BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Course No: EEE 304
Course Title: Digital Electronics Laboratory

Level/Term: 3/2

Project Title: 4-bit Arithmetic and Logic Unit (ALU) using Proteus

Submitted by:

1706002

1706029

1706014

1706005

Abstract

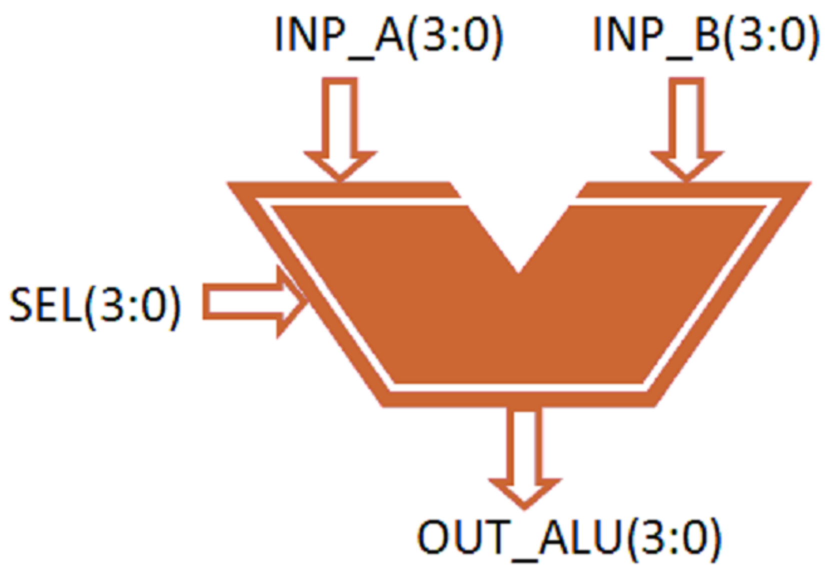
Digital design is an amazing and very broad field. The applications of digital design are present in our daily life, including computers, calculators, video cameras etc. In fact, there will be always need for high speed and low power digital products which makes digital design a future growing business. ALU (Arithmetic logic unit) is a critical component of a microprocessor and is the core component of central processing unit. Furthermore, it is the heart of the instruction execution portion of every computer. ALU's comprise the combinational logic that implements logic operations, such as AND, OR, NOT and arithmetic operations, such as ADDITION, SUBTRACTION, MULTIPLICATION, SUBTRACTION. We designed a 4-bit ALU (Arithmetic logic unit) through Proteus creating four main blocks that performs the arithmetic and several logical operations. The circuit design, as well as operation, mainly depends on the concepts of **Boolean algebra** as well as logic gates.

KEYWORDS: Arithmetic operation, Logical operation, Hexadecimal format, 7 segment decoder

Introduction

A **CPU** consists of three main sections: **memory for variables (registers)**, **control circuitry (microcode)**, and the **ALU**. The **ALU (Arithmetic Logic Unit)** is the part of a CPU that does calculations and condition testing.

For example, if we wish to add two binary numbers, it is the ALU that is responsible for producing the result. If any program needs to execute some code if two values are equal it is the ALU that performs the comparison between the values and then sets flags if the condition is met or not. In many CPUs, separate units exist for arithmetic operations (the arithmetic unit, AU) and for logic operations (the logic unit, LU).



Here A, B are two 4-bit inputs. Different Arithmetic & Logical operations are performed upon these two bits & Output is shown according to the value of selector pin.

Our ALU takes two 4-bits inputs (A and B) and performs 6 arithmetic functions and 2 logic functions. The 6 arithmetic functions we carried out are:

• Addition

• Subtraction

• Multiplication

• Subtraction

• Increment

• Decrement

And the 2 logical operation we carried out are:

• Logical OR

• Logical AND

Objectives:

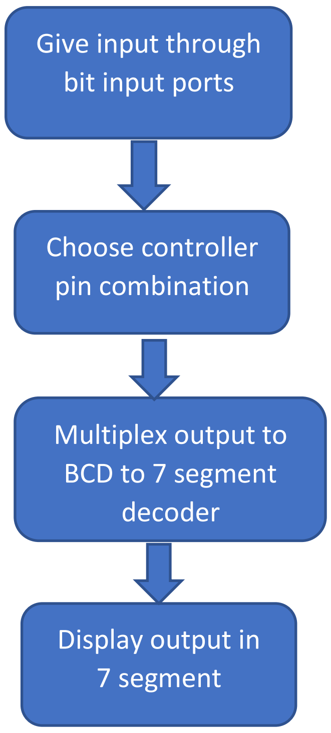
The main goals of our project are:

• To implement arithmetic operational circuit of ALU

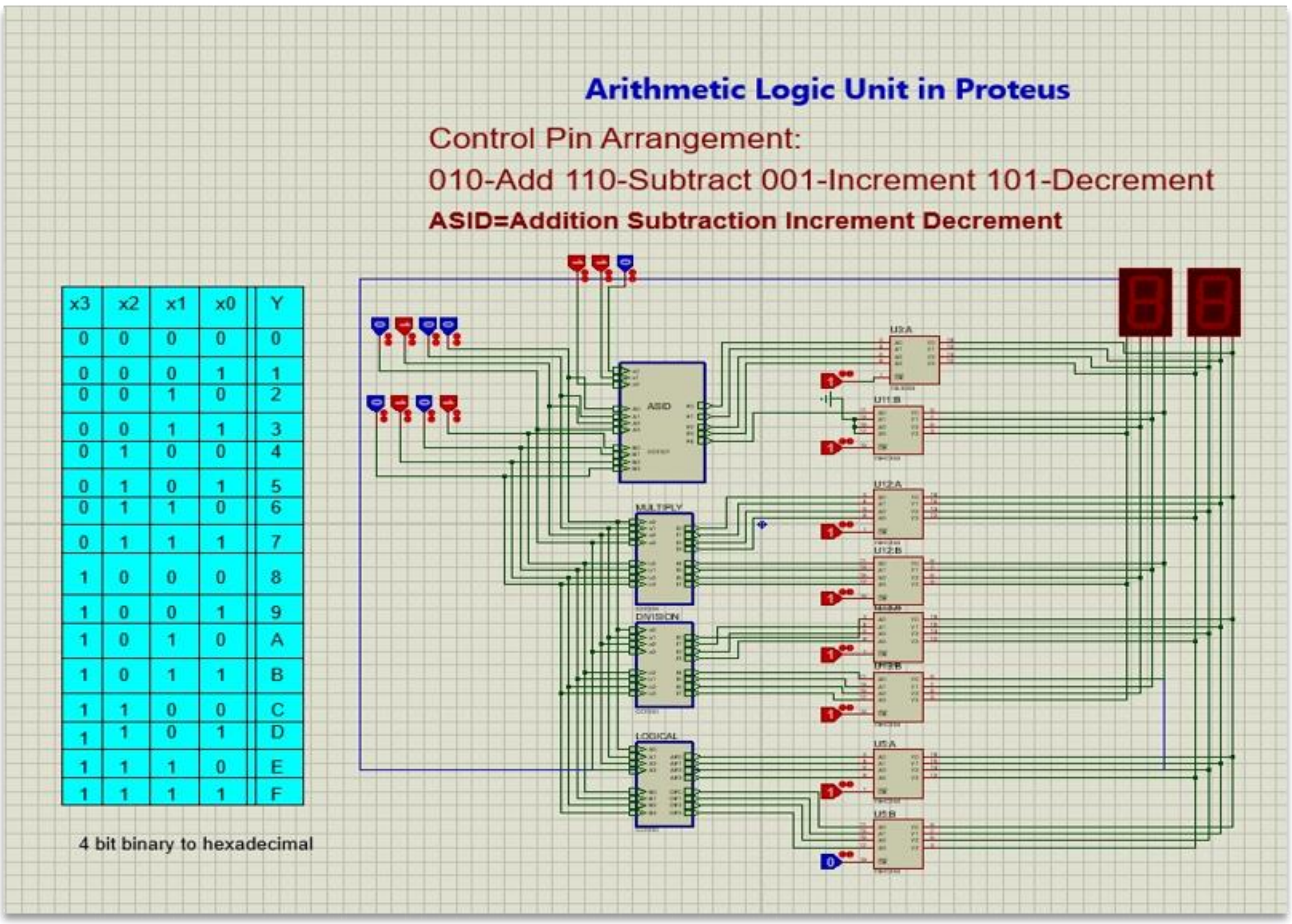
• To implement logical operational circuit of ALU

• To observe the output results in hexadecimal format

Workflow Chart


Give input through bit input ports
Choose controller pin combination
Multiplex output to BCD to 7 segment decoder
Display output in 7 segment

Main Circuit in Proteus Model



Arithmetic Logic Unit in Proteus

Control Pin Arrangement:

010-Add 110-Subtract 001-Increment 101-Decrement

ASID=Addition Subtraction Increment Decrement

x3	x2	x1	x0	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

4 bit binary to hexadecimal

Here, in the left side there are main 4 blocks ASID block, multiplication block, division block and logical operational block respectively and in the right side there are 8 hexadecimal-to-7 segment ICs. Here, every IC has an enable pin which is inverted. We need to keep the enable pin of a specific IC at active low while keeping others at active high to carry out a specific operation. And at the top right corner there is output display where the output result will be showed in hexadecimal format. We have showed the outputs in hexadecimal format to cover more range. So, in a display we can see 16 different digits and letters.

ASID:

This is the performs the following operations based on the control signal values:

- Addition
- Subtraction

- Increment
- Decrement

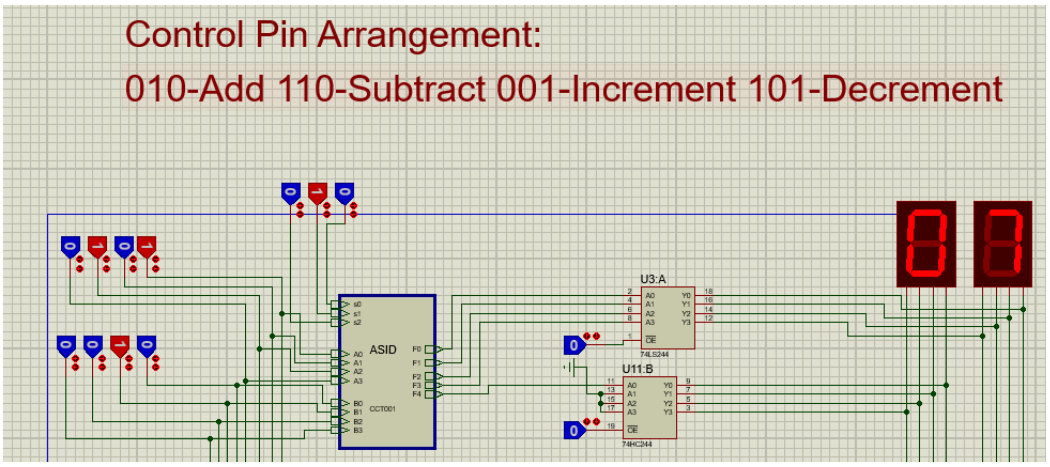


Figure- ASID subcircuit and connections.

