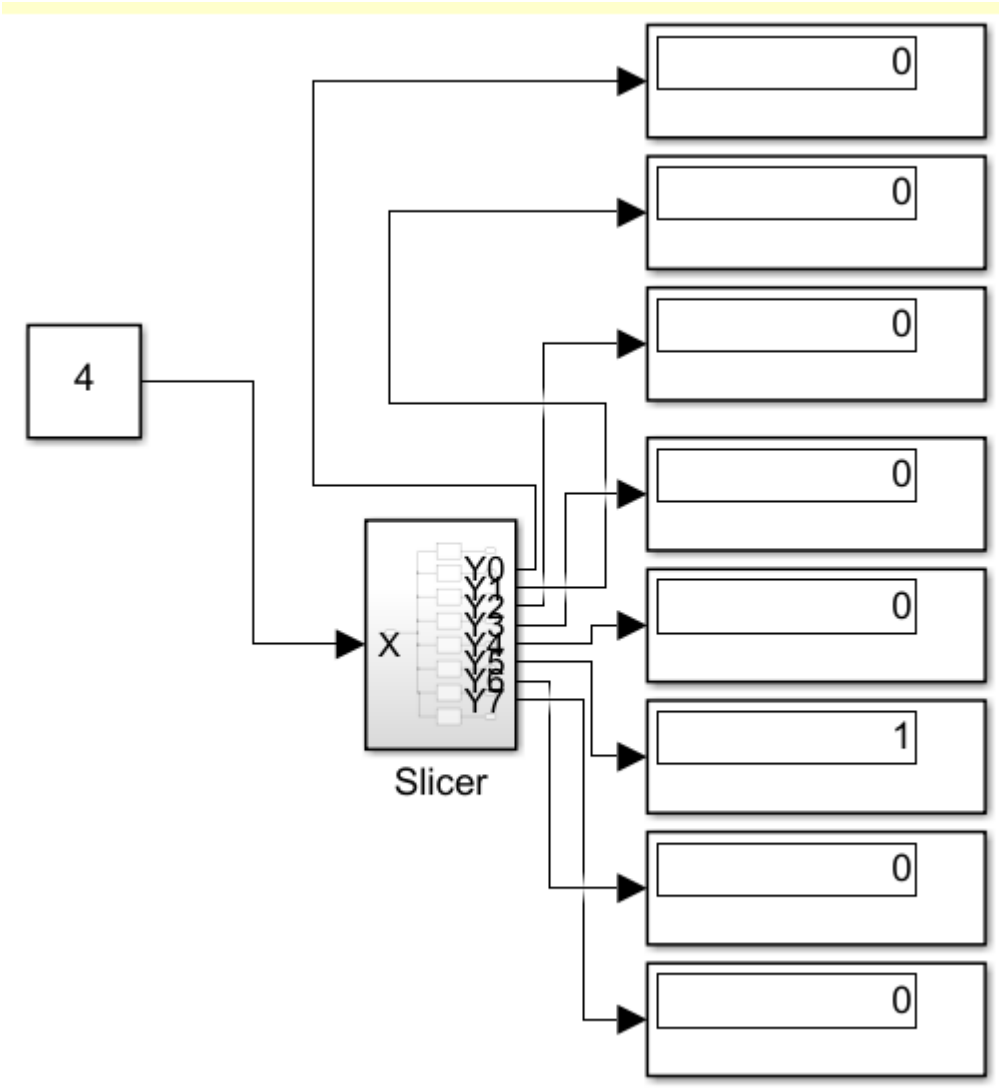
2160 Lab 7 Report

Patrick Taylor and James Wachala

Section Number: 1

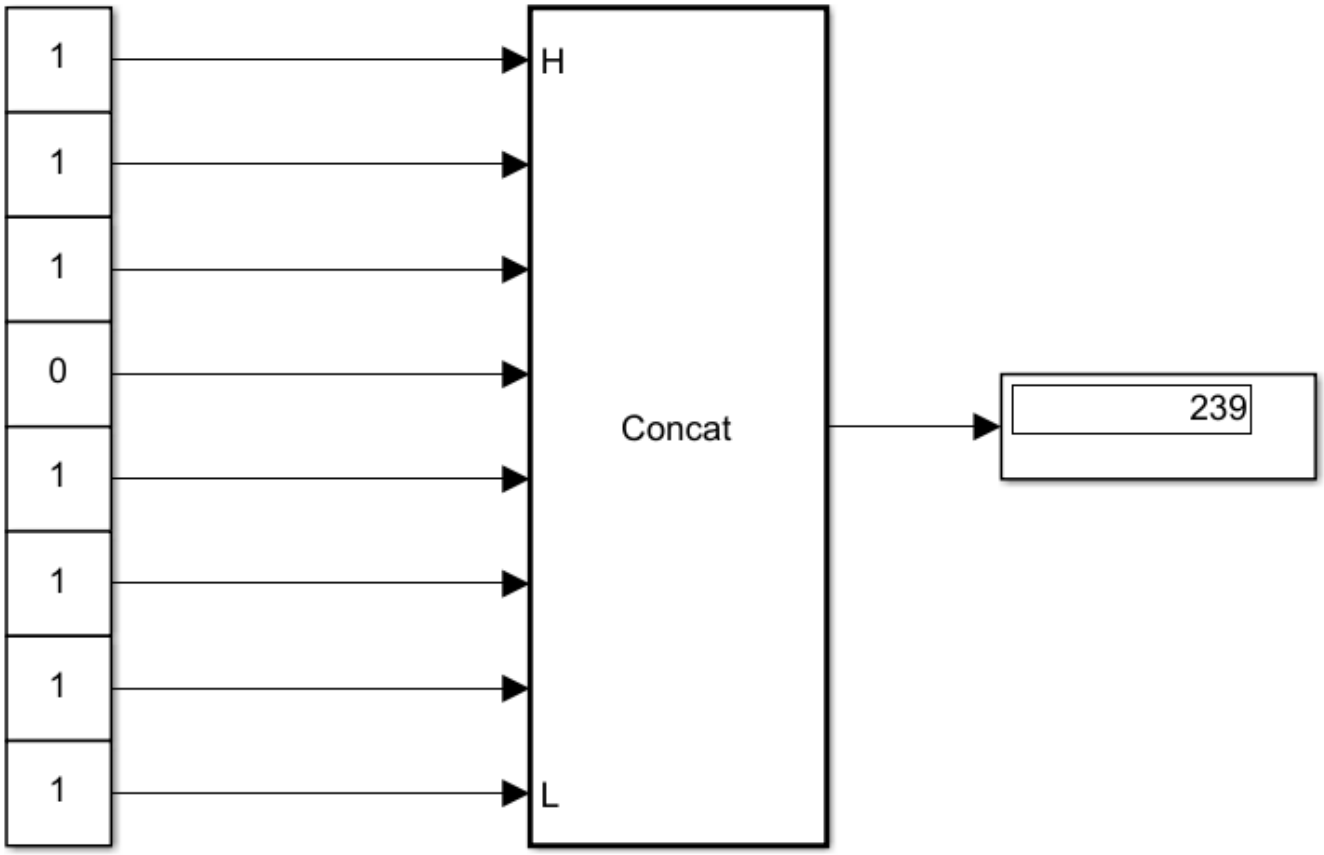
March 13th, 2019

Assignment 1



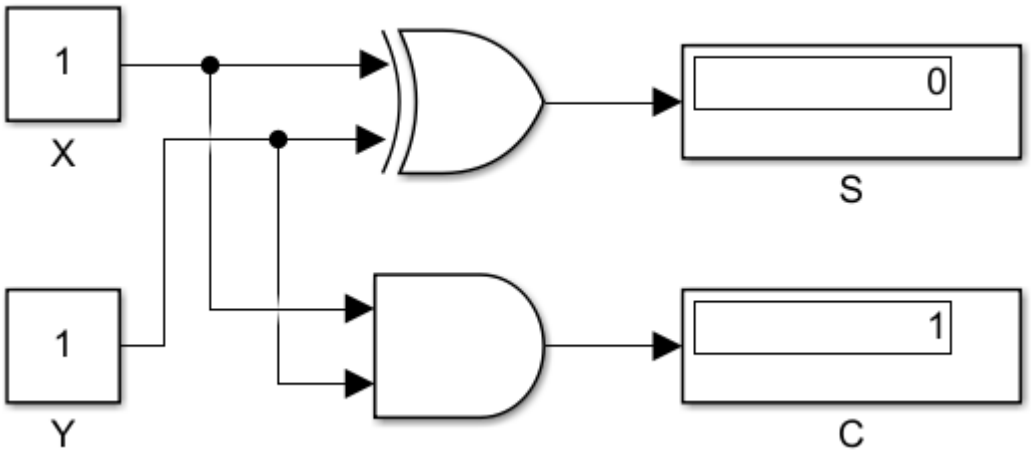
The output displayed here is meant to be a binary representation of the number put into the splicer, split into individual bits. This output is correct, as the binary form of 4 is 100. So if you read from top to bottom, this is that result.

Assignment 2



Displayed here is the opposite behavior of the above, this takes 8 bits and returns a number in decimal form. Each bit is given place value and concatenated to create a binary number, which the output displays in decimal. The displayed is correct as 11101111 in base two is 239 in base ten.

