LAB 2: 3-Bit Binary Sign-Extended Adder/Subtractor

ELEC2607 - LOE, Mon 2:35pm – 5:25 pm

Saturday, February 22nd, 2020

Youssef Ibrahim
Abdalrahman Shaheen

1.0 INTRODUCTION

The purpose of this lab is to create a circuit that finds the sum of two 3-bit two's-complement binary numbers. The circuit will output the sum of both numbers, and indicate if there is an overflow ton confirm if the answer is correct or no. The circuit does this by sign-extending both binary numbers and comparing the sign of the most significant bit. The circuit performs subtraction by finding the two's complement of the number to be subtracted and then adding both numbers afterwards. Using the two's complement made it possible to use the same circuit for addition and subtraction of binary numbers. The circuit was built using a software called Xilinx ISE. The following reports shows the Specifications in part 2.0, Design in part 3.0, Implementation and Testing in part 4.0, and the Conclusion in part 5.0 of the telephone switch circuit.

2.0 SPECIFICATIONS

The circuit was designed to receive two 3-bit two's complement number and output a their sum which is a sign extended 4-bit binary number. There are seven inputs, 3-bits for each number and a carry input to indicate whether the circuit is adding or subtracting. An additional output bit is used to indicate the presence of a 3-bit overflow (ovf), ovf = 1 means overflow and ovf=0 means no overflow. The two 3-bit two's complement binary numbers are represented by X_2,X_1,X_0 and Y_2,Y_1,Y_0. The carry bit is represented by C_0, where C_0= 0 means subtraction and C_0=1 means addition. The results are shown on five output lines, where the sum is being represented by $S=S_3S_2S_1S_0$, S_3 is the sign bit of the number where S_3=0 means positive and S_3=1 means negative.

The design is simulated using Xilinx ISE, the inputs are set by running the simulation using ModelSim. This lab was not limited to any number or type of gates.

3.0 DESIGN

The first part of designing the 3-bit binary sign-extend adder/subtractor circuit is shown below in Figure 3.0.

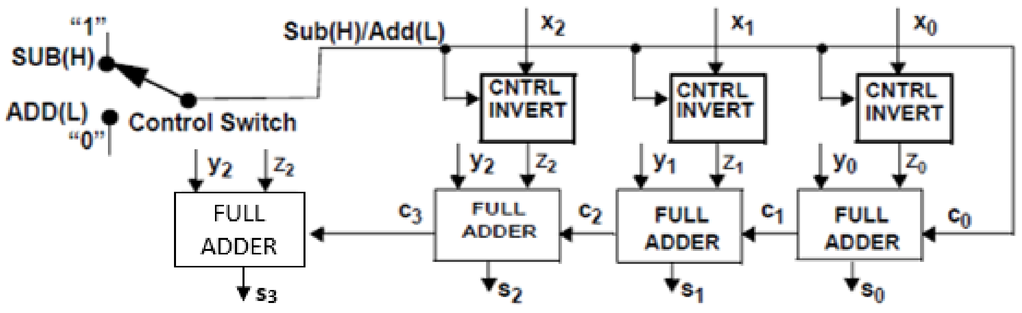


Figure 3.0: 3-bit binary sign-extend adder/subtractor circuit [Lab 2 – p7, modified]

The circuit above consists of three control inverters, four full adders, and a control Switch.

3.1 Control Inverter

Figure 3.1 below shows a controlled inverter circuit.

Sub(H)/Add(L)	X_i	Z_i
1	0	1
1	1	0
0	0	0
0	1	1

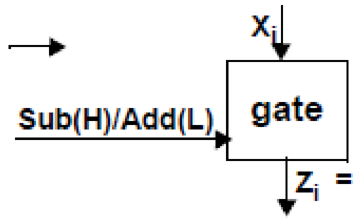


Figure 3.1: Control Inverter [Lab 2 – p7, modified]

The gate used in the circuit shown in Figure 3.1 above is an X-OR gate. This circuit will invert the input X_i whenever the switch is at Sub(H). The output Z_i depends on the value of the input X_i and Sub(H)/Add(L), if Sub(H)/Add(L) = 0 this means that the switch is at ADD(L) then $Z_i = X_i$, if Sub(H)/Add(L) = 1 this means that the switch is at SUB(H) then $Z_i = \overline{X_i}$. A NOT gate would not work effectively in this circuit.