Here, two 4-bit buffer circuits are connected to the ASID subcircuit with inverted enabling pin. If we want to perform ASDI operation, we need to keep the enabling pin of these buffer circuits active low while keeping others active high to keep others disabled.

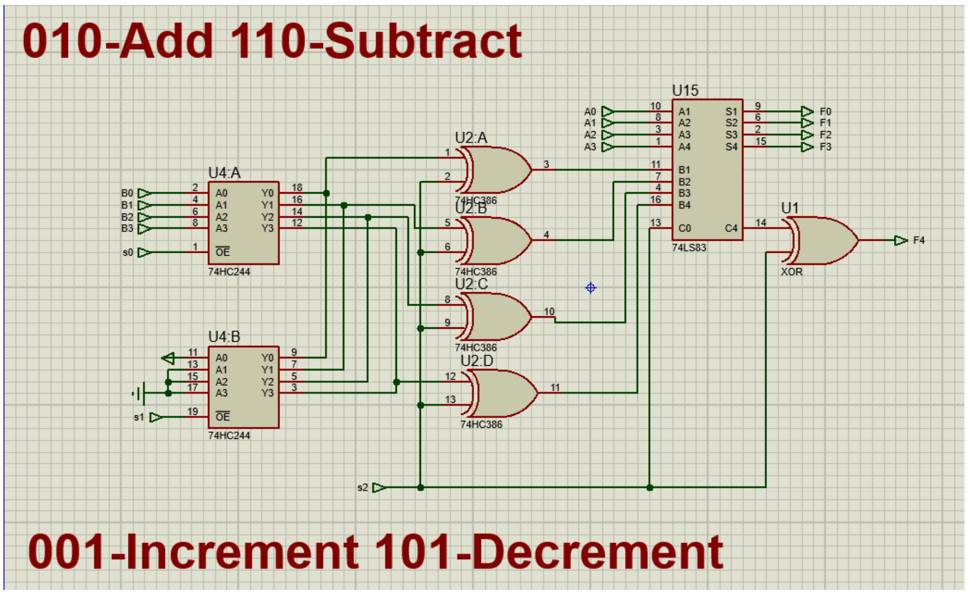
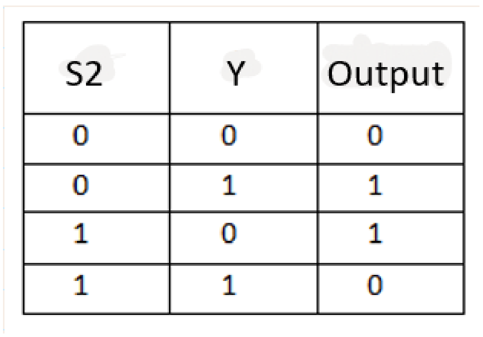


Figure- Subcircuit child sheet

Inside the subcircuit, the features are

- There is a full adder where A is directly fed into.
- There are two buffer circuits with inverted enabling pin.
- U4:A has B as input and S0 as enabling pin.
- U4:B has 0001 as input and S1 as enabling pin.
- Both U4:A & U4:B has same output wire connected to them which is Y
- Y goes into XOR operation with S2 sand the resultant is fed into the full adder circuit later.

Truth Table of XOR operation is-



S2	Y	Output
0	0	0
0	1	1
1	0	1
1	1	0

This S2 also goes into the full adder circuit as input carry. So, we can see

- If S2 = 0, full adder input = Y and
- If S2 = 1, full adder input = inverted Y + input carry S2 = ~Y + 1 = 2’s complement of Y.

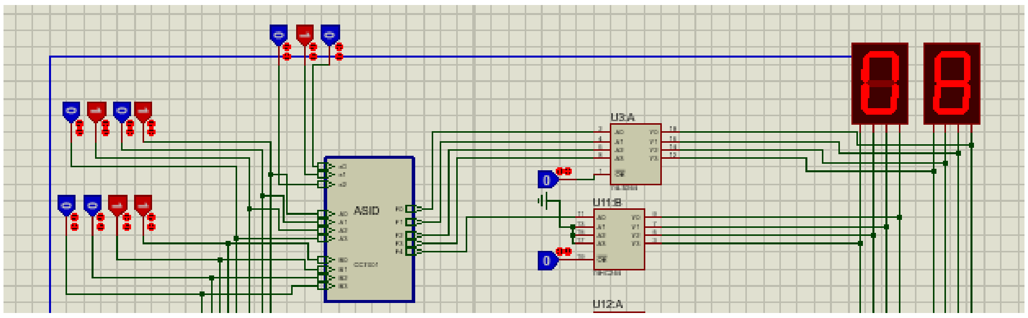
Thus, selector pin-based operation-

S2	S1 & S0 combination	Buffer circuit output Y	XOR operation output	Result F	Operation
0	10 U4:A enabled U4:B disabled	Y = B	Output = Y = B	A+B	**ADDITION**
	01 U4:A disabled U4:B enabled	Y = 0001	Output = Y = 0001	A+0001	**INCREMENT**
1	10 U4:A enabled U4:B disabled	Y = B	Output = ~Y = inverted B	A+ (~B+1) = A-B	**SUBTRACTION**
	01 U4:A disabled U4:B enabled	Y = 0001	Output = ~Y = inverted 0001	A+(~0001+1) = A-0001	**DECREMENT**

Examples:

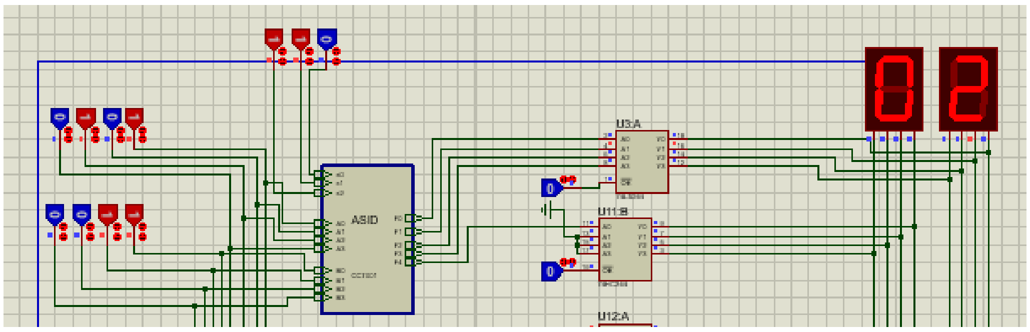
Addition:

$5_{10}+3_{10}=0101_2+0011_2=8_{10}$



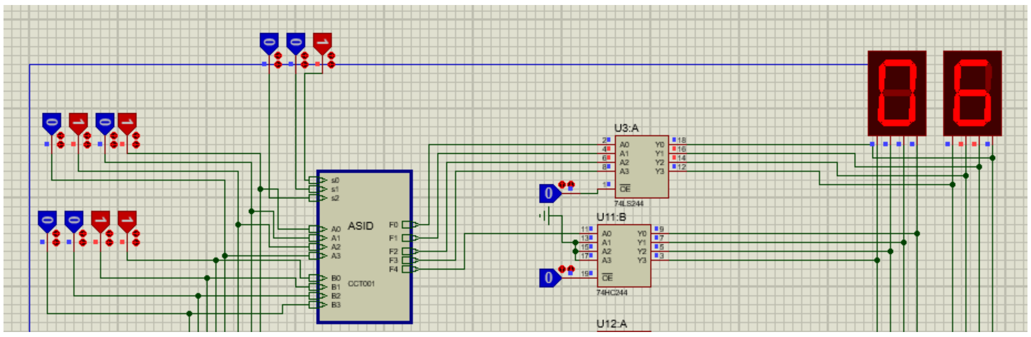
Subtraction:

$5_{10}-3_{10}=0101_2-0011_2=2_{10}$



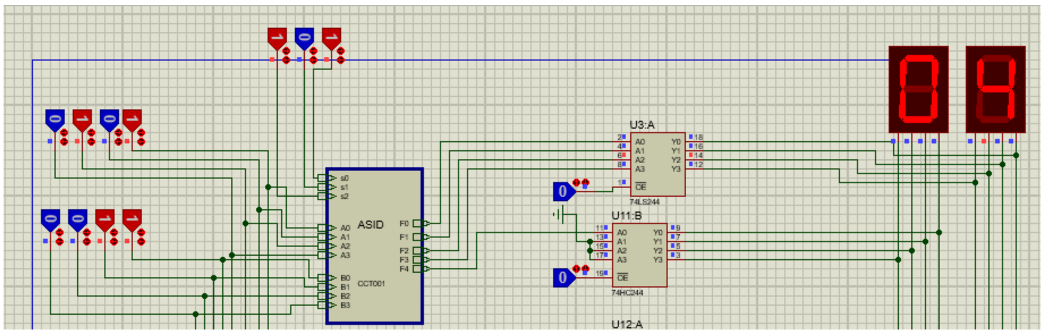
Increment:

$5_{10}+1_{10}=0101_2+0001_2=6_{10}$



Decrement:

$5_{10}-1_{10}=0101_2-0001_2=4_{10}$



Multiplication:

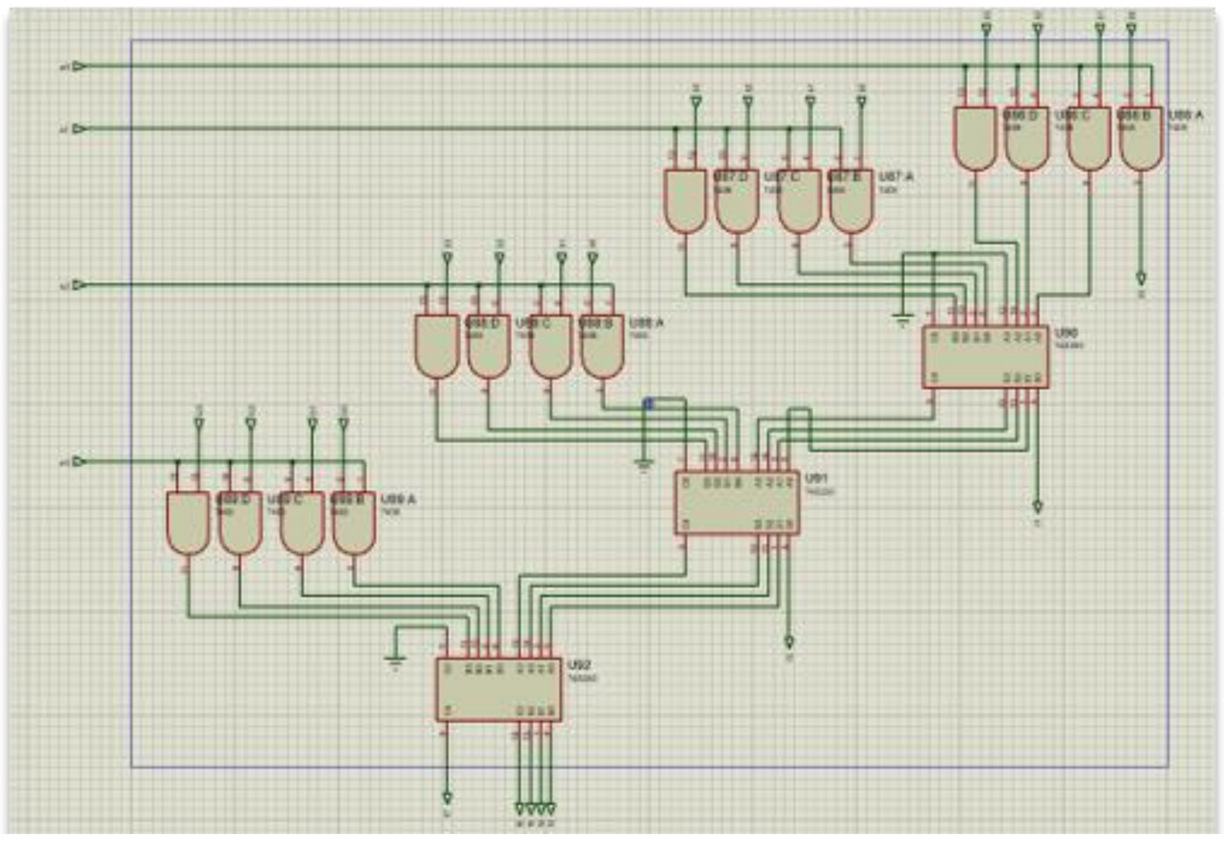


Figure: Multiplication Subcircuit

In this section, we have used:

- 16 AND gates
- 3 Full Adder

Methodology:

Here multiplier is A= $a_3a_2a_1a_0$, and multiplicand is B= $b_3b_2b_1b_0$

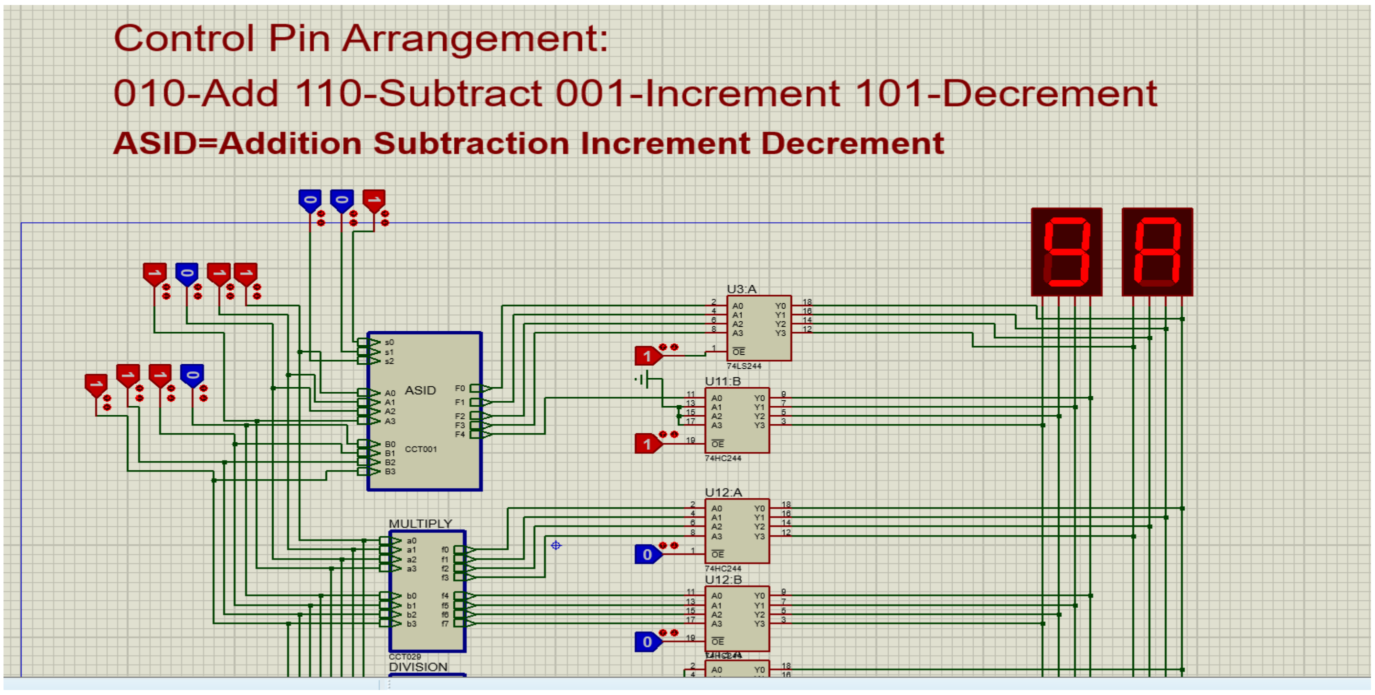
- Partial product 0 (PP0) is obtained by using and of a0 with each bit of B
- Partial product 1(PP1) is obtained by using and of a1 with shifted version of B and then adding it with PP0
- Partial product 2(PP2) is obtained by using and of a2 with shifted version of B and then adding it with PP1
- Partial product 3(PP3) is obtained by using and of a3 with shifted version of B and then adding it with PP2

- Bits of PP3 and LSB's of PP2, PP1, PP0 is the desired result.

