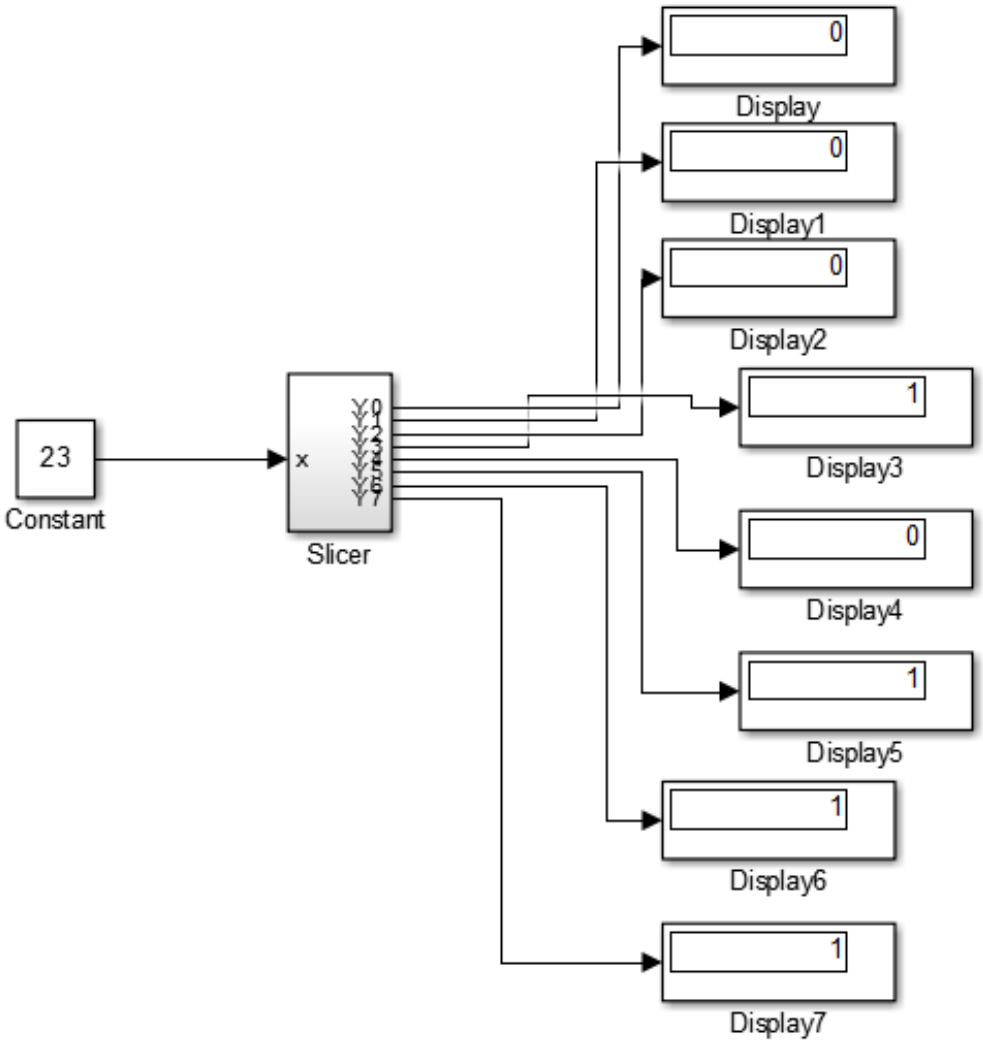
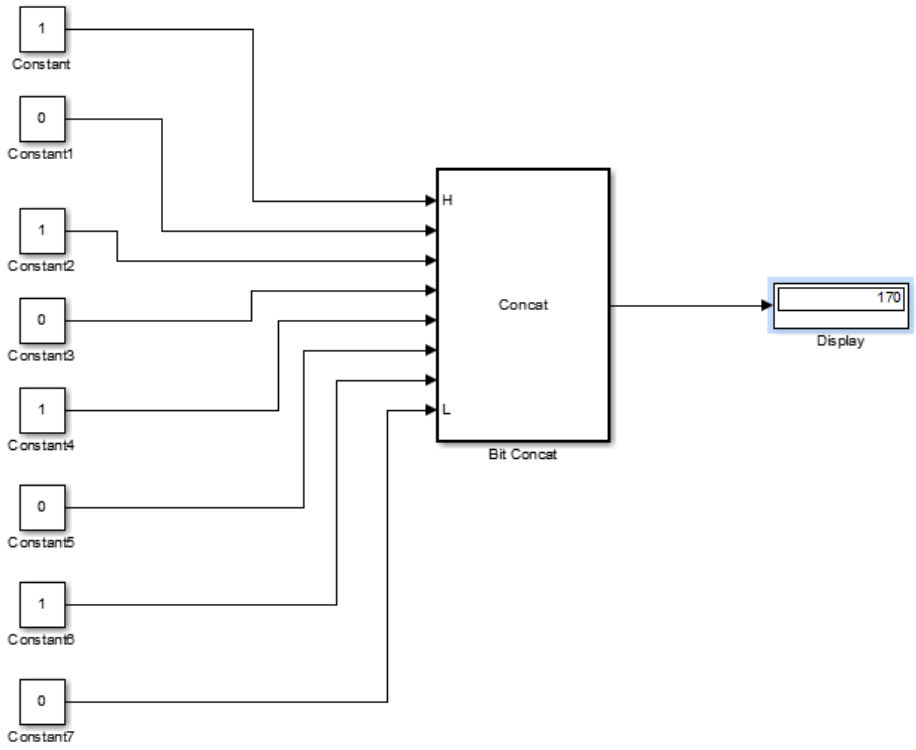
Evan Noyes and Henry Gridley
Lab 7