3.2 Full Adder

Figure 3.2 below shows a Full Adder that is used in building the final circuit.

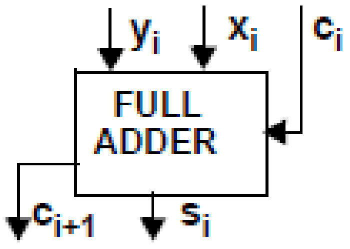
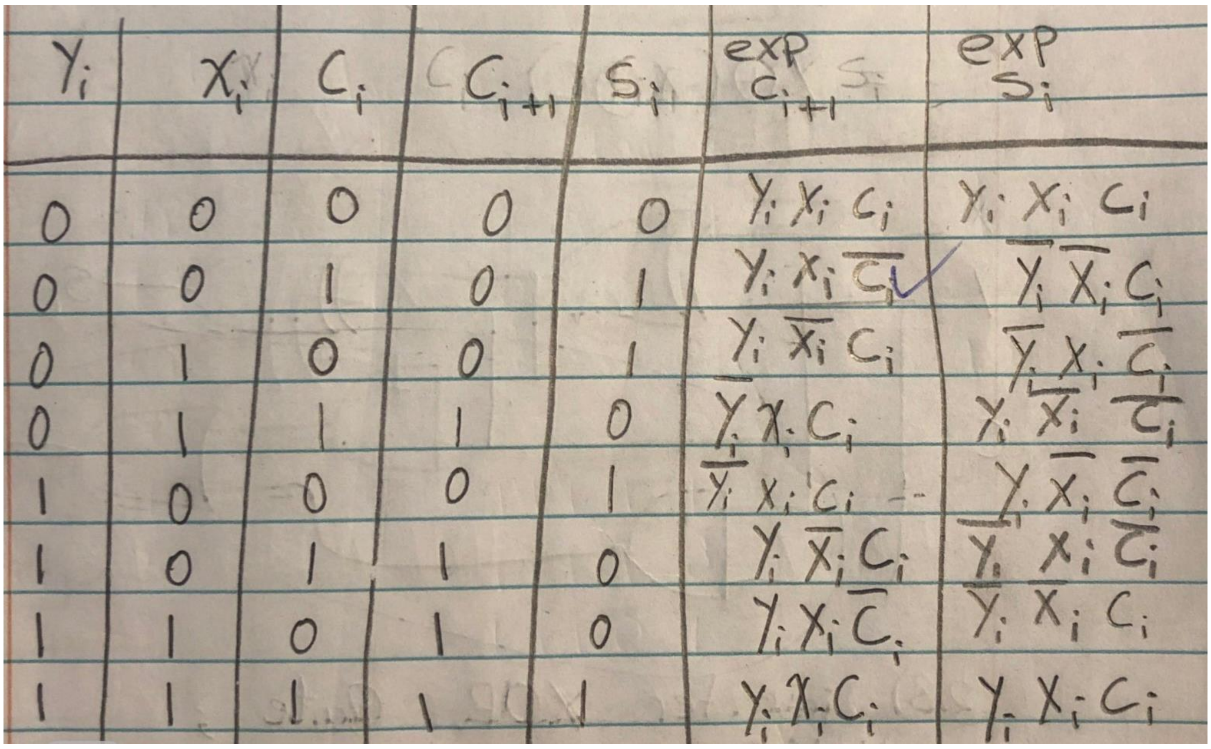


Figure 3.2: Full Adder [Lab 2 – p6, modified]

Figure 3.2 above shows a full adder that adds three binary numbers as input Y_i, X_i, C_i and outputs two 1-bit binary numbers, a sum S_i and a carry C_{i+1}.

