Final Project
Design of a Simple Sequential Processor

Mohammed Albo Hasabullah
albohasabullah.m@husky.neu.edu
Jonathan Cohen
cohen.jon@husky.neu.edu

Submit date: 4/14/2020
Due date: 4/14/2020

Abstract

This lab is further and the final practice with simulink which is a programming environment in MATLAB which allows modeling and simulation of dynamical systems. This lab will design a simple sequential processor capable of multiplying, adding, and subtracting and logic of bitwise “AND” in the simulink environment.

Introduction

This lab is a continuation to the hardware section of the course. It uses Simulink to create a simple sequential processor which can add, subtract and multiply two 8-bit numbers. Such numbers range from 0 to 255. Note that the output of such processes can only be displayed up to 255 which is the max number representable by an 8-bit number. The lab consists of eight assignments. In order, the assignments are concerned with creating an 8-bit adder, subtractor, multiplier, bitwise "AND" operator, 4-to-1 multiplexer, adding 8-bit functionality to the multiplexer, register which stores the values, and then finally putting the designed pieces together to make the processor.

The lab assumes decent previous knowledge of Simulink and only uses MATLAB-Simulink to simulate the results as there is no access to a ZedBoard at this time.

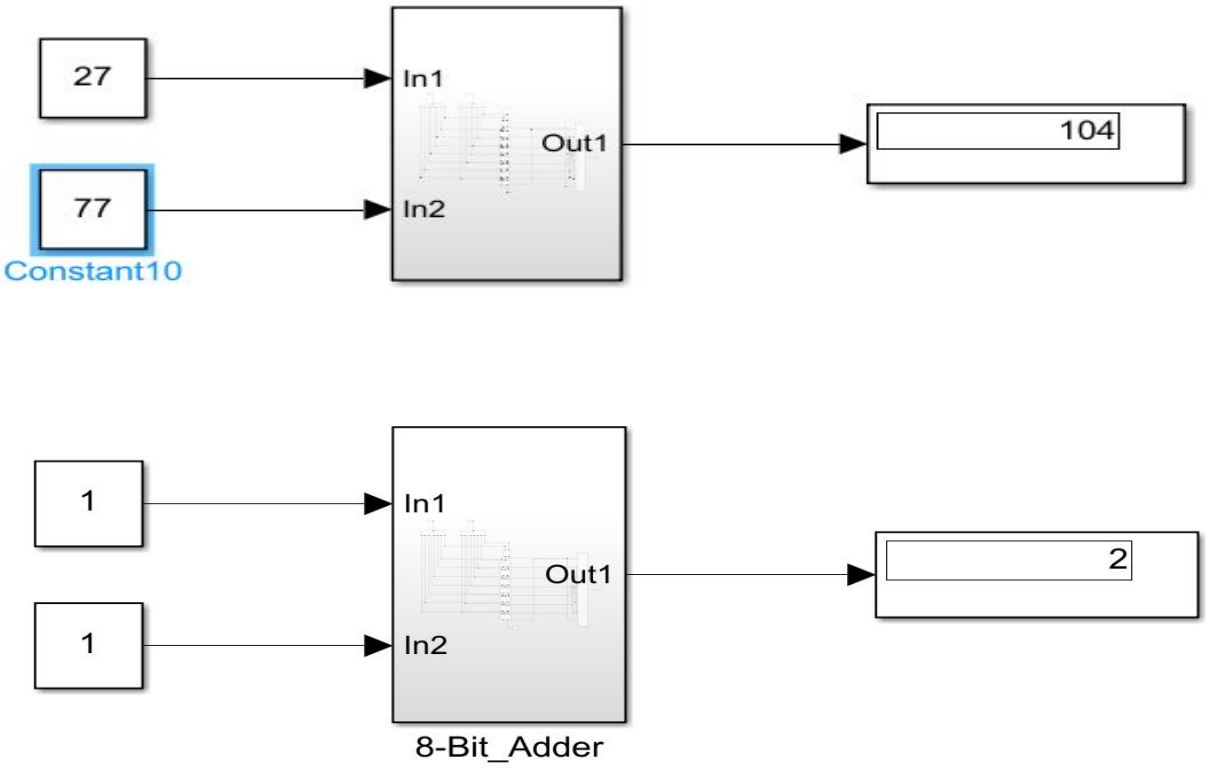
Lab Discussion

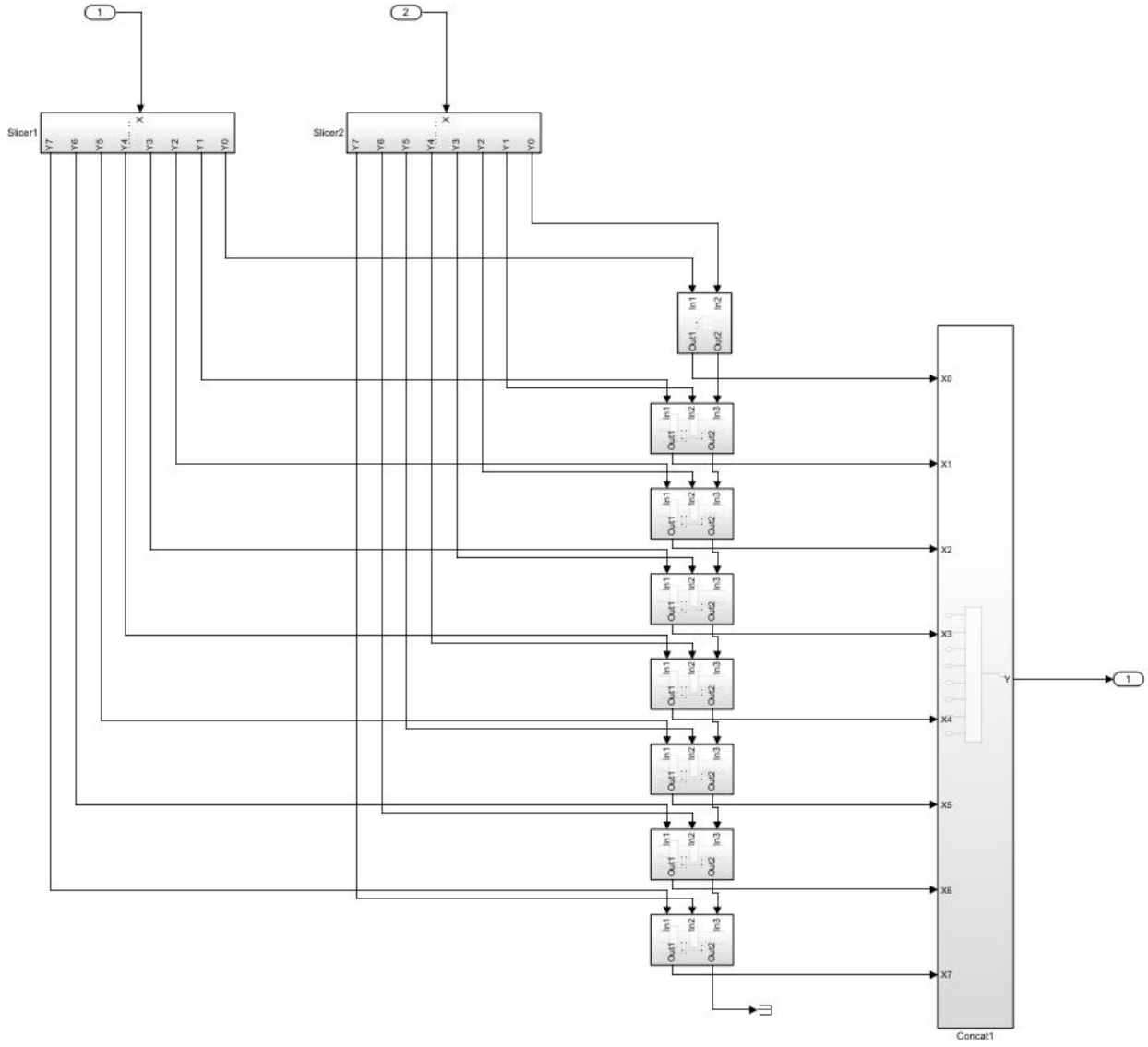
This lab had no pre-lab although it did use previous designs such as the full-adders, half-adders, 8-bit adders…

Results and Analysis

Assignment 1

The first assignment had the students design an 8-bit adder which is what does the addition process. The block takes in two unsigned integers and outputs the sum. The subsystem was passed two 8-bit decimal numbers which were then sent to two seperate slicers. Each pair of equivalent importance bits were passed to an adder. The 0th bit was passed to a half adder while the others were passed to full adders where the third input was the carry over from the previous adder. The sum from the adder was then passed to the concat subsystem which converted the resulting 8-bit binary number back into decimal form. The following pictures show the functionality of the adder and it's subsystem.

