1001	1	0
6	1000	0011	0	1000	0	1
7	1100	0100	0	0011	1	0
8	1001	1000	0	0100	1	0
9	0110	0101	0	1000	0	1
10	0101	0110	1	1100	0	1
11	1000	1001	1	1100	0	1
12	0100	1100	1	0101	0	1
13	0011	1000	1	1000	0	1
14	1011	1011	1	0011	1	0
15	1000	0011	1	1000	1	0
16	1100	0100	1	1011	1	0
17	1001	1000	1	0100	1	0
18	0110	0101	1	0100	1	0

Please include an image of the simulated waveforms, which should match the values filled into the table above. Add extra space, as necessary.



Design Questions

Answer the questions below. Only a few sentences are probably sufficient. Add extra space if necessary. Your descriptions can be in words, logic functions (i.e. $S_1 = A_1B_1 + M'B_2$, for example), or a combination of both.

1. **Is there a relationship between the carry-out (control output `C`) and the borrow (control output `N`)? Explain your answer. [3 marks]**

There is an inverse relationship between the carry out, C, and the borrow out, N – essentially meaning that there will be no scenario in which both values are equal (when C = 1, N cannot equal 1). This is because its impossible to have both a borrow and carry bit in binary arithmetic.

2. **If you look only at the output of your circuit ($S_3S_2S_1S_0$, C, N), how can you tell if *S* = *A* + *B* (i.e., a valid addition)? [3 marks]**

Simply enough, you can do binary addition of the value of A and B. Once you have the result of that, you can compare it to the result of the circuit wavelength result. If its the same value, then S is in fact correctly calculated through addition.

3. **If you look only at the output of your circuit ($S_3S_2S_1S_0$, C, N), how can you tell if *S* = *A* - *B* (i.e., a valid subtraction)? [3 marks]**

Simply enough, you can do binary subtraction of the value of B from A. Once you have the result of that, you can compare it to the result of the circuit wavelength result. If its the same value, then S is in fact correctly calculated through subtraction.

4. **Describe how you implemented the circuit to convert from the unused excess-3 codes to valid excess-3 codes. [5 marks]**

This took a fair bit of research/experimentation, but it was found that to convert from the unused excess-3 codes to valid excess-3 codes, you had to add or subtract 0011 to make the necessary conversion. For example, if there was a carry out of 1, it would make a subtraction of 3. On the other hand, if there was a borrow bit – it would cause an addition of 3 to convert the result (S) to a valid excess-3 codes.

5. **Describe how you resolved the ±3 issue when adding or subtracting excess-3 codes. [5 marks]**

As described in the previous question, having either a carry out bit of 1 (addition of 0011) or a borrow bit of 1 (subtraction of 0011), allows the resulting S to be properly shifted (±3) to the correct output of the excess-3 addition/subtraction code. In the circuit sense, this is done by the secondary part and the intermediary step. The first 4-bit adder produces an unusable excess-3 code and those two additional parts convert it back to a valid excess-3.

6. **Justify your design choices in terms of the relative cost of your circuit to other possible designs (you don't need to calculate the exact cost of your circuit, but you should have a rough idea whether your circuit is more or less cost-efficient than other designs). [6 marks]**

One opposing design that I looked at and sort of worked towards was a conversion between 3-excess codes and BCD. The process seems simply at first, but would obviously incur a much higher circuit cost at first glance:

1. Convert Excess-3 Codes to BCD by subtracting 0011
2. Perform BCD addition/subtraction
3. Convert from BCD back to Excess-3 by adding back that 0011

There are other potential circuit designs that also provide a varying cost that is seemingly less efficient than the one shown above. Another key example is doing a k-map minimization of the variables on hand using a less cost-efficient 4-bit adder design.