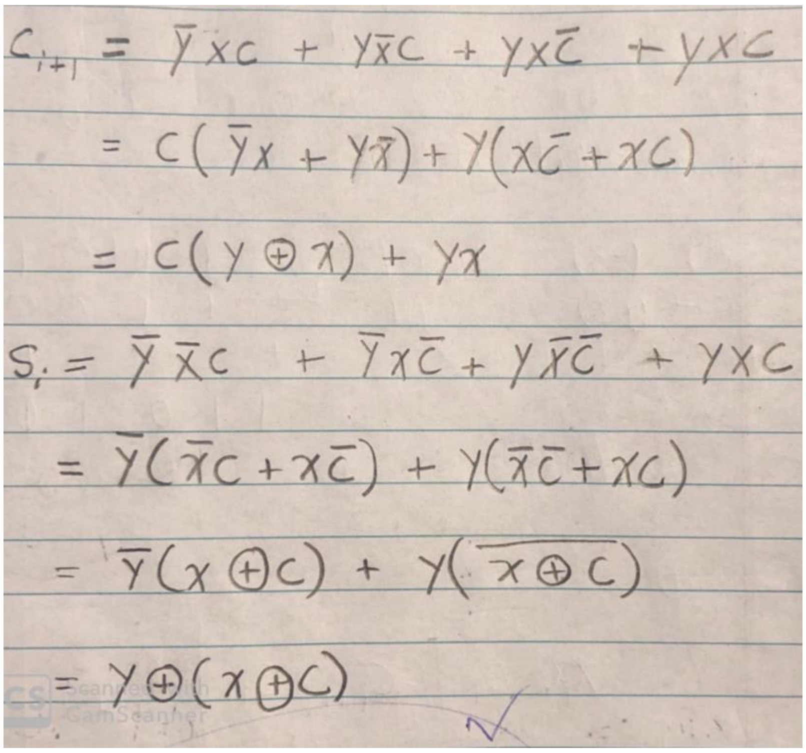
Figure 3.3 below shows the partial truth table of a generic full adder.



Y_i	X_i	C_i	C_{i+1}	S_i	exp C_{i+1}	exp S_i
0	0	0	0	0	$Y_i X_i C_i$	$Y_i X_i C_i$
0	0	1	0	1	$Y_i X_i \overline{C_i}$ ✓	$\overline{Y_i}\,\overline{X_i} C_i$
0	1	0	0	1	$Y_i \overline{X_i} C_i$	$\overline{Y_i} X_i \overline{C_i}$
0	1	1	1	0	$\overline{Y_i} X_i C_i$	$Y_i \overline{X_i}\,\overline{C_i}$
1	0	0	0	1	$\overline{Y_i} X_i C_i$	$Y_i \overline{X_i}\,\overline{C_i}$
1	0	1	1	0	$Y_i \overline{X_i} C_i$	$\overline{Y_i} X_i \overline{C_i}$
1	1	0	1	0	$Y_i X_i \overline{C_i}$	$\overline{Y_i}\,\overline{X_i} C_i$
1	1	1	1	1	$Y_i X_i C_i$	$Y_i X_i C_i$

Figure 3.3: Partial Truth Table for a Generic Full Adder [Prelab, Youssef Ibrahim]

Using the truth table above, equations for the generic full adder were obtained as shown below in Figure 3.4.



$$C_{i+1} = \bar{y}xc + y\bar{x}c + yx\bar{c} + yxc$$
$$= c(\bar{y}x + y\bar{x}) + y(x\bar{c} + xc)$$
$$= c(y \oplus x) + yx$$
$$S_i = \bar{y}\bar{x}c + \bar{y}x\bar{c} + y\bar{x}\bar{c} + yxc$$
$$= \bar{y}(\bar{x}c + x\bar{c}) + y(\bar{x}\bar{c} + xc)$$
$$= \bar{y}(x \oplus c) + y(\overline{x \oplus c})$$
$$= y \oplus (x \oplus c)$$

✓

Figure 3.4: Equations for a generic full adder [Prelab, Youssef Ibrahim]

Using these equations, we were able to construct a circuit for the full adder as shown below in Figure 3.5.