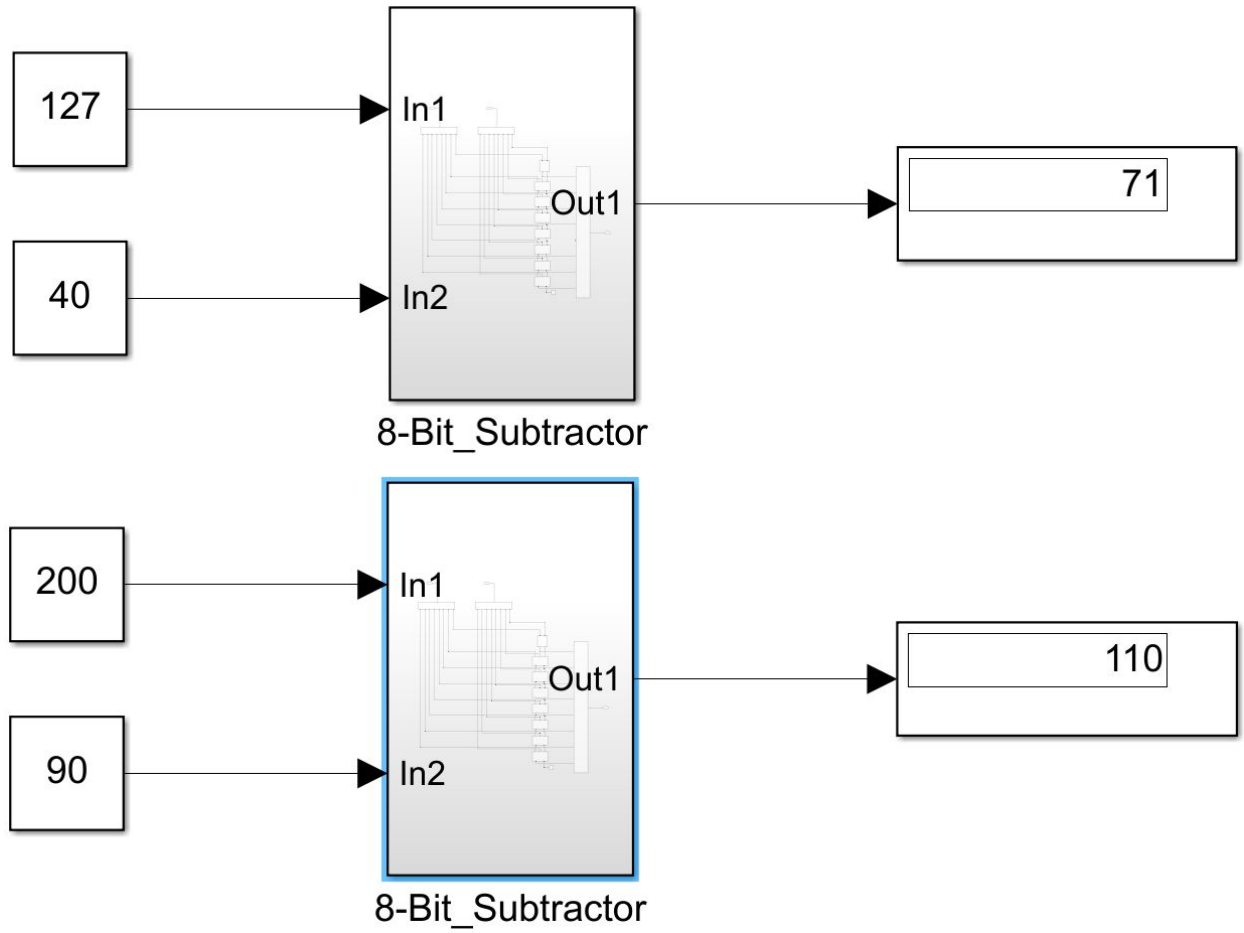




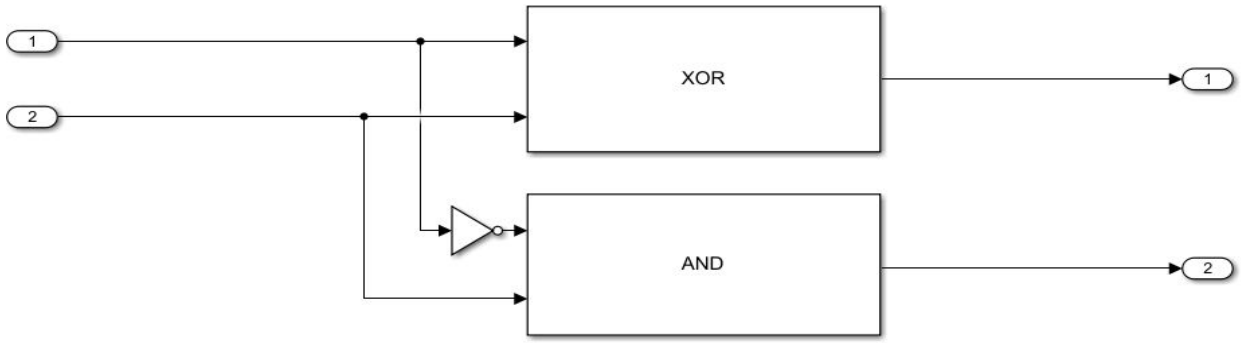
The subsystems take in the 8-bit numbers 1 and 2 which feed into the full adders and the half adder for the least significant bits. Every previous adder feeds into the adder of the next greater significant bit and its corresponding position on the concat. Note that the most significant bit adder terminates its carry over. Finally, the concat puts the bits of the sum together and outputs.

Assignment 2

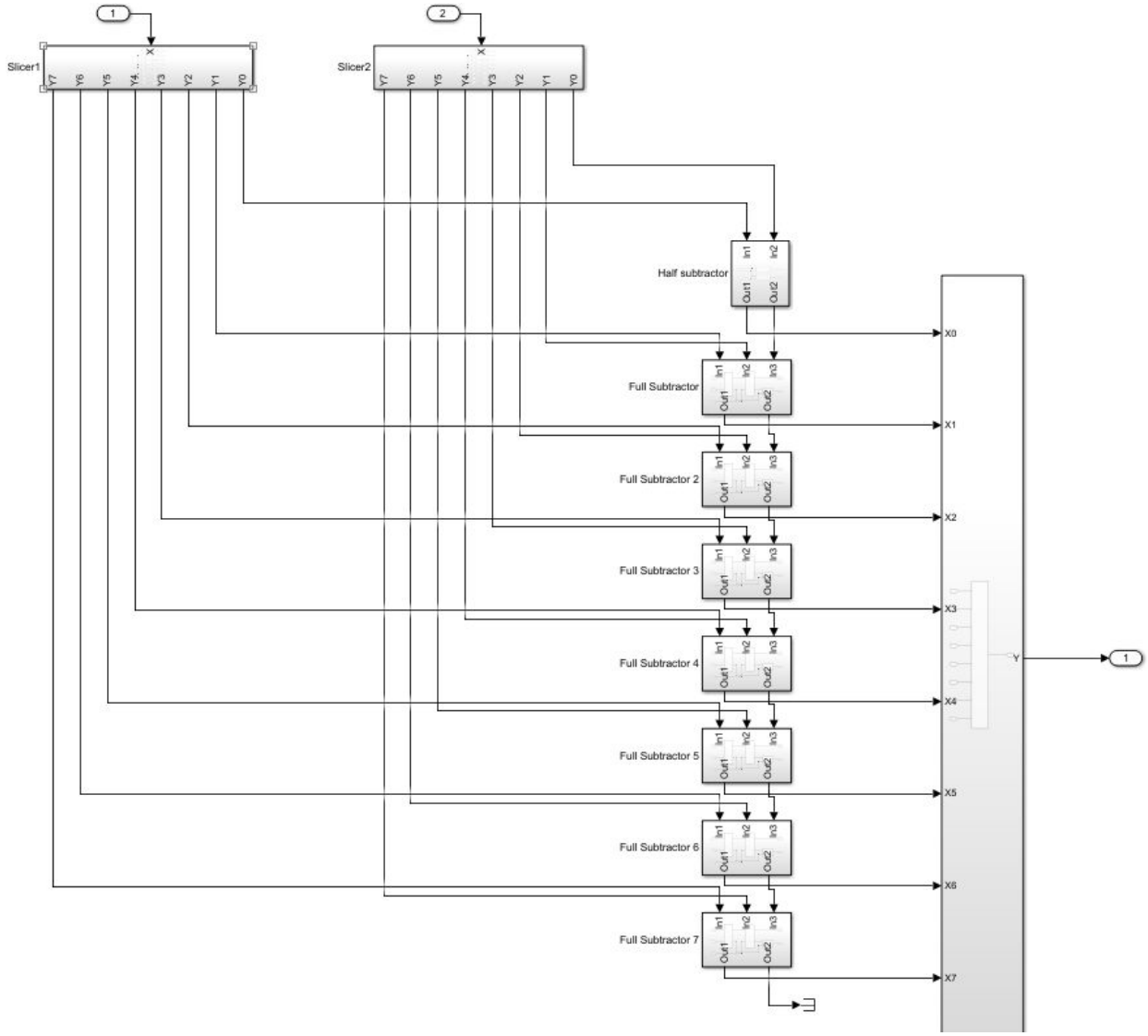
In assignment 2 students modified the adder from assignment 1 into a subtractor.