Example:

A=1011; B=1110

AB= $(10011010)_2 = (9A)_{16}$



Division:

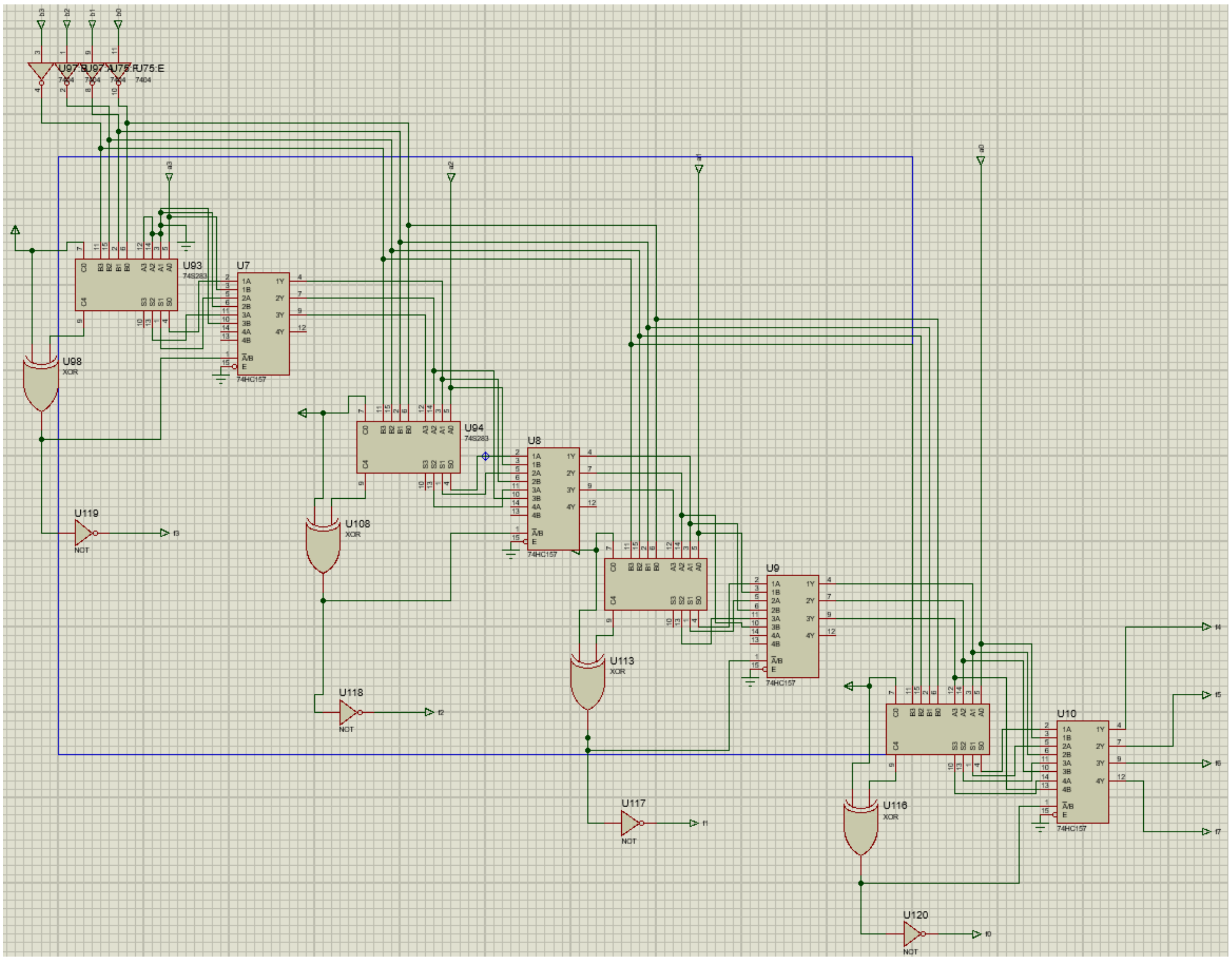


Figure: Division Subcircuit

In this section, we have used Iterative Subtraction Circuit. The circuit has-

- 4 XOR gates
- 4 four-bit binary adder (IC7483)
- 8 NOT gates
- 4 16x1 Multiplexer IC (IC74157)

