BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CSE 306

Computer Architecture Sessional

ASSIGNMENT NO: 01

ASSIGNMENT NAME: 4-BIT ALU SIMULATION AND IMPLEMENTATION

SECTION: B2

GROUP: 06

GROUP MEMBERS: 1905104, 1905105, 1905107, 1905108, 1905109

A. INTRODUCTION

Arithmetic Logic Unit, ALU, is one of the most important components of a microprocessor in a computer system. It is a combinational circuit that performs both arithmetic and logical operations. Alongside the input and the output bits, there are selection bits that determine which operation to perform. Furthermore, some flags provide us with some extra valuable information based on the results.

B. PROBLEM SPECIFICATION

It is required to design a 4-bit ALU that can perform the following operations:

Control Signals			**Functions**
Cs2	Cs1	Cs0	
X	0	0	Sub
0	0	1	OR
X	1	0	NEG A
0	1	1	AND
1	0	1	Transfer A
1	1	1	Add

Here, csX are the selection bits that will determine the operation to be performed. The inputs are A and B, which are both 4-bit numbers.

In addition to these operations, it is required to implement 4 flags:

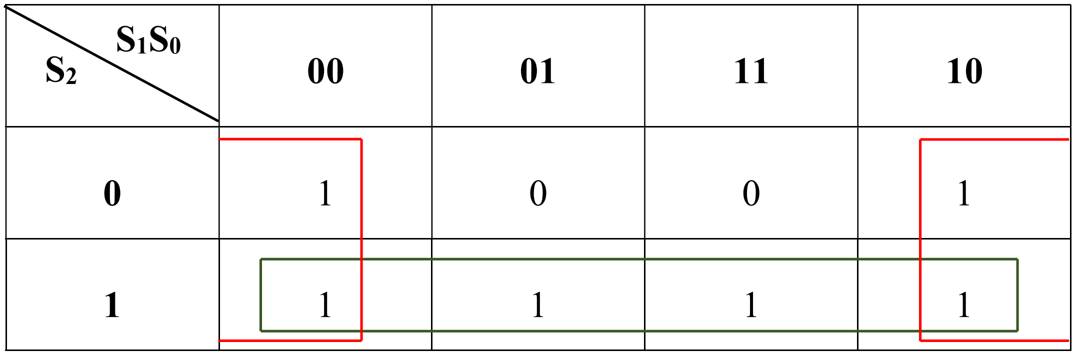
- Carry (C)
- Sign (S)
- Overflow (O)
- Zero (Z)

C. DESIGN STEPS

Calculating Selector Equation for the Arithmetic / Logical Selecting MUX :

Control Signals			Functions	Output
S_2	S_1	S_0		
0	0	0	Sub	Arithmetic /1
0	0	1	OR	Logical /0
0	1	0	NEG A	Arithmetic /1
0	1	1	AND	Logical /0
1	0	0	Sub	Arithmetic /1
1	0	1	Transfer A	Arithmetic /1
1	1	0	NEG A	Arithmetic /1
1	1	1	Add	Arithmetic /1

Table 1:TRUTH TABLE FOR SELECTING ARITHMETIC/LOGICAL OPERATION



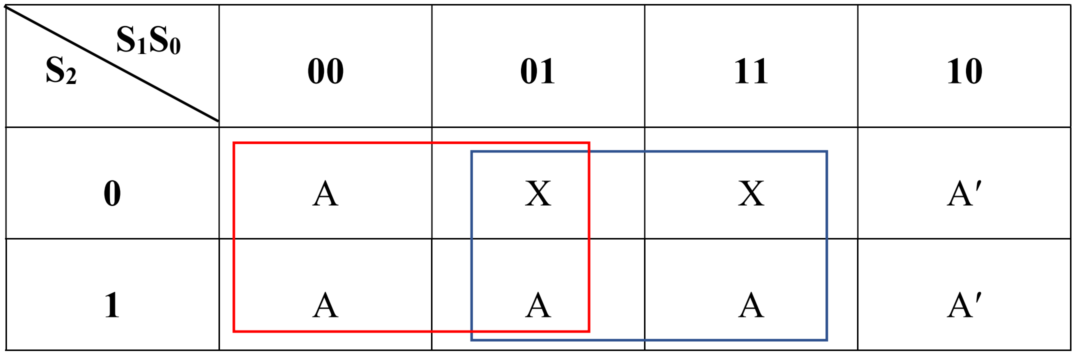
S_2 \ S_1S_0	00	01	11	10
0	1	0	0	1
1	1	1	1	1