The main difference between the adder and subtractor is that the subsytem has been changed so that the half and full adders function as subtractors by adding a NOT to the AND gate as follows.

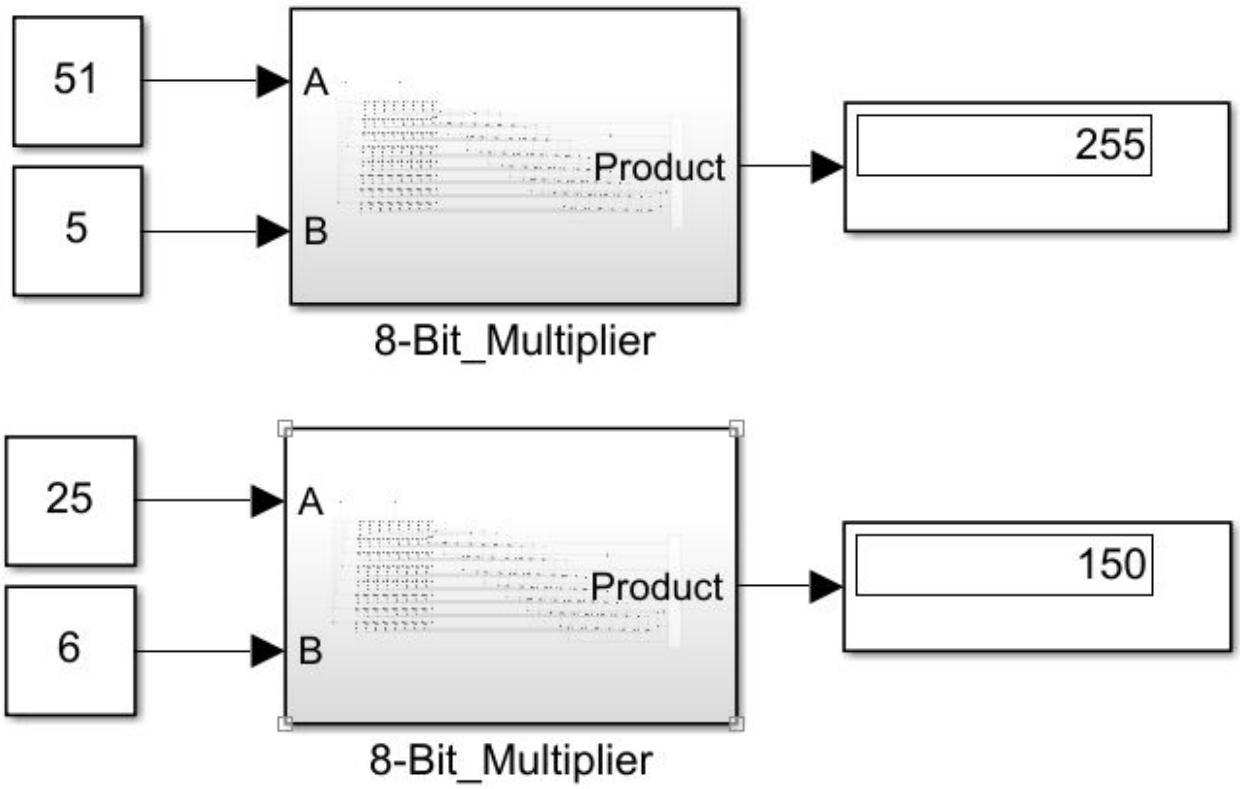


The following shows the subsystem of the 8-bit subtractor with the adders being modified into subtractors.