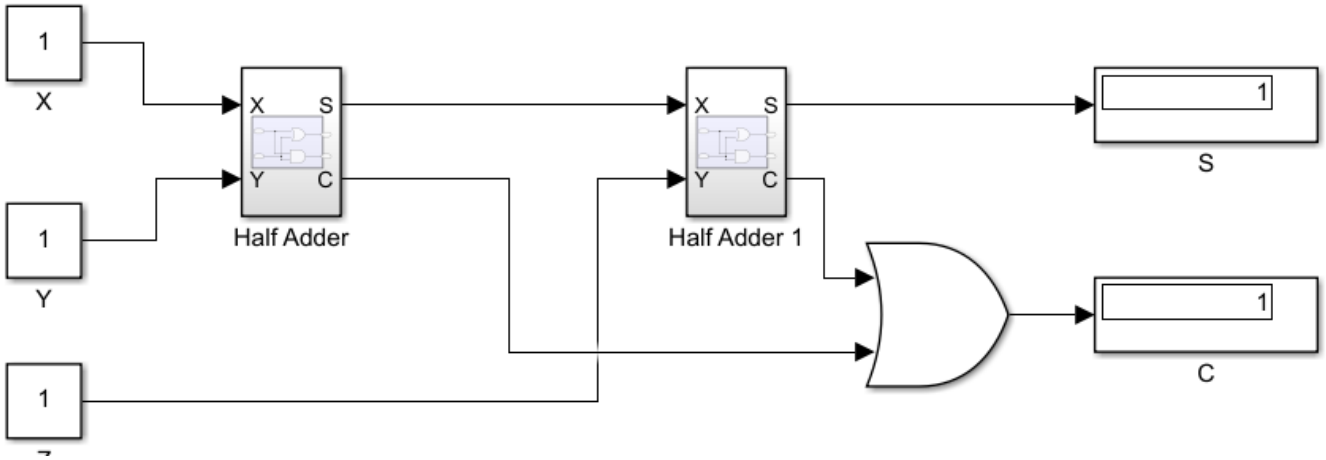
Assignment 3



As s represents the least significant digit and c is the most significant (and carry) digit, the half adder can add 2 bits, as it takes 2 bits as input, and therefore, can have up to two bits as an output. In three cases, the sum is one bit, so is represnted entirely in s. (0 or 1). However, when the sum is two, one bit is not enough. Instead, the two is represented in c, carrying higher place value. In this case, c should be one and s should be zero. To create this behavior, an XOR gate does the job for x and an AND gate does the job for c.

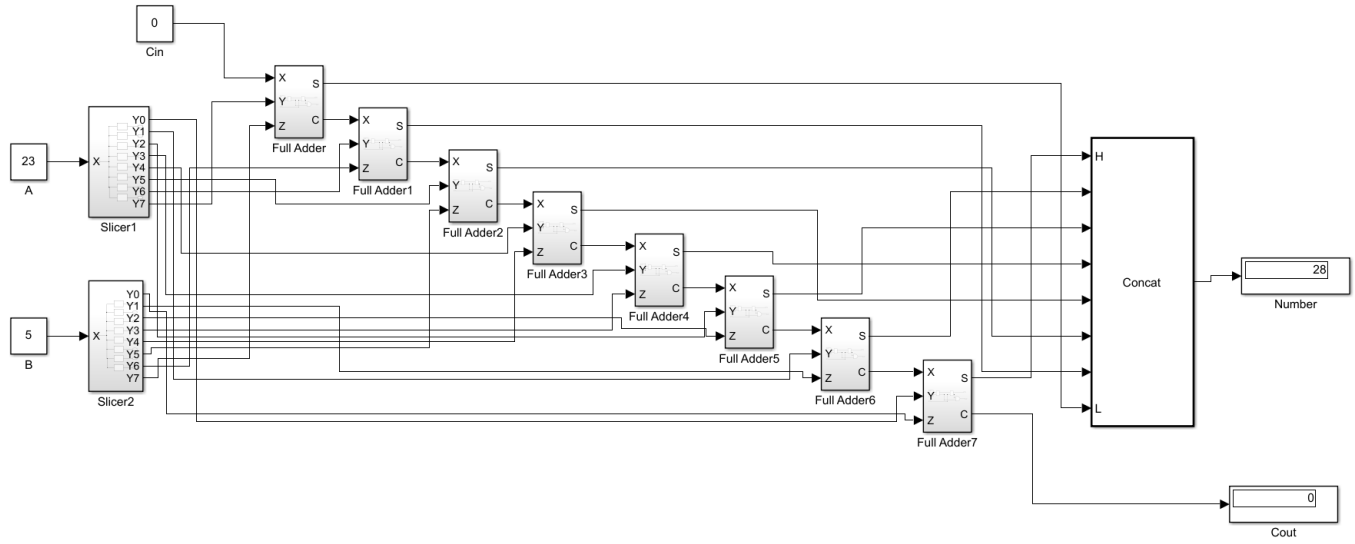
x	y	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Assignment 4