Selector Equation: $S_2 + S_0'$

Calculating Selector Equation for the A / A′ Selecting MUX :

Control Signals			Functions	Output
S_2	S_1	S_0		
0	0	0	Sub (A+B′+1)	A
0	0	1	OR (A∨B)	X
0	1	0	NEG A (A′+0+1)	A′
0	1	1	AND (A∧B)	X
1	0	0	Sub (A+B′+1)	A
1	0	1	Transfer A (A+0+0)	A
1	1	0	NEG A (A′+0+1)	A′
1	1	1	Add (A+B+0)	A

Table 2:TRUTH TABLE FOR SELECTING A / A′



S_1S_0 S_2	00	01	11	10
0	A	X	X	A′
1	A	A	A	A′

Selector Equation: $S_1' + S_0$

Calculating Selector Equation for the B/ B′ Selecting MUX :

Control Signals			Functions	Output
S_2	S_1	S_0		
0	0	0	Sub (A+B′+1)	B′
0	0	1	OR (A⋁B)	X
0	1	0	NEG A (A′+0+1)	X
0	1	1	AND (A⋀B)	X
1	0	0	Sub (A+B′+1)	B′
1	0	1	Transfer A (A+0+0)	X
1	1	0	NEG A (A′+0+1)	X
1	1	1	Add (A+B+0)	B

Table 3:TRUTH TABLE FOR SELECTING B / B’

S_1S_0 S_2	**00**	**01**	**11**	**10**
0	B′	X	X	X
1	B′	X	B	X

Selector Equation: S_1

Calculating Selector Equation for the (B / B′) /0 Selecting MUX :

Control Signals			Functions	Output
S_2	S_1	S_0		
0	0	0	Sub (A+B′+1)	B′ (1)
0	0	1	OR ($A \vee B$)	X
0	1	0	NEG A (A′+0+1)	0
0	1	1	AND ($A \wedge B$)	X
1	0	0	Sub (A+B′+1)	B′ (1)
1	0	1	Transfer A (A+0+0)	0
1	1	0	NEG A (A′+0+1)	0
1	1	1	Add (A+B+0)	B (1)

Table 4:TRUTH TABLE FOR SELECTING (B / B’) / 0

S_1S_0 S_2	**00**	**01**	**11**	**10**
0	1	X	X	0
1	1	0	1	0

Selector Equation: $S_1'S_0 + S_0'S_1 = S_1 \oplus S_0$

Calculating Equation for C_{in} :

Control Signals			Functions	Output
S_2	S_1	S_0		
0	0	0	Sub (A+B′+1)	1
0	0	1	OR (A⋁B)	X
0	1	0	NEG A (A′+0+1)	1
0	1	1	AND (A⋀B)	X
1	0	0	Sub (A+B′+1)	1
1	0	1	Transfer A (A+0+0)	0
1	1	0	NEG A (A′+0+1)	1
1	1	1	Add (A+B+0)	0

Table 5:TRUTH TABLE FOR C_{in}

S_1S_0 S_2	**00**	**01**	**11**	**10**
0	1	X	X	1
1	1	0	0	1

Selector Equation: S_0'

Calculating Selector Equation for the AND/ OR Selecting MUX:

Control Signals			**Functions**	**Output**
S_2	**S_1**	**S_0**		
0	0	0	Sub (A+B′+1)	X
0	0	1	OR (A∨B)	OR
0	1	0	NEG A (A′+0+1)	X
0	1	1	AND (A∧B)	AND
1	0	0	Sub (A+B′+1)	X
1	0	1	Transfer A (A+0+0)	X
1	1	0	NEG A (A′+0+1)	X
1	1	1	Add (A+B+0)	X