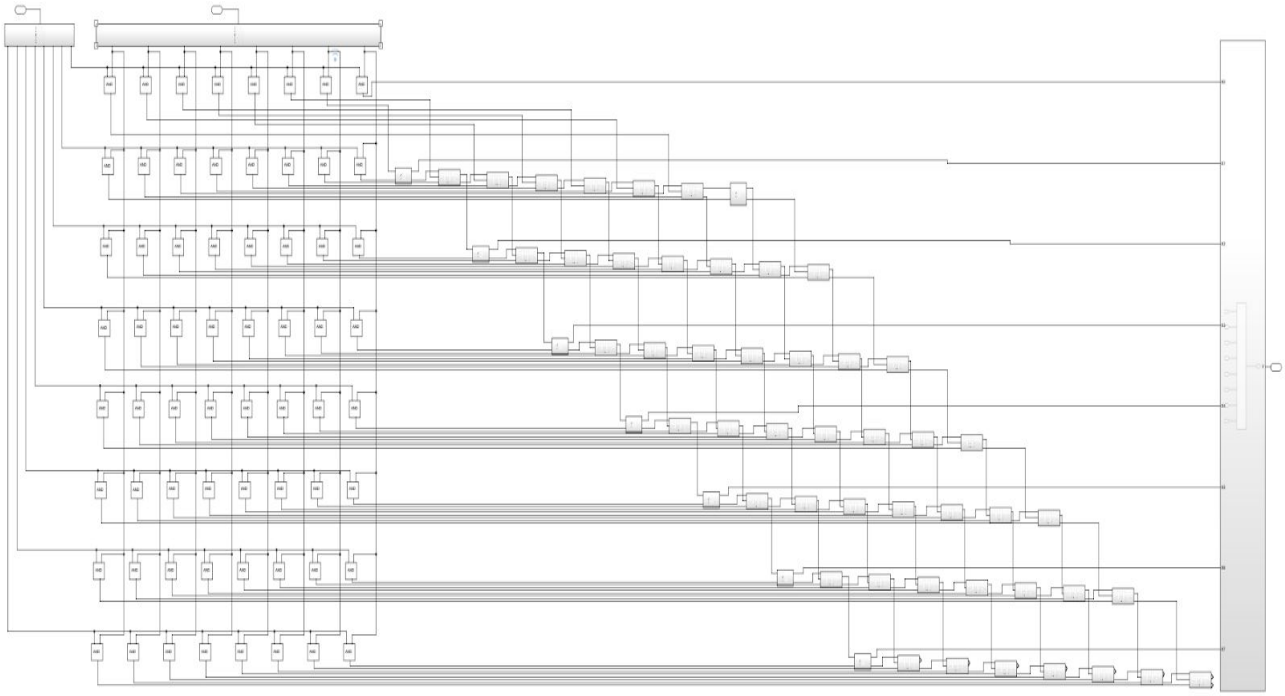


Assignment 3

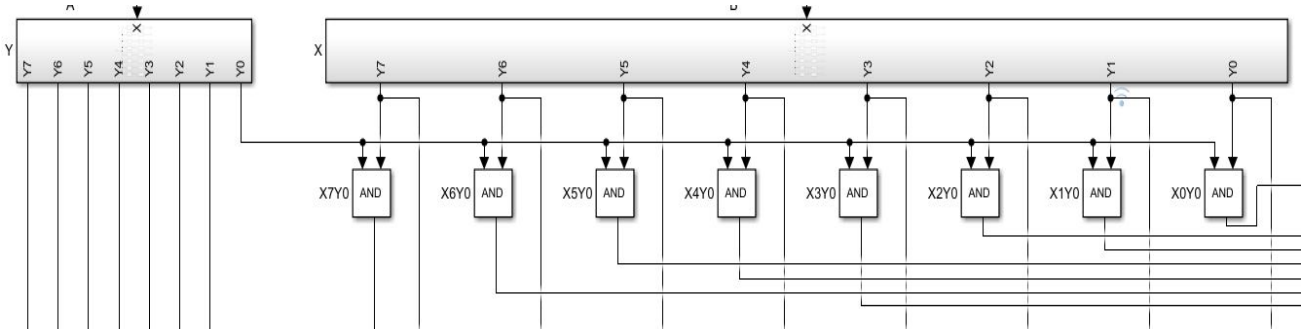
For the third assignment, students created the multiplier design. The multiplier also takes in two unsigned 8-bit numbers and then outputs the product of the two. Note that if the product is more than 8-bit, the multiplier will disregard any bits more significant than the 8th bit. The following shows the function of the multiplier.



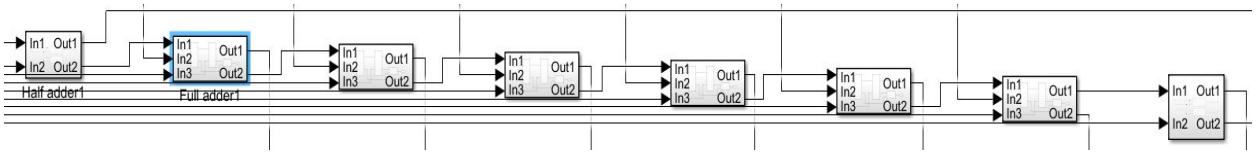
The following is the subsystem of the multiplier.



The multiplier works by multiplying the sliced 8-bit numbers as if it is in a matrix. This can be seen with the following, which is the very first of the multiplications



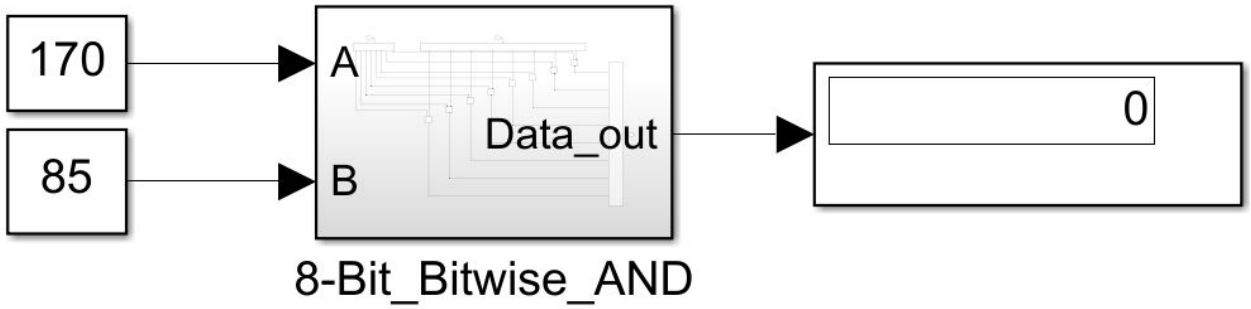
The least significant bit of Y is multiplied with every bit of X. Then the multiplied bits using AND gates are fed into Half and Full Adders as can be seen here until the last bit of Y.



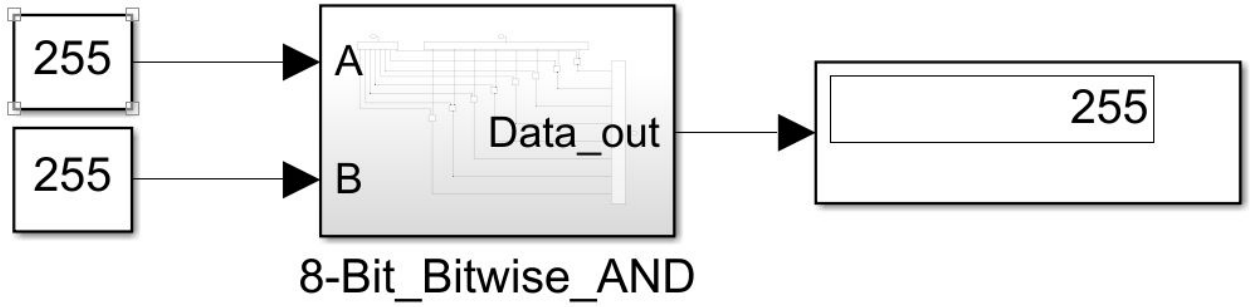
The outputs of the adders are then fed into the next adders in the row below in the matrix and into the concat which puts our product in decimal form.

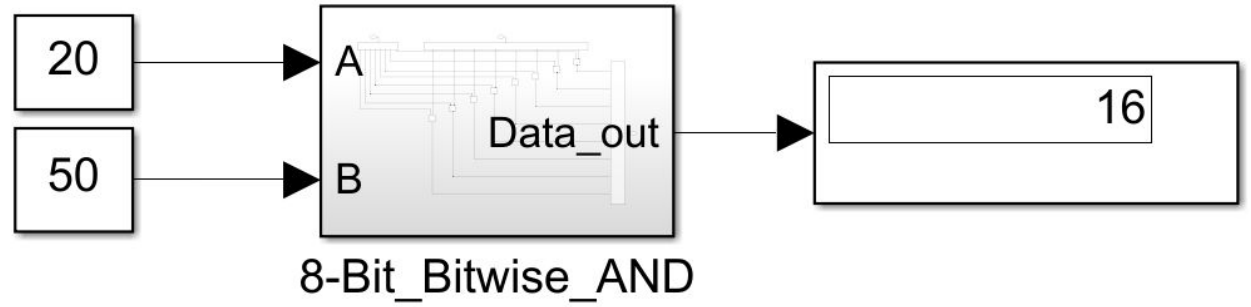
Assignment 4:

Assignment 4 had the students create a bitwise "AND" operator. The block can take in two unsigned 8-bit numbers and outputs a new number which has a binary representation of copied similar bits in the binary representation of the two inputs. Take for example 170(10101010 in binary) and 85(01010101 in binary) which have no similar bits in the same position, resulting in a new number 0(00000000 in binary).

