Benny Chen
2/23/2022
Lab 4

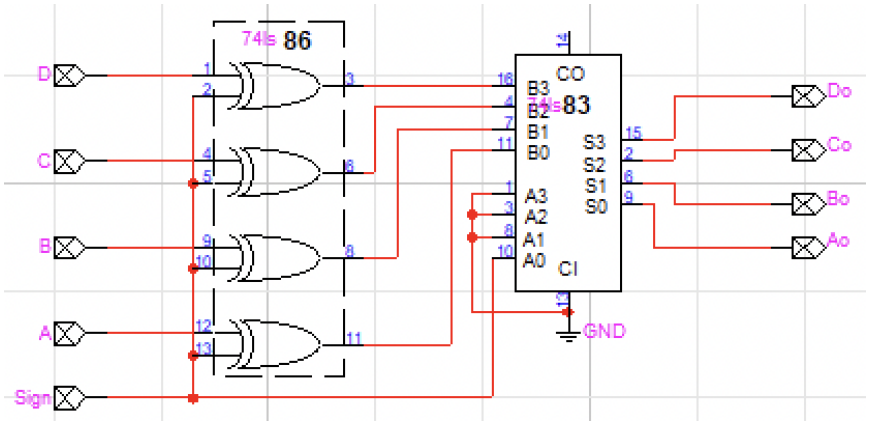
Theory

In this lab we created two circuits, one adding three unsigned values together with each value having a max of 15 that when added together has a max of 45. For the other circuit, we created another adder but for this one we have two values that are able to have signed values. In this lab we use new components like a bit adder and use techniques like 2's complement. 2's complement is very simple to do, flip every bit in a binary number then add one. We use 2's complement to represent negative numbers as positive and so converting negative numbers to positive to be able to do operations on them. We have to convert negative numbers to 2's complement as doing most operations on them will give out an overflow and also it's more effective in recognizing these negative numbers. For example using 2's complement it will understand that -0 isn't really a negative number and so computations would be smoother. As we are converting to 2's complement, we can only represent -8 to 7 in binary. For this circuit we first use 2's complement to convert the number to positive if it's negative, if not it wont. We do this by using XOR gates with each 4 bit value being XORed by its signed bit. We then use a 4 bit adder to add these two values together then we use two's complement again to flip it back.

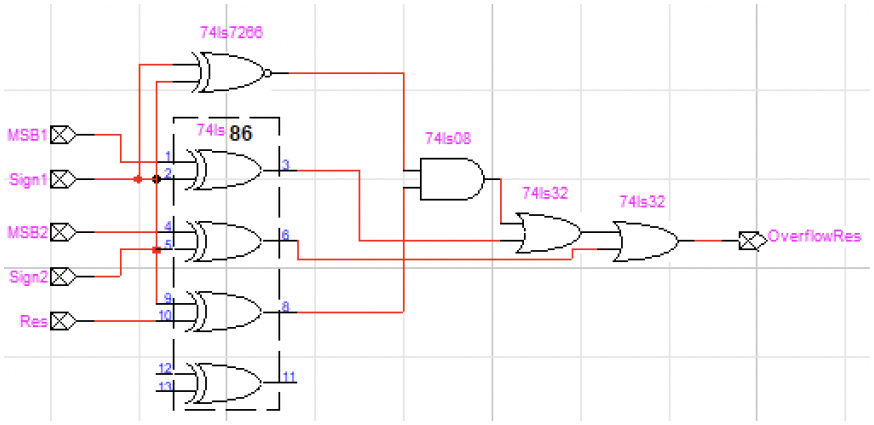
In this lab we also have to implement our own overflow detector as we are using 2's complement operations making the bit adder "carry out" very misleading. Since we are using 2's complement, we can only have numbers between -8 to 7. The bit adder would work for normal positive numbers but for this instance we detect overflows between -8 and 7 and also for negative numbers to create "subtracting". For example take 5 and -3, we technically are subtracting 5 - 3. When put in our circuit it'll give out 2 without overflow, but in the bit adder it has a "carry out" of 1. This is due to the -3, represented as 1 0011, with its 2's complement as 1 1101 which is then added to 5, represented as 0 0101, with its 2's complement of it being the same. The bit adder technically adds 5 and 3 to get 8 which is correctly an overflow, but this is the 2's complement so we then have to revert it back. This is why it's very misleading as 8 is an overflow but it doesn't factor in the 2's complement back which makes it 2, which isn't an overflow. To solve this issue we have to make our own overflow logic combination. For this logic combination we check for if the sign of the original values after getting the result is the opposite sign, if this holds true we know an overflow has occurred. For my overflow detector, I get the MSB of the numbers added together and the result and 2's complement to revert it back to its original form. I then compare the original signed bits and XNOR them to return 1 if they are the same signed bit or 0 if they aren't. This number is then ANDed with the resulting MSB that is in its original form to check if they match. All these numbers are then ORed together to get the overflow after checking.