Table 6: TRUTH TABLE FOR SELECTING AND / OR

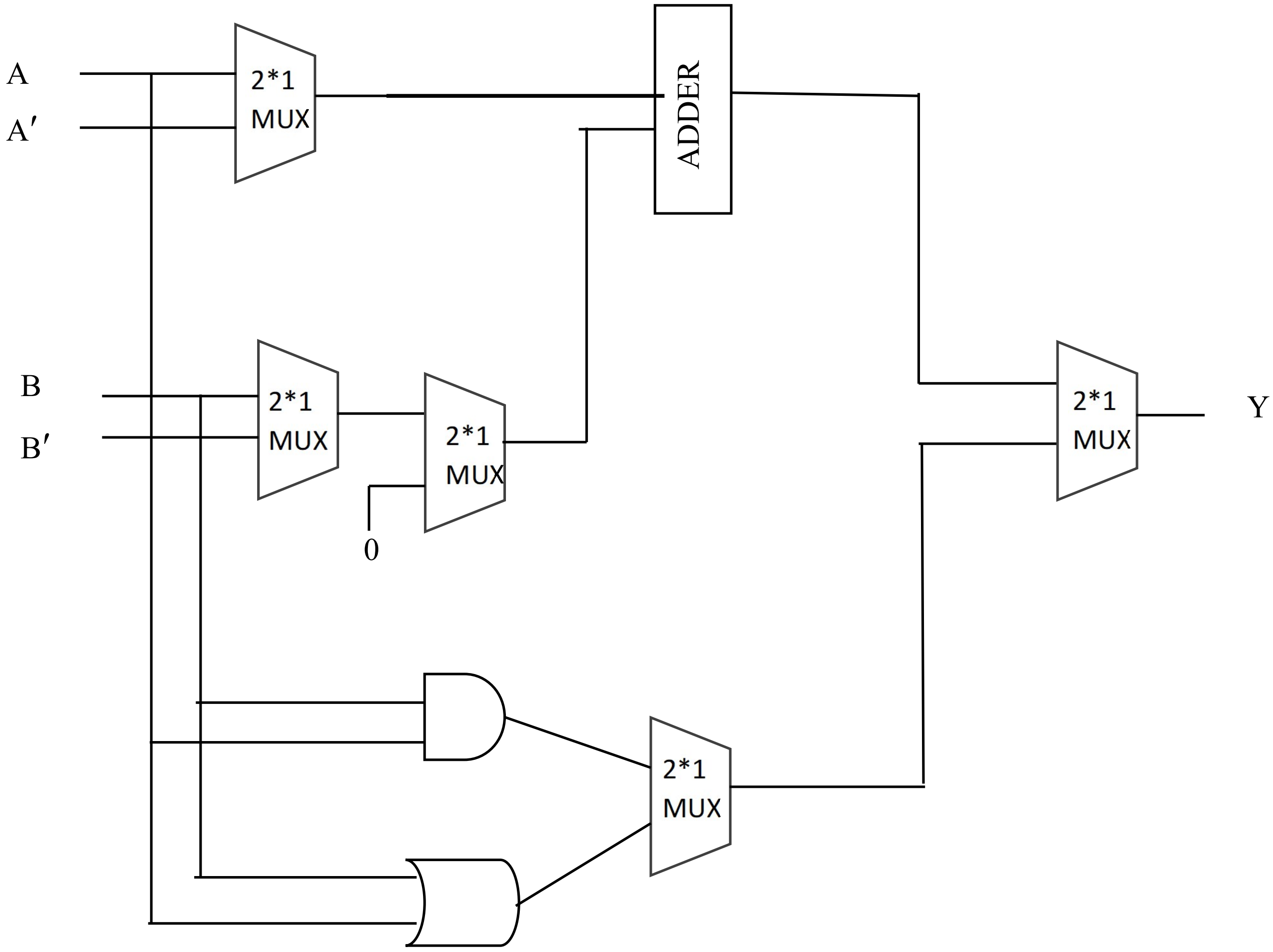
S_2 \ S_1S_0	**00**	**01**	**11**	**10**
0	X	OR	AND	X
1	X	X	X	X