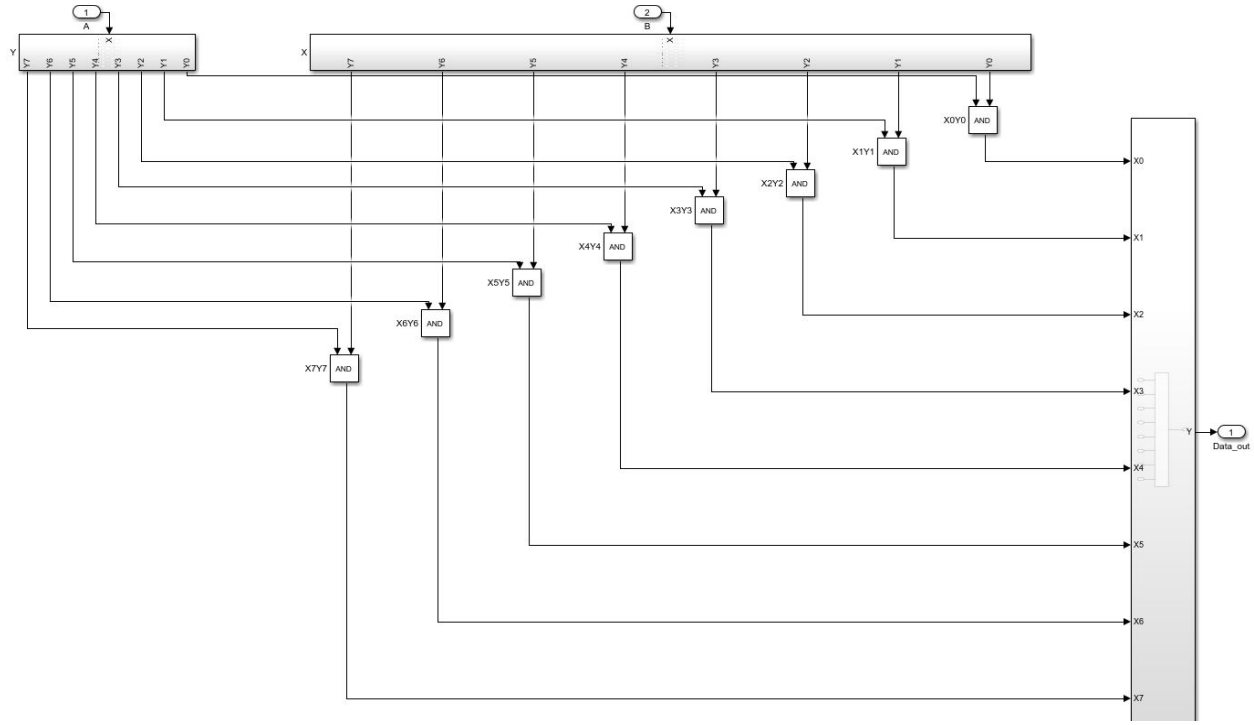


While when two similar numbers are the input, the output is the same number since all similar bits are copied.



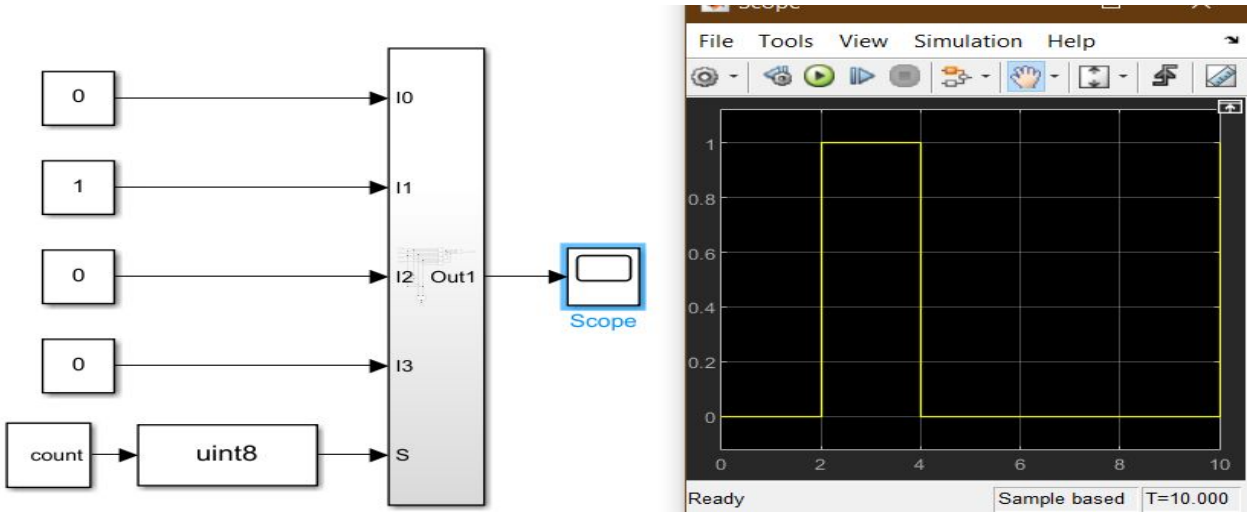


The following shows the subsystem of the operator. Every bit in Y is compared with the parallel bit in X, if the bits are the same, the AND gate outputs the similar bit, if they are different, it outputs a 0. As always, the bits are fed into a concat which puts the final number in decimal form

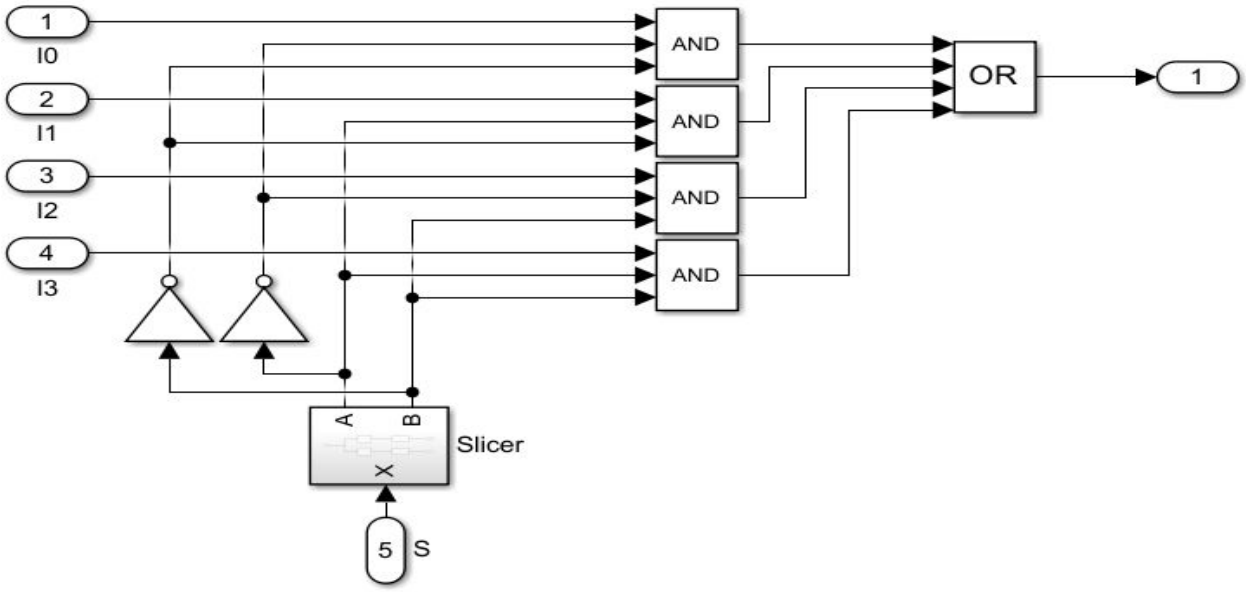


Assignment 5:

This assignment had students create a 1-bit 4-to-1 multiplexer which serves as the switch of the processor. It basically routes the result of the desired arithmetic operation to the final output. The multiplexer takes in the four data inputs of the addition, subtraction, multiplication, and bitwise "AND" operation in addition to a switch 2-bit value from 0-3 which each represent an operation. The following shows the functionality of the mux coder as the selector is fed through a counter which will change its value over time, since the counter will start at 0, the selector will read the value being inputted into the I0 port, then the selector will count up 1 and read the value at I1 which is 1. Then 2, then 3 which read 0.



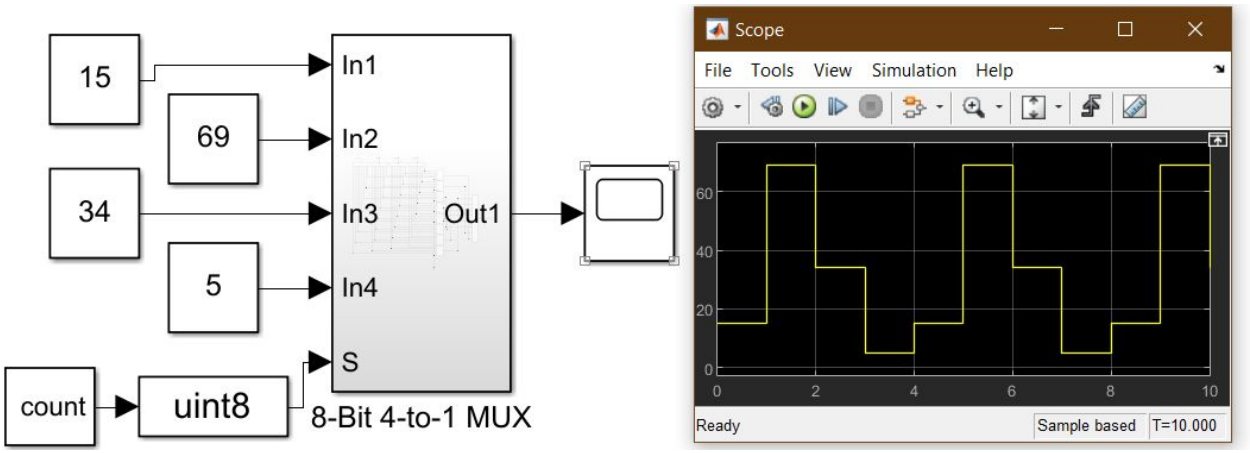
The following shows the subsystem of the 1-bit multiplexer.