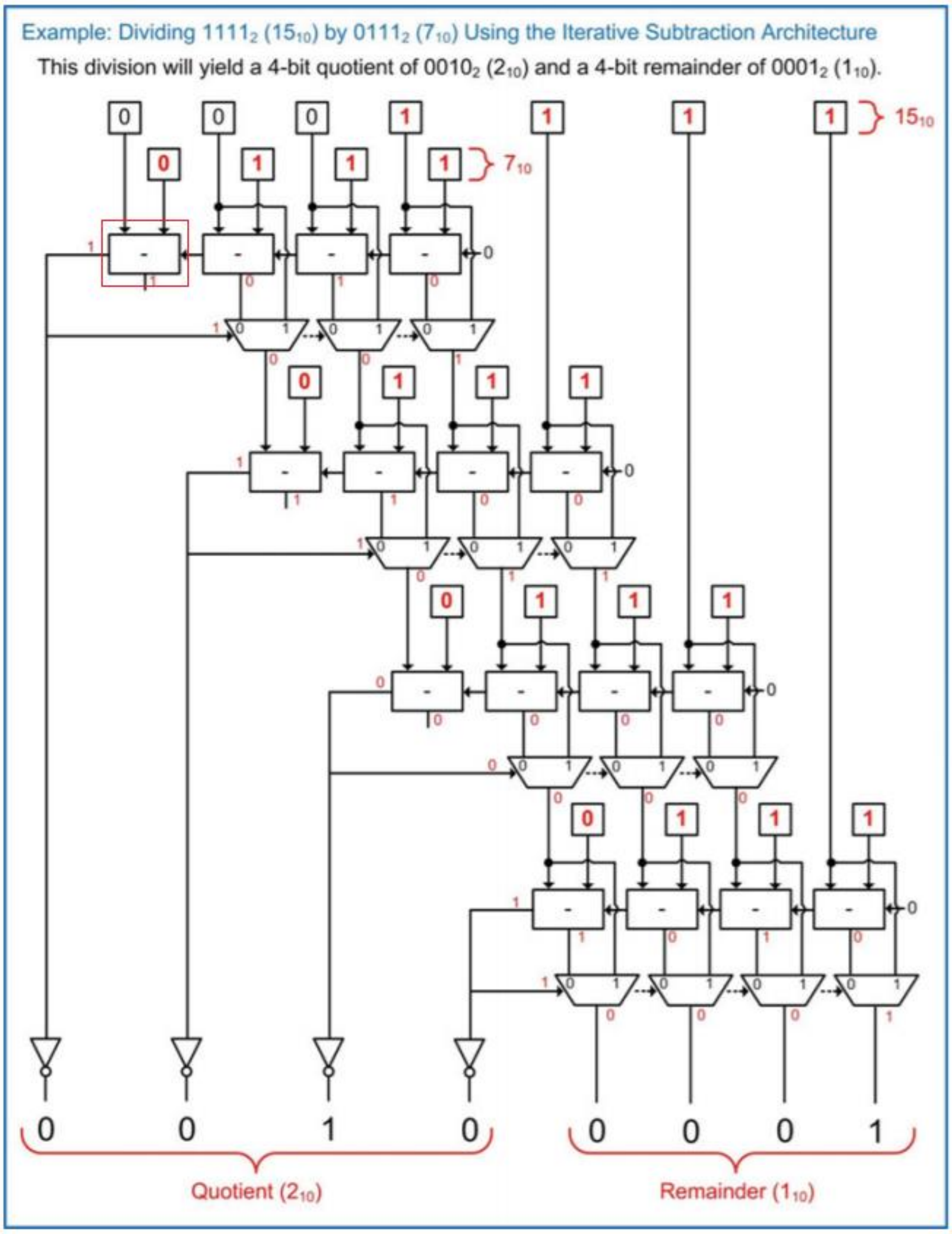
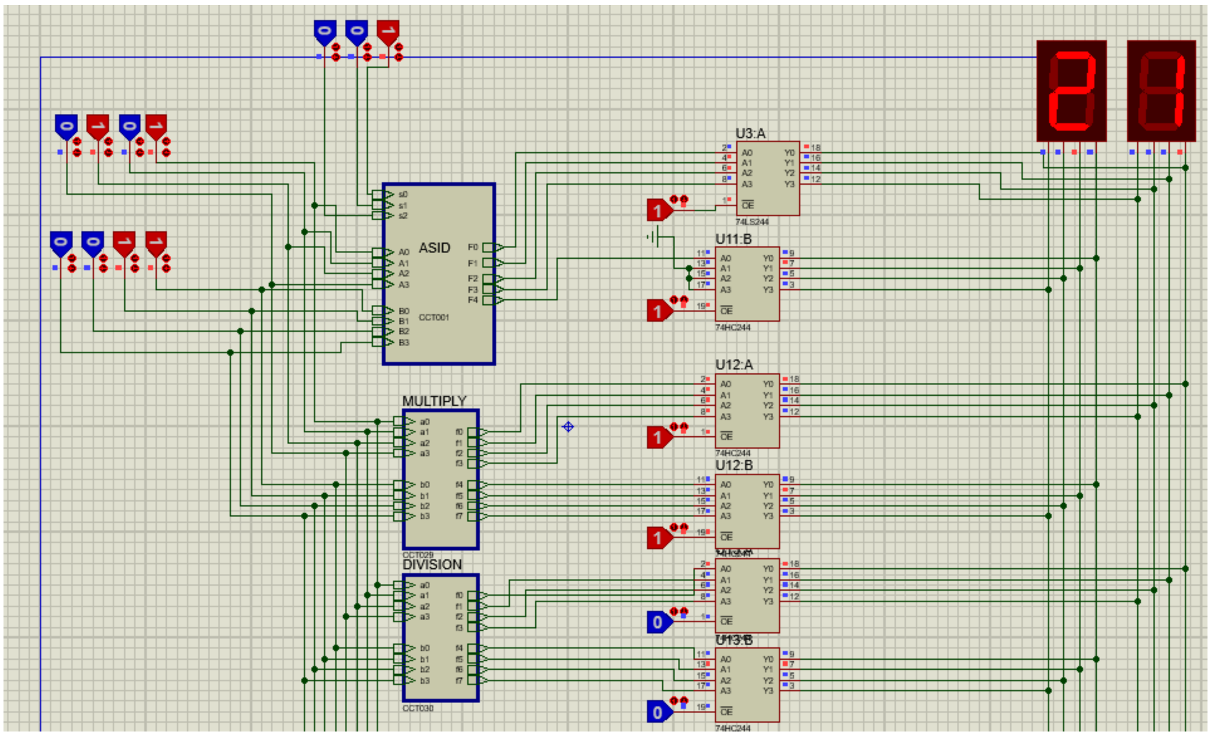

Example: Dividing 1111_2 (15_{10}) by 0111_2 (7_{10}) Using the Iterative Subtraction Architecture
This division will yield a 4-bit quotient of 0010_2 (2_{10}) and a 4-bit remainder of 0001_2 (1_{10}).
15_{10}
7_{10}
0 0 1 0
Quotient (2_{10})
0 0 0 1
Remainder (1_{10})