Assignment 1



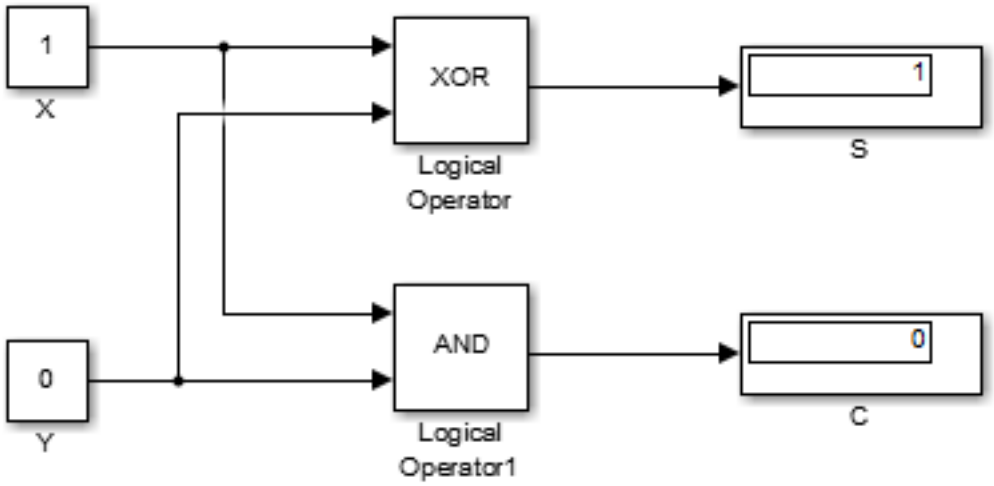
This output shows the number 23 being split into its correct binary representation from bottom to top: 11101000 (1 + 2 + 4 + 16)

Assignment 2



This shows how the binary number 10101010 can be converted into the decimal number 170 (2 + 8 + 32 + 128)