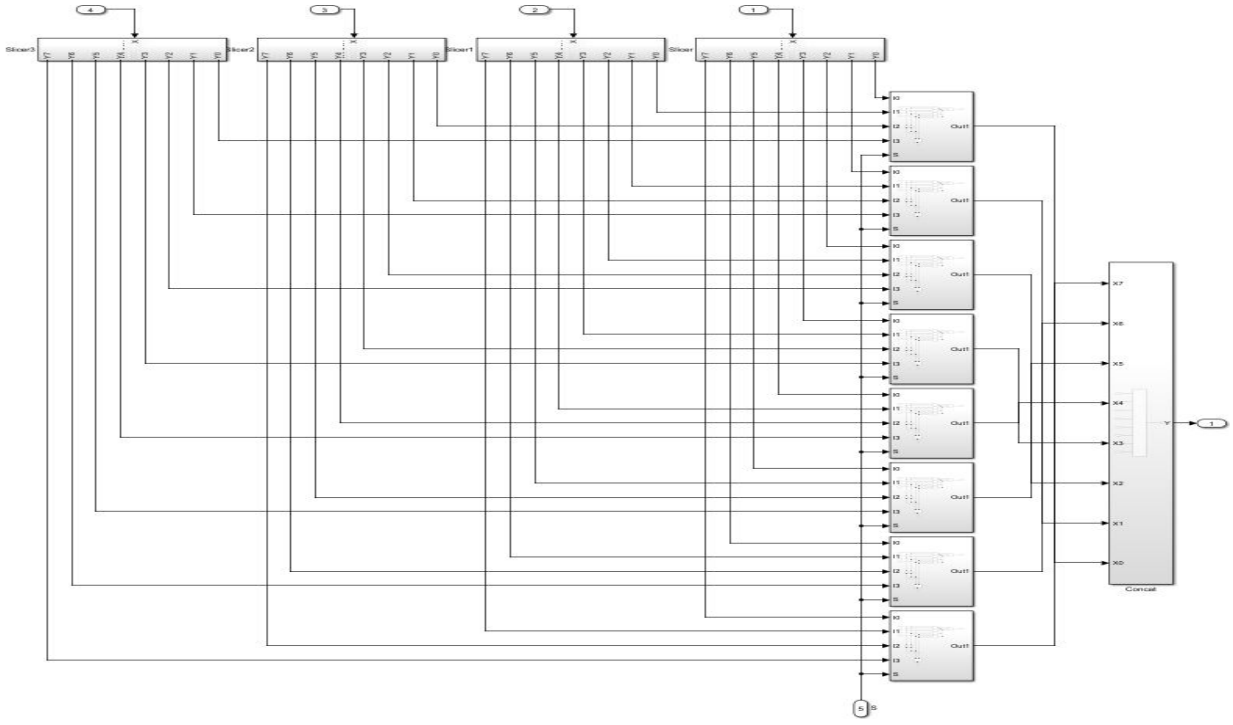
The subsystem uses AND gates in order to route the desired output. The selector slices the 2-bit number it receives and feeds it into the AND gates which will in turn give the desired output of the desired operation.

Assignment 6:

This assignment had the students modify the multiplexer into an 8-bit multiplexer which can take in 8-bit data inputs. The following shows a simulation of the multiplexer being run with a counter in order to show the different output values with a variable selector value.

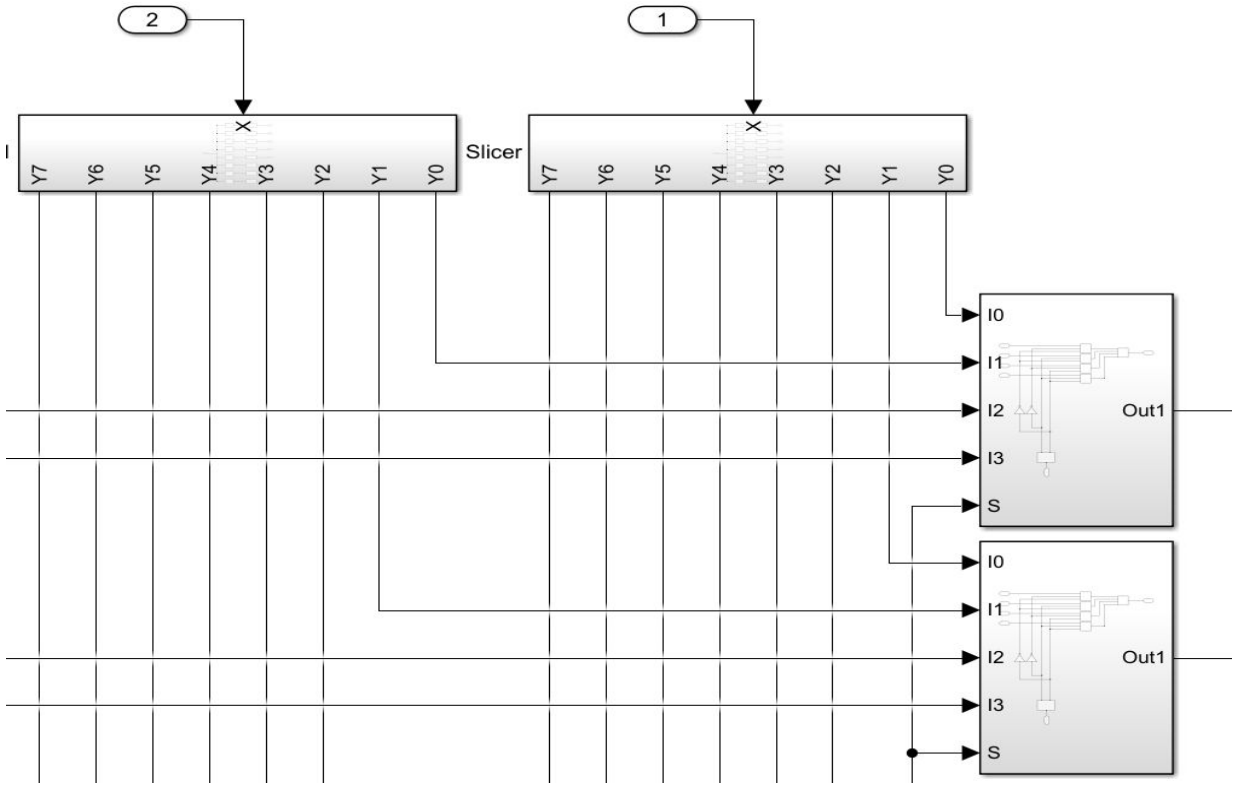


The following shows the subsystem of the multiplexer.