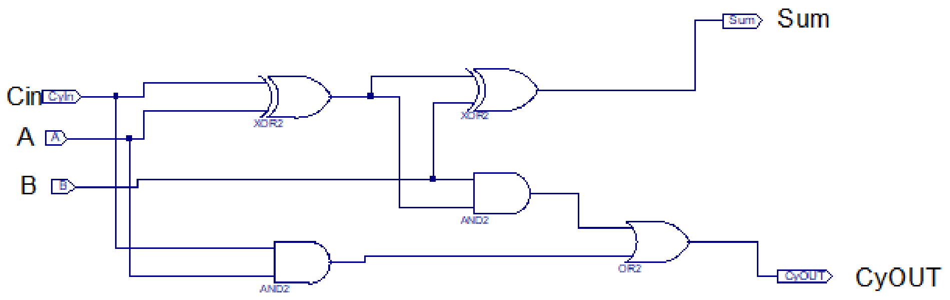


Figure 3.5: Full Adder schematic

The adder above is made of 2 XOR gates, 2 AND gates, and 1 OR gate. It takes the 3 inputs Y_i, X_i, and C_i to get the outputs S_i , and C_{i+1}.

3.3 Overflow (Ovf)

Overflow occurs when the sum of two 3-bit numbers exceeds the range of the bit field of those two numbers. To check for overflow, add the numbers with the sign extension, and if the two leftmost bits (S_3 and S_2) are not equal this means that there is an overflow. Figure 3.6 below shows examples about overflow.

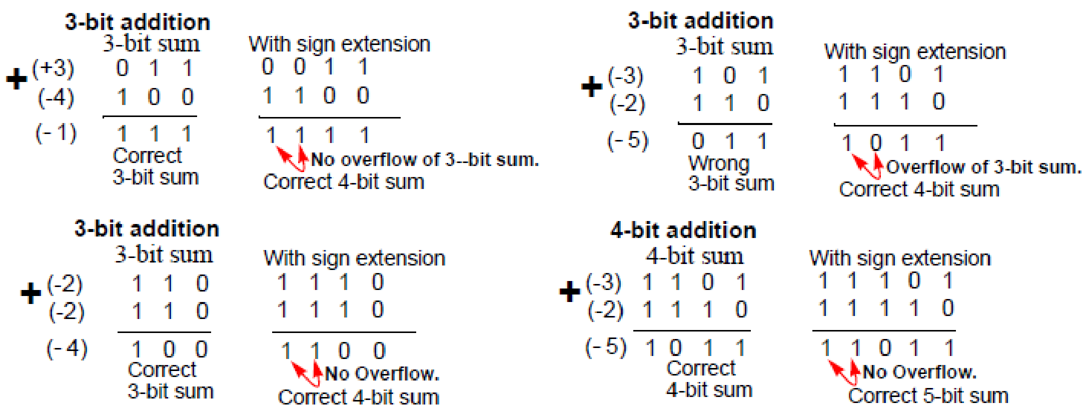


Figure 3.6: Examples of addition showing overflow [Lab 2 – p5, modified]

To compare values of S_3 and S_2 a XOR gate is required to check for overflow. Figure 3.7 below shows how a XOR gate is used to detect overflow.

S_2	S_3	Ovf $S_2 \neq S_3$
1	0	1
1	1	0
0	0	0
0	1	1

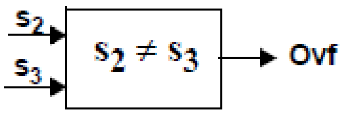


Figure 3.7: Inequality detection circuit [Lab 2 – p8, modified]

3.4 Final 3-bit binary sign-extend adder/subtractor circuit

Figure 3.8 below shows the final 3-bit binary sign-extend adder/subtractor circuit with an XOR gate added to check for overflow.