Selector Equation: S_1

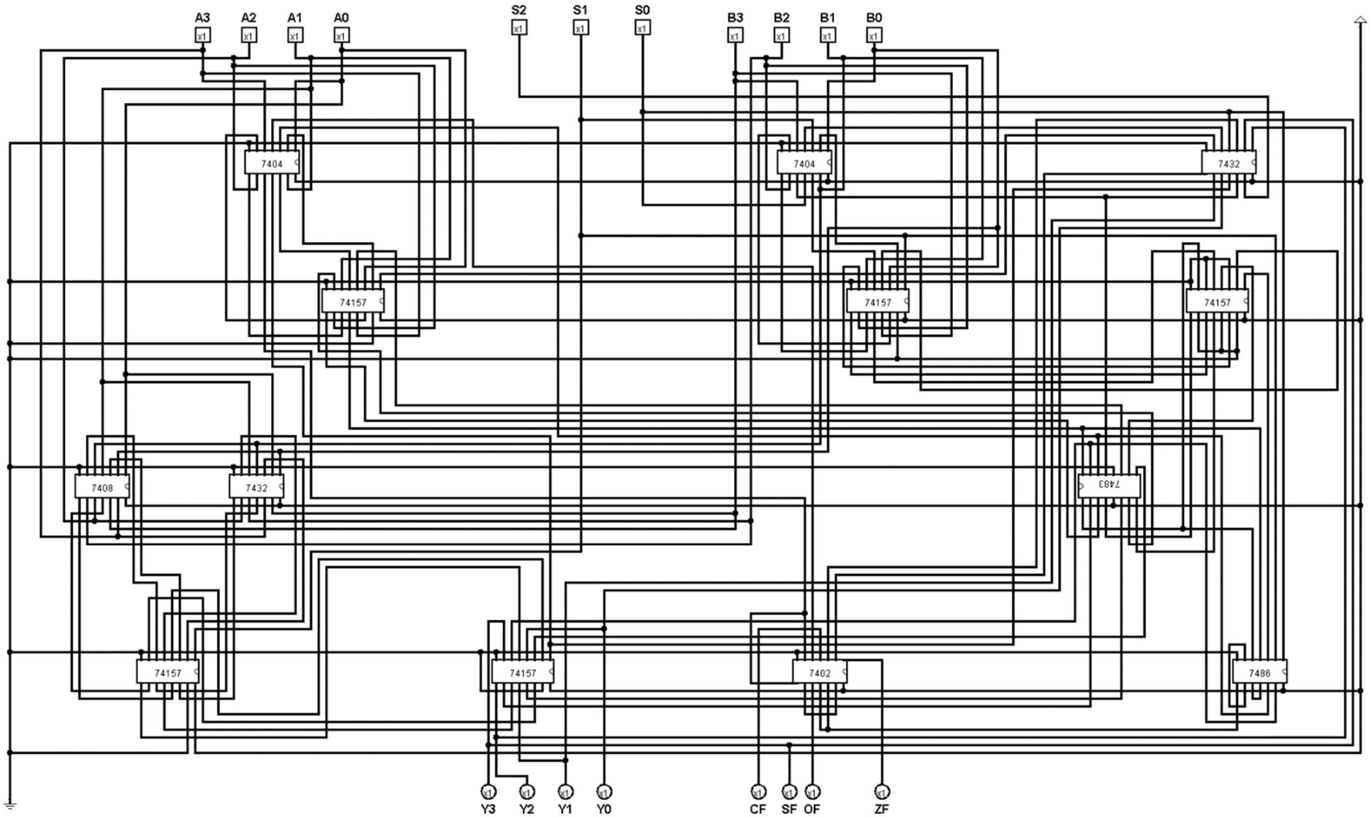
D. TRUTH TABLE

All the required truth tables were discussed in the (C. DESIGN STEPS) section.

E. BLOCK DIAGRAM



F. CIRCUIT DIAGRAM



G. IC CHART

IC	Count
7402	1
7404	2
7408	1
7432	2
7483	1
7486	1
74157	5

H. SIMULATOR

Logisim Generic 2.7.1

I. DISCUSSION

In this experiment, a 4-bit ALU was designed which takes two 4-bit inputs, and based on the selector bits, the ALU could calculate proper output and flags. A total of 13 ICs were needed for the experiment. The IC count could have been quite less if the selector bits were not random. Also with the given specifications, it could be designed with 12 ICs but then the design would have been a lot more complicated. The flags were simplified a little bit for some operations.

• All the AND-OR operations were done by 2-input SSIs.

• At first it was intended to build arithmetic and logical operations combined using an adder. But the design was complex and needed more ICs than the current design. So, arithmetic and logical operations were done separately and connected using MUX.

• To implement all arithmetic operations using one adder, every logic was made with three literals.
ADD == A + B + 0
SUB == A + B′ + 1

NEG A == A′ + 0 + 1
TRANSFER A == A + 0 + 0
So, the options are -
Adder’s first input - A/A’
Adder’s second input - B/B’/0
C_{in} = 0/1
A/A’ were selected using one 2x1 MUX
To select B/B’/0 two 2x1 MUX were used. One for B/B selection and the other was for selecting 0 or the output of the first MUX. As there were 3 options extra IC was needed and the IC count increased. Moreover, using a decoder with two 4x1 MUX combining logical and arithmetic operations in the Adder didn’t reduce IC because of the 3 choices in the adder’s second input.
C_{in} was calculated using the logical operation of selection bits.

- While doing the hardware work, no MUX was giving proper output at first. Which caused a waste of a lot of time. Later it was recognized that the switches were not properly grounded. As a result, MUX was always selecting the high voltage output.

- The third bits carry doesn’t come out of the adder IC. So, overflow carry was implemented by the XOR logical operation.