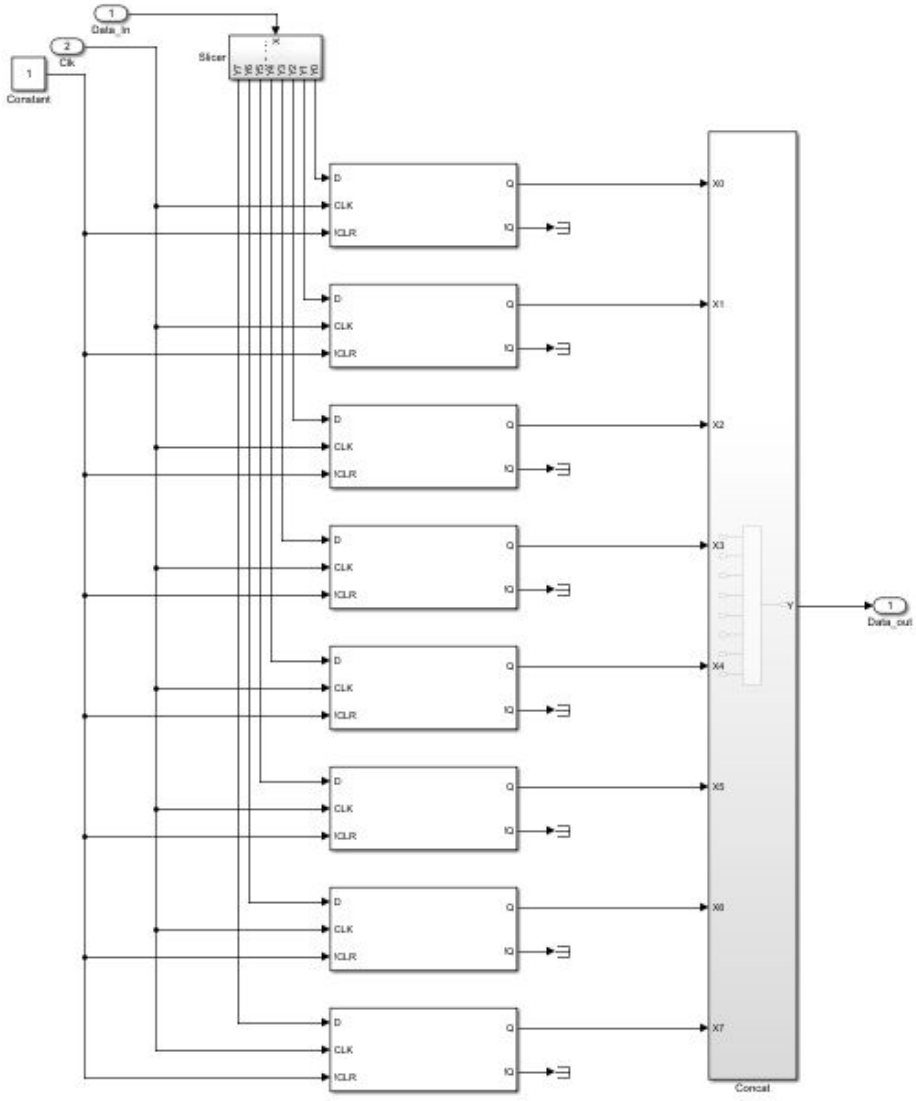
The 8-bit slicers at the top feed into the 1-bit slicers created in the previous assignment as follows. There are eight 1-bit multiplexers inside, each responsible for an nth bit from the 8-bit

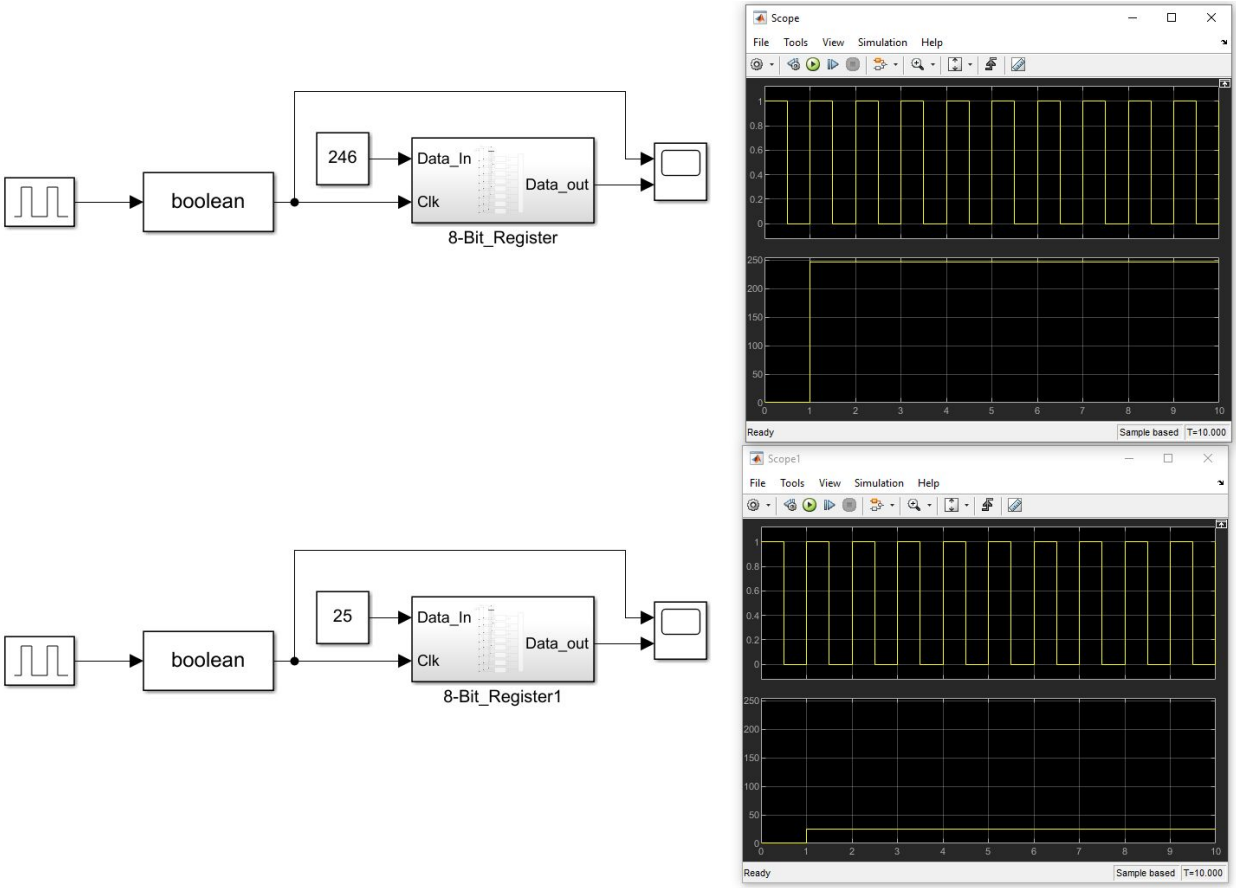
inputs. Then the 1-bit multiplexer acting together sends out the appropriate or desired bits, serving as constructors and deconstructors of the 8-bit multiplexer.



Assignment 7:

This assignment had the students create an 8-Bit Register as a way to save the value of an 8-Bit unsigned integer so that it would persist over multiple cycles until instructed to overwrite the value. An array of 8 1-Bit flip-flops is used to accomplish this. Each of the flip-flops stores an nth bit from the 8-bit unsigned integer input. The array of flip-flops also all share a 1 period clock signal in order to keep them in synchronization with one another. The output from the array of flip-flops and converted into an 8-Bit integer again by a concat, by default this will be 0.

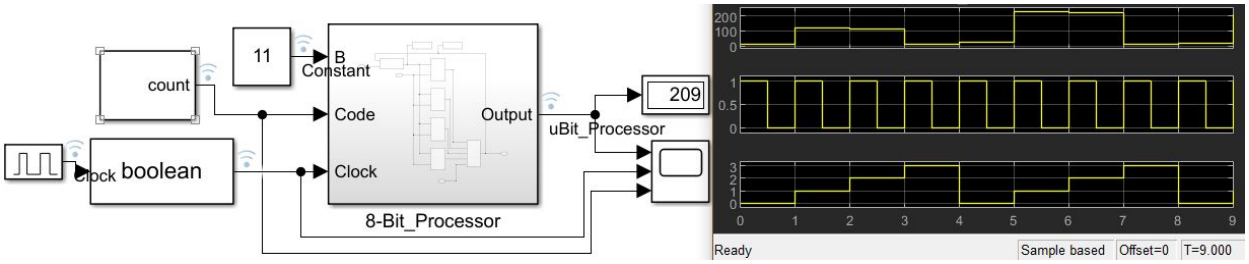




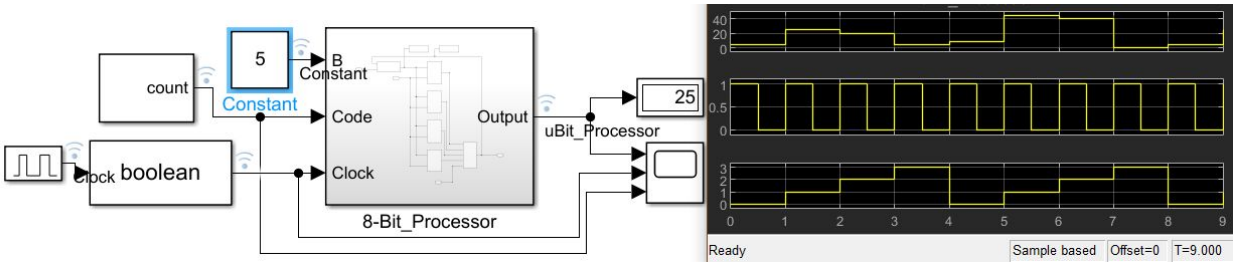
From the output scopes of the 8-Bit Registers it is apparent that the registers are behaving in the intended manner. The output value of the register remains zero for the first clock cycle until a new value is stored on the second cycle. This value equal to the constant input persists for the remainder of the cycles performed.