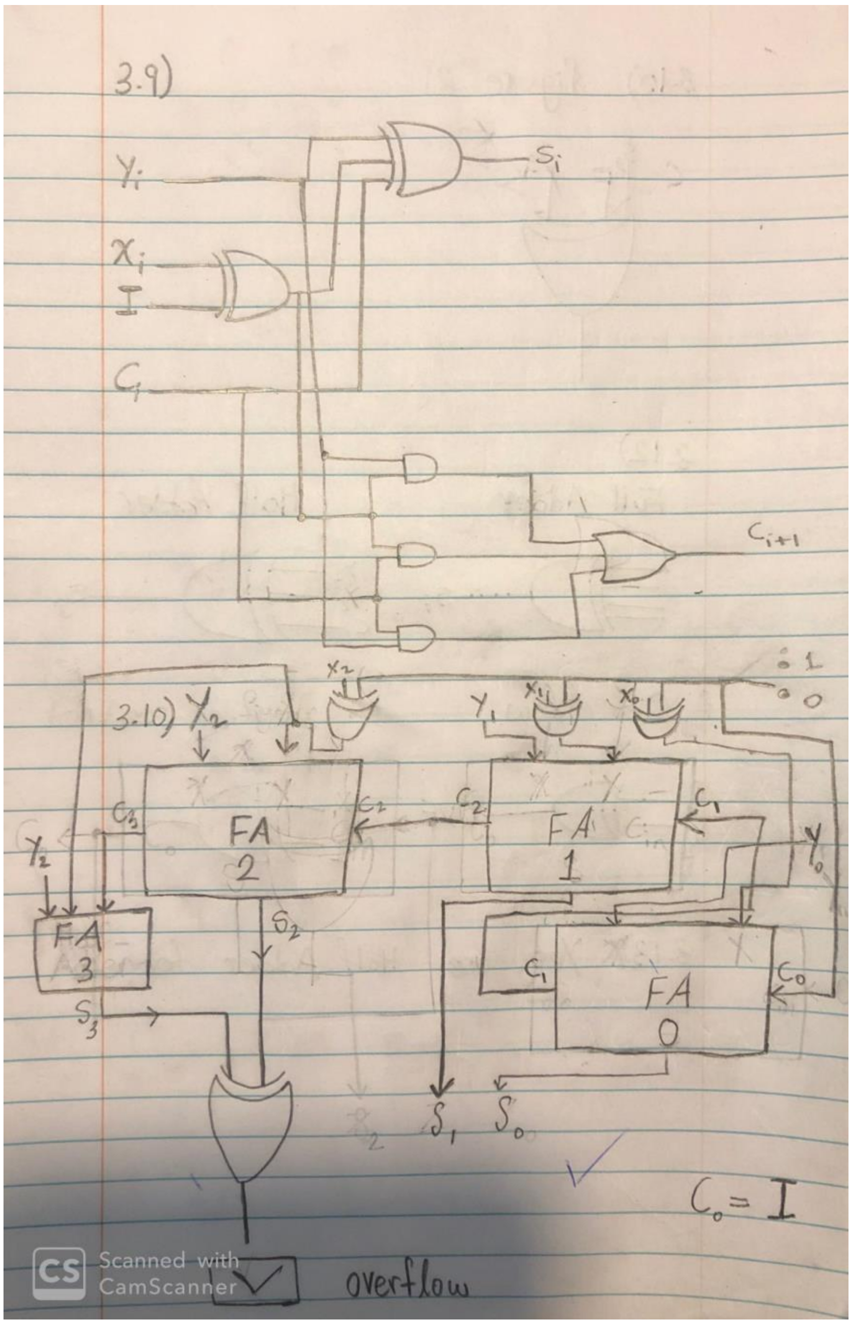


Figure 3.8: 3-bit binary sign-extend adder/subtractor circuit [Prelab, Youssef Ibrahim]

Figure 3.8 above was further simplified by replacing the Full-Adder 3 by a 3-input XOR gate that will work to output the sign-extended value as shown in Figure 3.9 below. Then, 2-input XOR gate was used to compare sum values calculated by the third full adder and the sign-extended sum value. If the values are not the same, it proves the overflow in the design, which means the output value could not be represented as a two’s complement binary number.