Figure: An example of binary division using iterative subtraction architecture

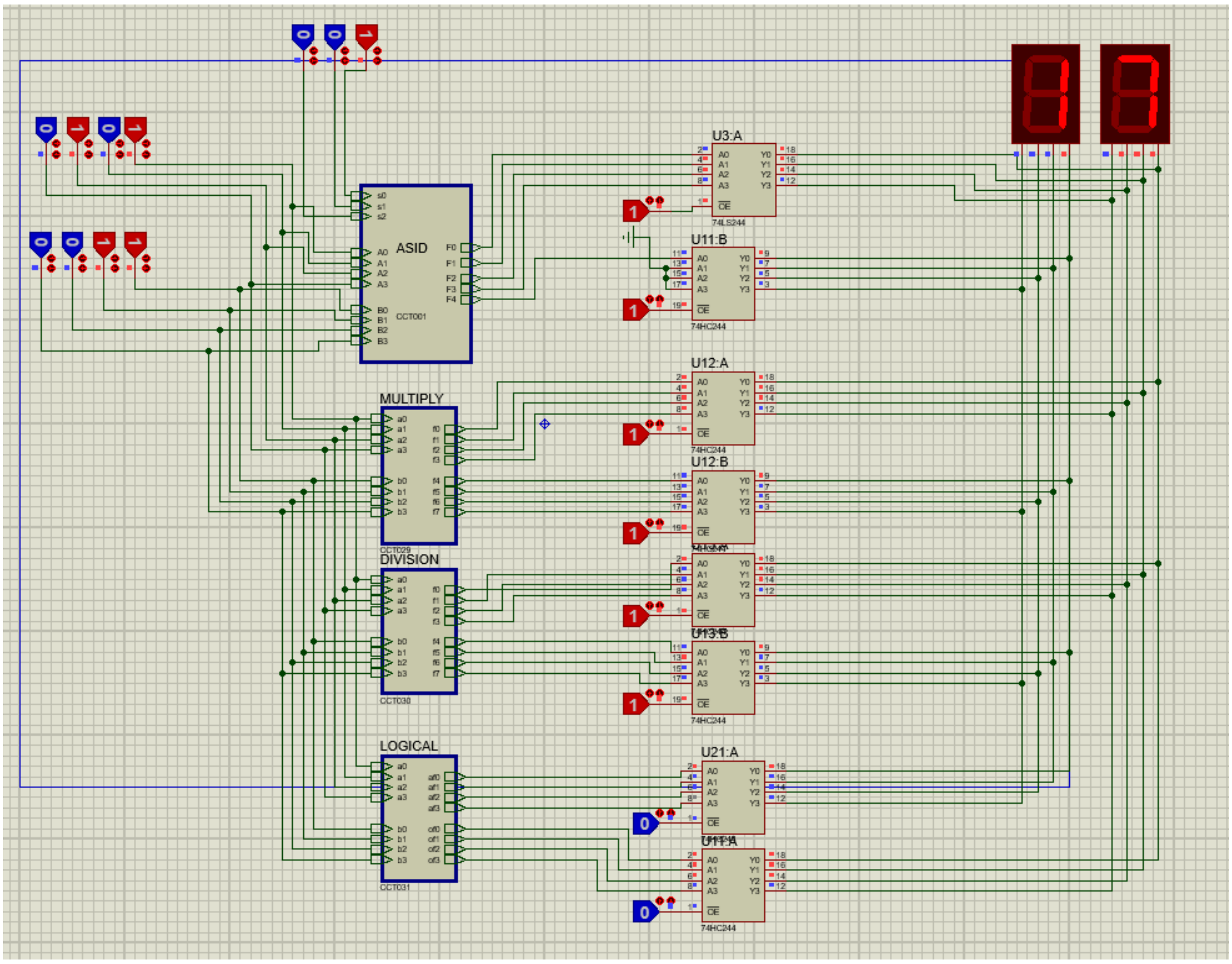
The output is shown through two Hexadecimal to 7 segment display decoders. Left display shows the remainder, right one shows the quotient.



Here, dividend is $0101_2=5_{10}$, divisor is $0011_2=3_{10}$. So, remainder is 2 and quotient is 1, as expected.

Logical operations:

In our logical unit, we implemented two logical operational circuits, one is bitwise OR, another is bitwise AND. Since we implemented 4-bit ALU, we needed 4 OR and AND gates for each operation.



Here, inputs are 0101 and 0011, their bitwise AND output is 0001 which is 1 in hexadecimal and is shown on the left 7 segment display. Their bitwise OR output is 0111 which is 7 in hexadecimal and is shown on the right seven segment display.

We didn’t implement bitwise XOR and XNOR operation because XOR indicates addition and XNOR indicates subtraction in 2’s complement form.

Limitations:

• Due to online class shifting and absence of our 3 group members in Dhaka, we couldn't do the hardware implementation of this project.

• Division by zero is undefined. But in our subcircuit, division by zero shows quotient as F and remainder as 0. This is because in iterative subtraction architecture, it is checked how many times the divisor can be subtracted from the dividend. In this case it will yield 1 from each adder IC. Hence the output is 1111 and remainder is 0000.

The division process doesn’t go in any loop because since there are 4 adder ICs, the process is terminated after the final IC yields output.

• In ASID subcircuit, s1 and s0 cannot be simultaneously off. Because then they conflict, and the circuit gives random incomprehensible value as outputs.

Conclusion:

In this project, a 4-bit 8-operational ALU was designed. The design is implemented on Proteus. All the output results of our carried out arithmetic and logical operations were correct. So, it can be said that the motto of our project is fulfilled successfully. However, hardware implementation can be done by following this same logic without any change. If we could get the opportunity in offline class, we would be able to do so.

Acknowledgement:

First, we would like to express our sincere gratitude to Hamidur Rahman Sir, Associate Professor, Dept. of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology for guiding us through the whole project. His advice and suggestions were much valuable for us to complete the project properly.

We would also like to thank Rifat Shahriar Sir, Lecturer (PT), Dept. of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology for helping us to complete our project with his invaluable guidance and constructive criticisms. In short, without their wholehearted support, it would not be feasible to complete the project properly.