Assignment 8:

Finally, this assignment had the students put together all of the previously designed circuits in order to create the bigger picture, which is the simple sequential processor. The processor can take in three inputs, the first is the 8-bit number referenced as B which will be second in the operations, a 2-bit number which is the code for the desired operation, and then a 1-bit number for the clock representing the rising and falling edge. Codes 00, 01, 10, 11 are addition, multiplication, subtraction, and bitwise "AND" respectively. The following shows a simulation with B = 11.



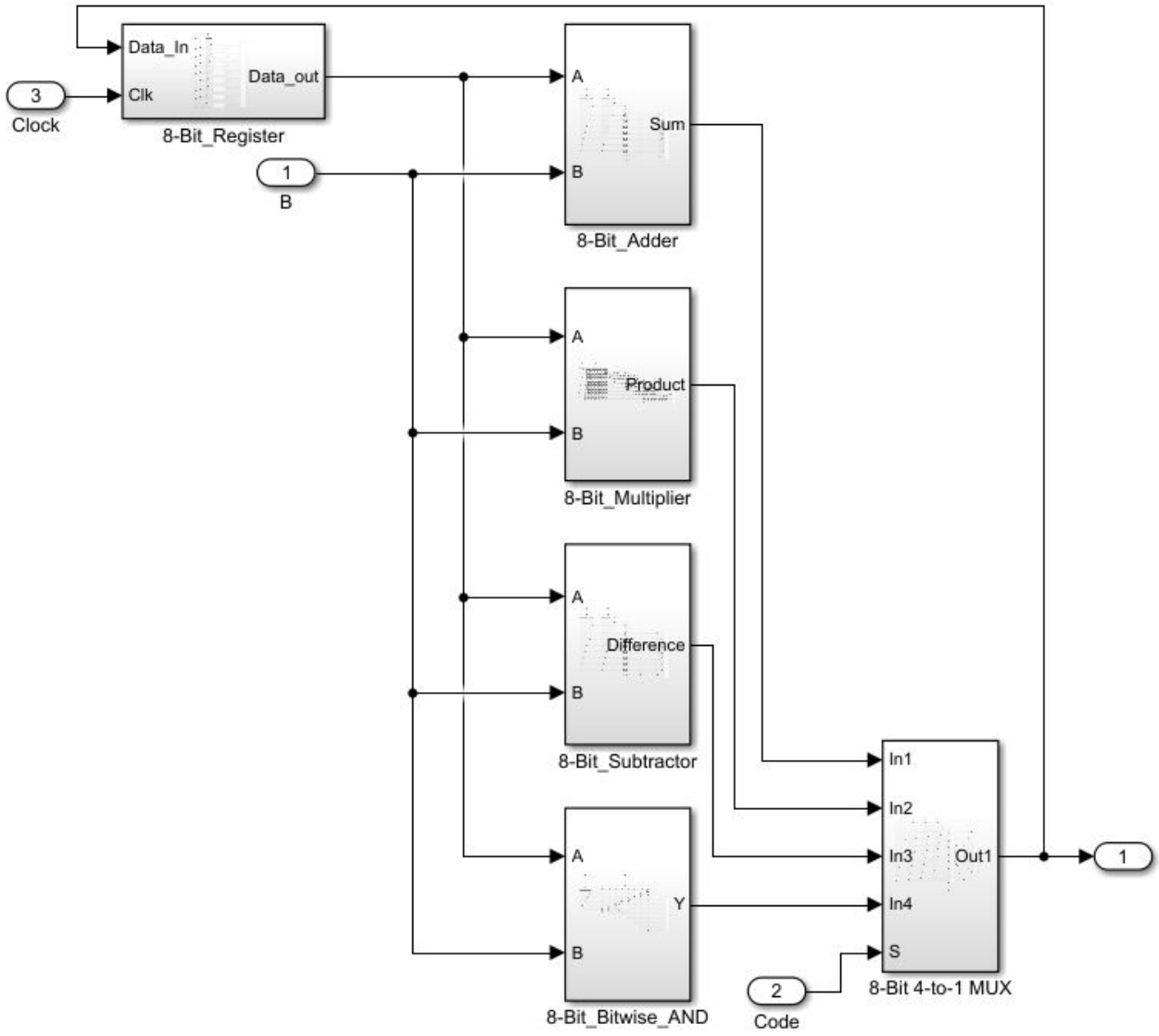
As can be seen on the scope display, the first from the top is the value at the specific time, then the second in the middle shows the rising and falling edge of the clock, and finally the bottom one shows the changing selector value which goes from 0-3 then 0-3 for a total of 10 operation cycles. An initial B = 11 will result in a final output of 209.



The following table shows the step-by-step calculations for B = 11 and B = 5 which verifies that the 8-bit processor works well.

00	Adder	$0 + 11 = \mathbf{11}$	$0 + 5 = \mathbf{5}$
01	Subtractor	$11 * 11 = \mathbf{121}$	$5 * 5 = \mathbf{25}$
10	Multiplier	$121 - 11 = \mathbf{110}$	$25 - 5 = \mathbf{20}$
11	Bitwise AND	$110 \,\&\, 11 = \mathbf{10}$	$20 \,\&\, 5 = \mathbf{4}$
00	Adder	$10 + 11 = \mathbf{21}$	$4 + 5 = \mathbf{9}$
01	Subtractor	$21 * 11 = \mathbf{231}$	$9 * 5 = \mathbf{45}$
10	Multiplier	$231 - 11 = \mathbf{220}$	$45 - 5 = \mathbf{40}$
11	Bitwise AND	$220 \,\&\, 11 = \mathbf{8}$	$40 \,\&\, 5 = \mathbf{0}$
00	Adder	$8 + 11 = \mathbf{19}$	$0 + 5 = \mathbf{5}$
01	Subtractor	$19 * 11 = \mathbf{209}$	$5 * 5 = \mathbf{25}$

The following shows the subsystem layout of the 8-Bit processor and the components it consists of. In this circuit an unsigned 8-Bit integer is passed in alongside a one period clock signal and a 2-Bit operation code. The 8-Bit integer is passed to every possible operation with the value stored in an 8-Bit register. All four operational blocks output to the 8-Bit 4-to-1 MUX, which then, depending on the operation code it was passed, chooses which operation to output and store into the register.



Conclusion

The work done in this lab further expands the students knowledge of Simulink, a useful programming environment widely used for modeling of circuits, digital signal processing and various other applications. The lab contained eight assignments. The assignments are concerned with creating an 8-bit adder, subtractor, multiplier, bitwise "AND" operator, 4-to-1 multiplexer, adding 8-bit functionality to the multiplexer, a register and then finally putting the design all together in order to create the students' first sequential processor. The project served as a great final piece in order to sum the knowledge learned in the lectures that focused on logic and was a great showcase of how much is possible through Simulink.

Simulink will be a great tool for computer and electrical engineers and will help simulate and model the circuits they will come across and design. This lab further expands the students knowledge of simulink and provides them with further practice in hardware and the world of processors and circuits

Distribution of Duties:

Task	Mohammed	Jonathan
8-Bit Adder	**X**	
8-Bit Subtractor	**X**	
8-Bit Multiplier		**X**
8-bit Bitwise "AND"	**X**	
4-to-1 Mux	**X**	
8-bit 4-to-1 MUX		**X**
8-Bit Register		**X**
8-Bit Processor		**X**
Lab Report	**X**	**X**