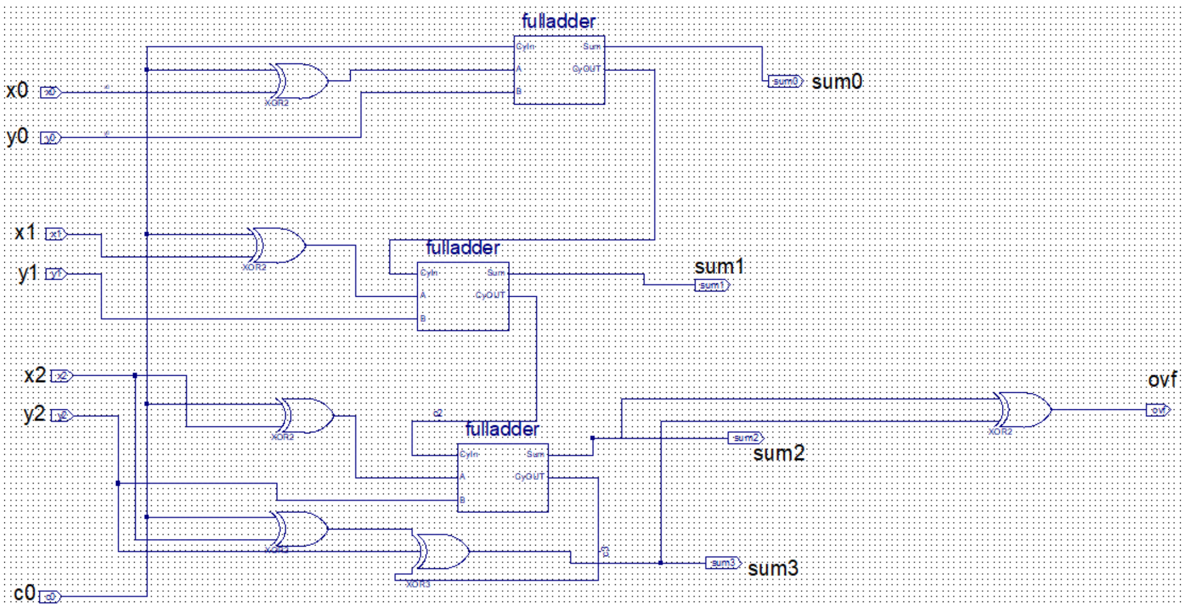
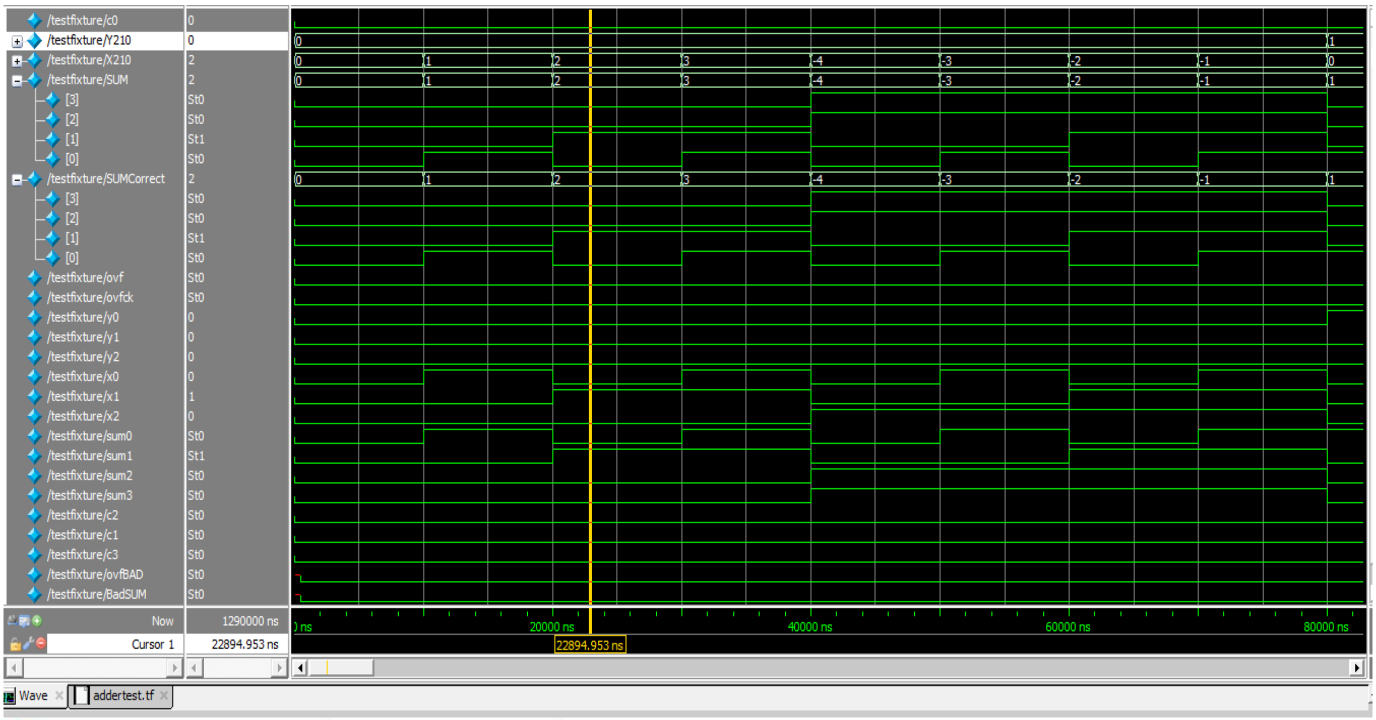