Deliverables

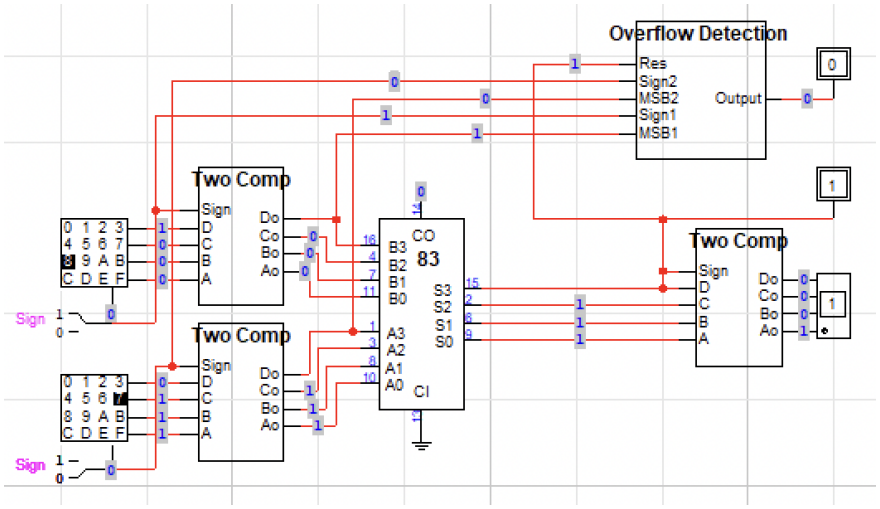
2's Complement Device Symbol



Overflow Detector Device Symbol



Circuit



Test Cases

Magnitude	Sign	Din	Cin	Bin	Ain		Sign	Dout	Cout	Bout	Aout	Overflow?
0	1	0	0	0	0							
0	0	0	0	0	0		0	0	0	0	0	0
-3	1	0	0	1	1							
5	0	0	1	0	1		0	0	0	1	0	0
3	0	0	0	1	1							
-5	1	0	1	0	1		1	0	0	1	0	0
-3	1	0	0	1	1							
-3	1	0	0	1	1		1	0	1	1	0	0
-4	1	0	1	0	0							
-5	1	0	1	0	1		0	0	1	1	1	1
3	0	0	0	1	1							
3	0	0	0	1	1		0	0	1	1	0	0
5	0	0	1	0	1							
5	0	0	1	0	1		1	0	1	1	0	1
-8	1	1	0	0	0							
7	0	0	1	1	1		1	0	0	0	1	0