Similar to the half adder, s represents the least significant digit and the most significant digit it c. The full adder utilizes two half adders too add two 1-bit integers together then returning the carryout and adding the least significant bit to the final integer. The OR gate determines if there is a carry digit. If the first half adder returns a carry OR the second returns a carry then the full adder returns a carryout (in the case where the inputs would add to a 2 - `10` or 3 - `11`)

Assignment 5

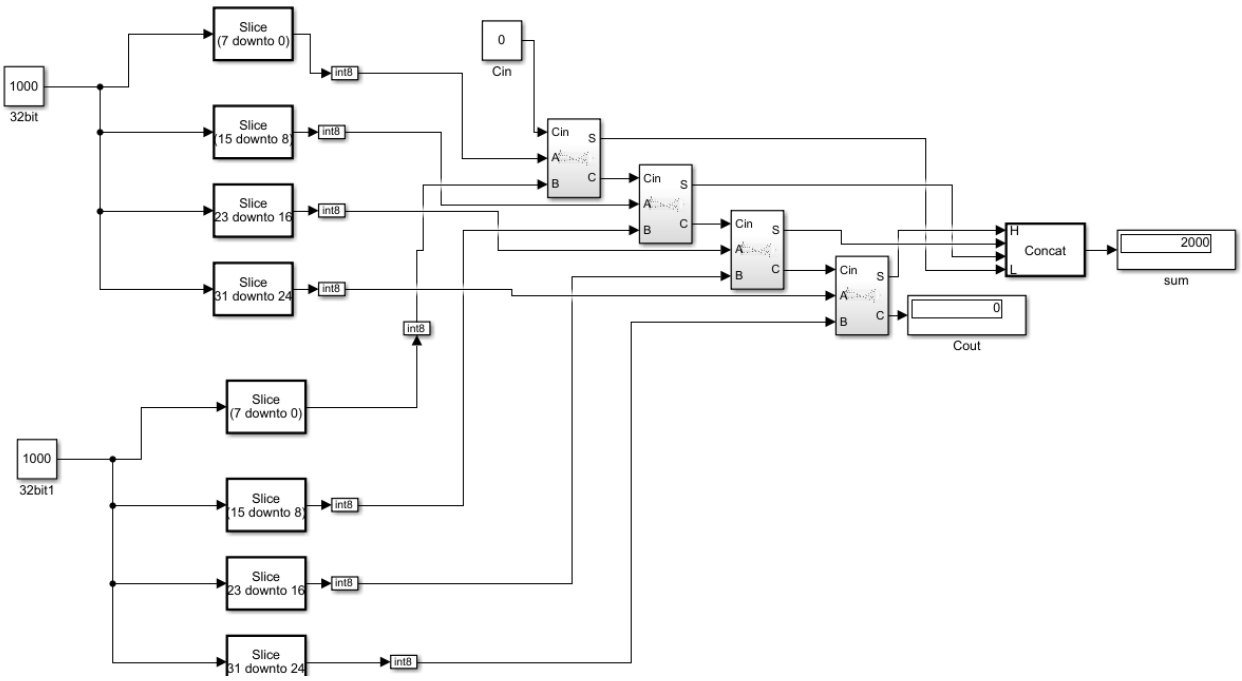


23 + 5 = 28 with no carryout.

An 8 bit slicer cuts the 8 bit number into 8 1-bit numbers and assigns the greatest diget to the last adder and the smallest digeit to the first adder for each of the input numbers at the y and z inputs. the carryout from

each full adder is dumped into the x input of the next full adder, which is the 3rd "input variable." The final Cout returns either a 1 or 0. A 1 is returned if the result is higher than 8-bits (greater than 127). A zero is returned in all other cases.

Extra Credit



The 32-bit adder is very similar to the 8-bit adder. The main difference is instead of the slicer cutting an 8-bit number to 1 bit, it cuts a 32-bit number into 4 8-bit and adds those bits accordingly in the same fashion as the 8-bit adder: Most significant digit at the last adder and least sig digit at the beginning. The biggest issue we ran into was "type" issues, so we used converters to make sure everything remained unsigned 8-bit integers (except for affeter concatinating to our displayed "2000" result).