Assignment 3

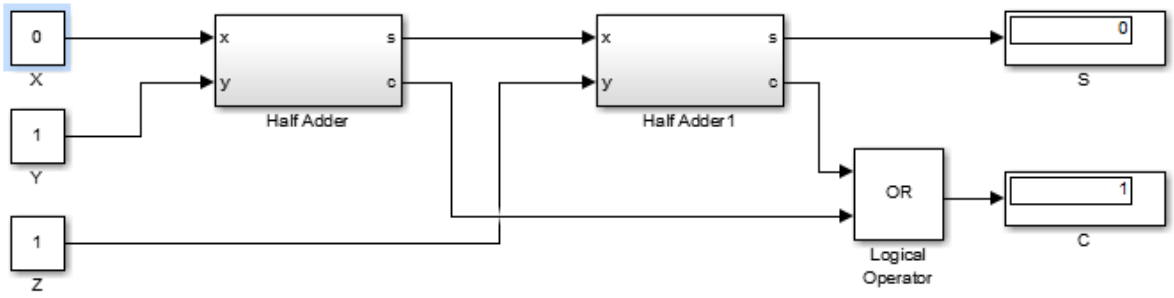


The XOR and AND gates solve the problem of adding two 1 bit numbers because they allow you to accurately get a sum and a carry bit.

In the following truth table you can see how this is represented.