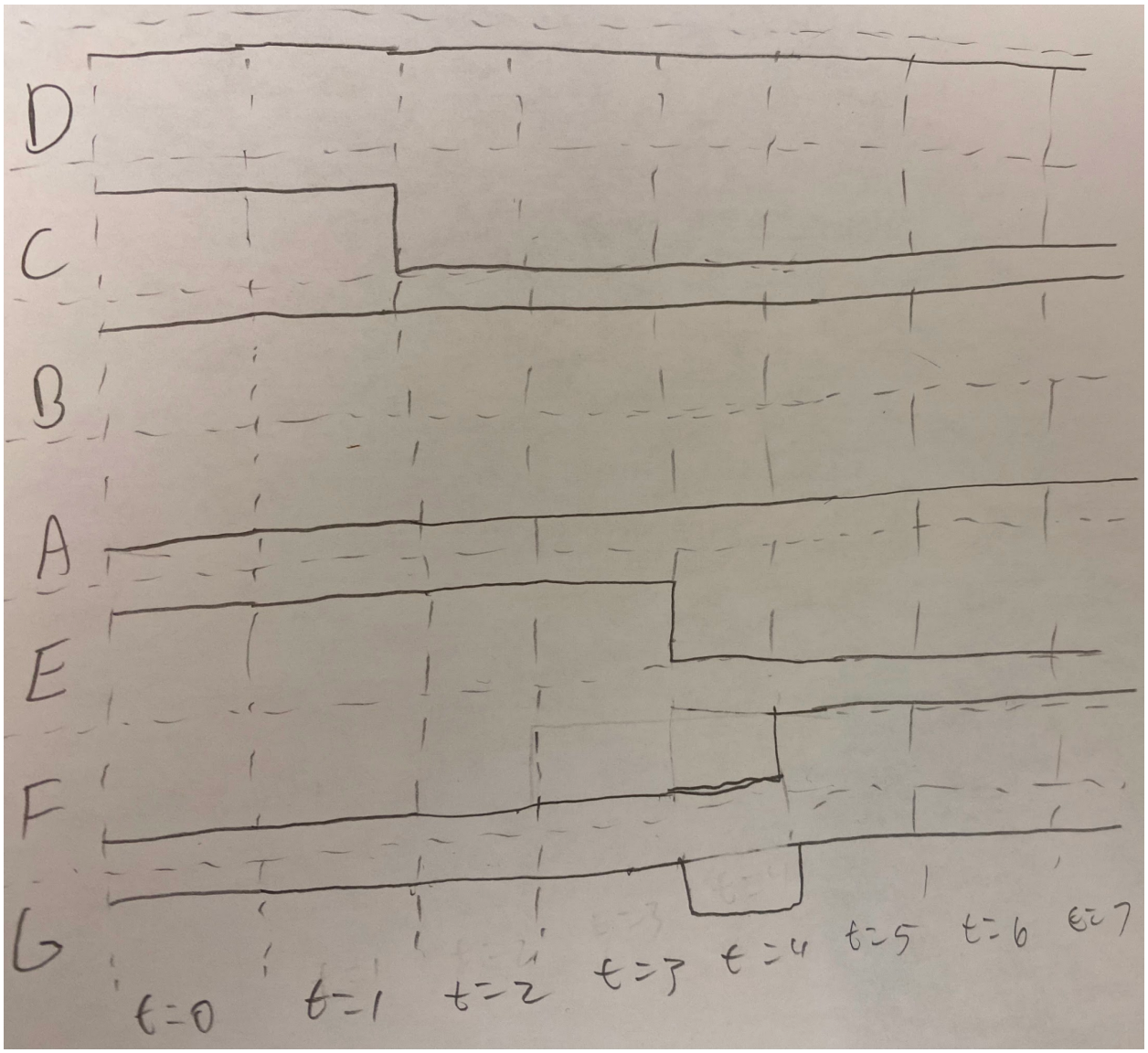
For the test cases I used all the edge inputs that are needed before an overflow and inputs that do cause an overflow. For example, I used 5 and 5 to simulate a positive overflow and -4 and -5 to simulate a negative overflow. I also demonstrated that -0 and 0 don't give out overflows as 0 isn't supposed to change signs. I also did normal operations with the numbers like -3 and -3 to give the correct answer of -6 and 3 and 3 to get an answer of 6. I also did operations with positive and negative numbers together with negative numbers being the first value then the second to test out subtraction. Lastly I did an operation between -8 and 7 which are the max values before an overflow.

Discussion

In this lab we were introduced to bit adders and implementing 2’s complement. From this lab I understand how 2’s complement can be used on top of how bit adders work. We also learned about creating a combinational circuit to detect overflows as the “carry out” isn't really the true overflow of the product. This part of the lab was the most challenging as all you can do is test out the outputs to see what the pattern is for an overflow. I then found out about using XNOR of the sign bits which we compare to the resulting bit using AND then ORing them together to find an overflow.

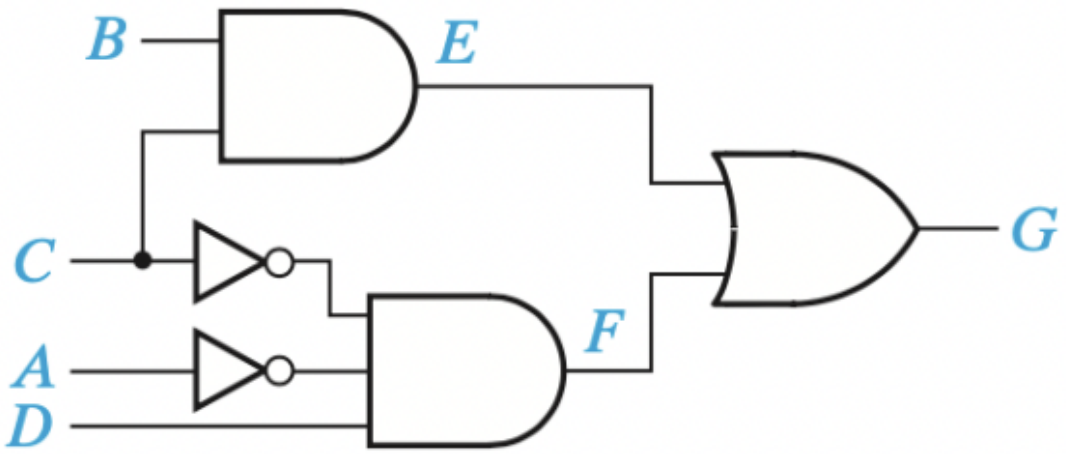
Questions

1. The following questions require knowledge from module 6. Use the image shown below. Assume that the inverters have a delay of 1ns and the other gates have a delay of 2 ns. Initially A = 0 and B = C = D = 1, and C changes to 0 at time = 2 ns. Draw a timing diagram and identify the transient that occurs.



2. **Modify the circuit to eliminate the hazard.**

Original:



Fixed:
Remove NOT gate from C