Figure 3.9: Complete 2's complement Addition/Subtraction Schematic

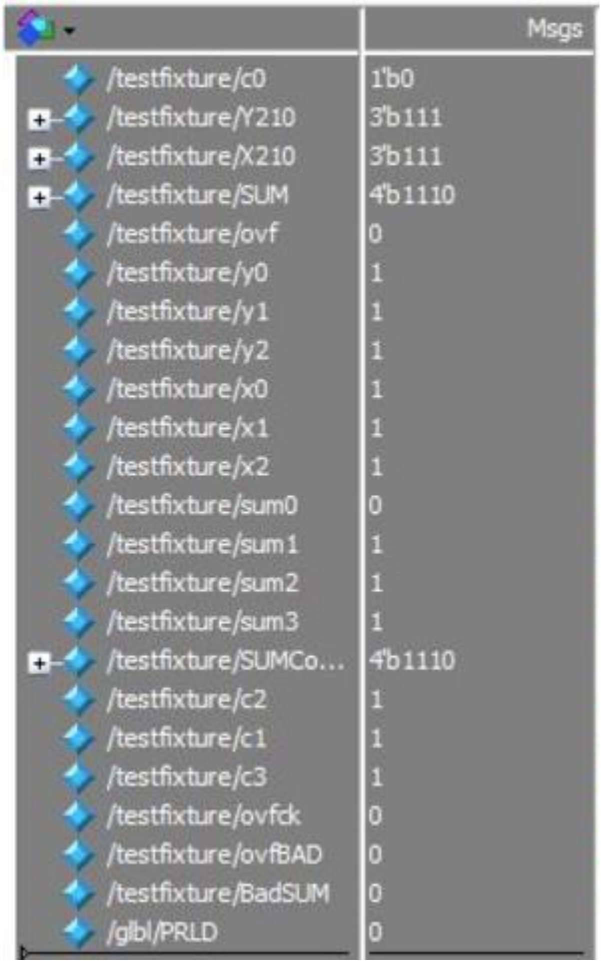
Generally, the design of the three-bit sign extended takes the three bits from the X input X_2, X_1, and X_0, for the y input Y_2, Y_1, and Y_0 calculates the sum and carry of all the bits. By getting the outputted value of sum, and then comparing it with the sum of the most significant bit full adder to see if there is an overflow.