X	Y	Sum	Carry
0	**0**	0	0
0	**1**	1	0
1	**0**	1	0
1	**1**	0	1

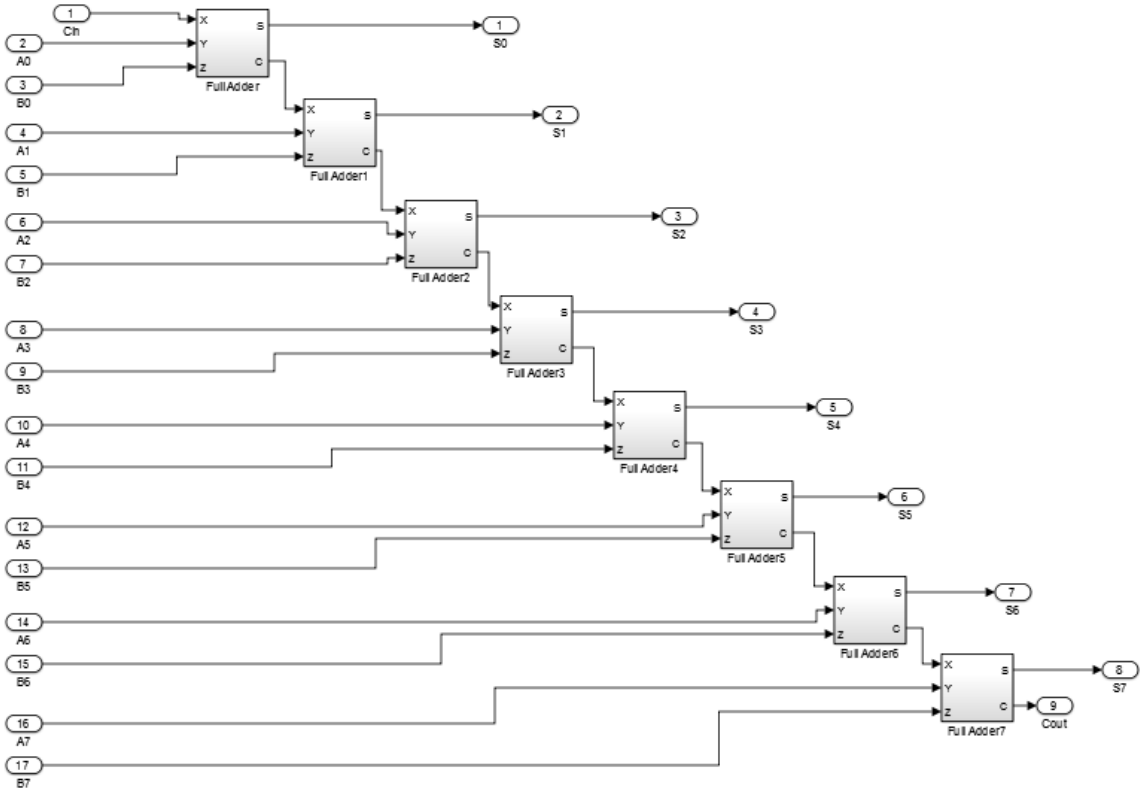
Assignment 4



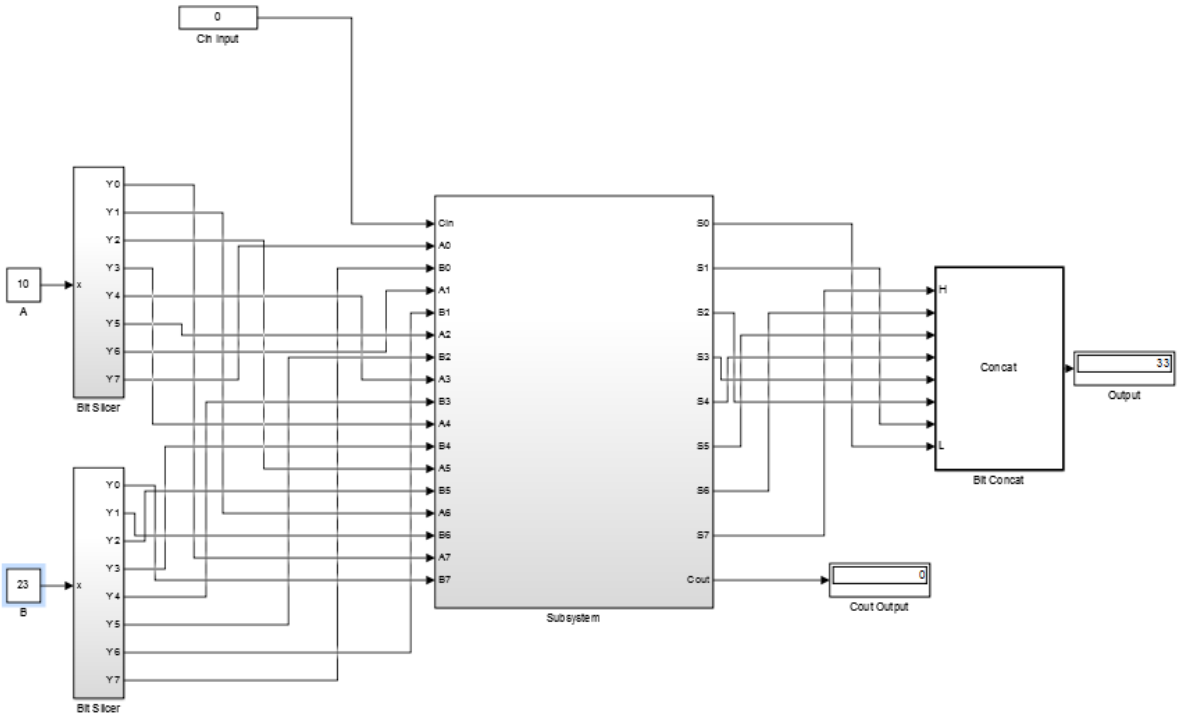
The full adder is similar to the half adder but also contains a way to pass along a persistent overflow value so that you can create a chain of them that will eventually add together larger numbers. Since the highest value of the result can be 3 (11_2) if there is an overflow bit inputted and created as a result of the adders, it is discarded by going through the OR gate. This ensures if there is at least one carry there will be only one carry.

Assignment 5

Here is the schematic for the binary adder



And here is the output



As you can see, the numbers 10 and 23 are correctly added to equal 33. The purpose of the output signal cout is to detect if an overflow occurred. If that signal is triggered than the two numbers have produced a result larger than 8-bits (256) , the maximum space to record the result