4.0 TESTING AND IMPLEMENTATION

In this lab, we used a computer program called Xilinx ISE, to build and test our circuit. testing this circuit was done using the circuit design in the prelab. This software program was very helpful while building the circuit and it had a nice and easy to use feature called "Check Schematic tool". using this feature, we can detect the errors in our design and manage to fix it. Working on the lab we had no errors identified with our circuit, simulation runs successfully, but we took a little more time to work on the specifications of the program. After finishing the simulation comes the schematic run to form a waveform from "run all" option on the menu. All results were compared against prelab binary calculation section. The matching between prelab values and the program indicates the successful design. The first run showed a red line on the left of our screen which indicated a wrong segment in our design. The red line error could be a disconnected wire or wrong gate but not the structure of the circuit. The TAs comes to check if the circuit is missing or has any wrong gates connection and we fix the wire and put it in the right place. By fixing the design we get a yellow line instead that means the gates are working properly. Moreover, the circuit will be checked one more time by selecting cleanup project files. Only the correct waveforms would be shown on the screen. The output for this test would either be overflow or no overflow. The operations that can be performed on this lab is subtraction or addition.



```
/testfixture/c0            0
/testfixture/Y210          0
/testfixture/X210          2
/testfixture/SUM           2
  [3]                      St0
  [2]                      St0
  [1]                      St1
  [0]                      St0
/testfixture/SUMCorrect    2
  [3]                      St0
  [2]                      St0
  [1]                      St1
  [0]                      St0
/testfixture/ovf           St0
/testfixture/ovfck         St0
/testfixture/y0            0
/testfixture/y1            0
/testfixture/y2            0
/testfixture/x0            0
/testfixture/x1            1
/testfixture/x2            0
/testfixture/sum0          St0
/testfixture/sum1          St1
/testfixture/sum2          St0
/testfixture/sum3          St0
/testfixture/c2            St0
/testfixture/c1            St0
/testfixture/c3            St0
/testfixture/ovfBAD        St0
/testfixture/BadSUM        St0
Now                        1290000 ns
Cursor 1                   22894.953 ns
```



```
                           Msgs
/testfixture/c0            1'b0
/testfixture/Y210          3'b111
/testfixture/X210          3'b111
/testfixture/SUM           4'b1110
/testfixture/ovf           0
/testfixture/y0            1
/testfixture/y1            1
/testfixture/y2            1
/testfixture/x0            1
/testfixture/x1            1
/testfixture/x2            1
/testfixture/sum0          0
/testfixture/sum1          1
/testfixture/sum2          1
/testfixture/sum3          1
/testfixture/SUMCo...      4'b1110
/testfixture/c2            1
/testfixture/c1            1
/testfixture/c3            1
/testfixture/ovfck         0
/testfixture/ovfBAD        0
/testfixture/BadSUM        0
/glbl/PRLD                 0
```

5.0 CONCLUSION

In this lab, we used Xilinx ISE program to draw out circuit. The use of the program made it easy for us to figure out how to connect the wires and use the gates from the provided list, also, the steps that show how using the program were clear and simple to understand. The objective was achieved since the schematic was identical to the prelab values. The program is running and works as an iterative circuit adder for multibit numbers and providing the result. To improve this lab, the authors will have to be more familiarized with Xilinx ISE to obtain a much faster working steps and not use the program for the first time during the lab period.

References:

[1] J.Knight, "*Laboratory 2.0 A 3-bit Binary Sign-Extended Adder/Subtractor*", Carleton University, Department of Electronics, January 26, 2011