CSE 210

Computer Architecture Sessional

Assignment-1: 4-bit ALU Simulation

Section - C2
Group - 04

Members of the Group:

- **2105155** - Md. Siam Ahamed
- **2105158** - Al Shahriar Alif
- **2105160** - Imdadul Hasan Hamim
- **2105172** - Mezba-Us-Salaheen

1 Introduction

ALU, elaborated as Arithmetic and Logic Unit, is quite simply the mathematical brain of a computer. It is mainly a combinational digital circuit that performs arithmetic and bitwise and logical operations on integer binary numbers. Needless to say, it is a core building block of any computing unit, ranging from Central Processing Unit (CPU) to Graphics Processing Unit (GPU)s.

As the name suggests, an ALU comprises of two main units: Arithmetic Unit and Logic Unit. It supports a wide range of operations including arithmetic ones like addition, subtraction, increment, decrement, transfer, and logical ones like NOT, OR, XOR, AND etc. The control unit routes the source information from registers into ALU inputs. To select a particular operation, an ALU has some selection lines or bits. Decoding system allows to support 2^k different operations for k selection lines or bits. In our implemented ALU, there are 3 control selection inputs.

A parallel adder is the heart of the arithmetic part of the ALU. Multiplexed inputs to the IC paves the way to achieve expected results for varieties of arithmetic operations. Besides, logical operations can be performed with their respective ICs, and in some efficient designs, a particular IC is used to do a different operation other than its intended one. There are also 4 status outputs (flags) in our designed ALU. They are denoted by C(Carry Flag), Z(Zero Flag), V(Overflow Flag), S(Sign Flag). Their representations carry out the following meanings:

C: **C** is simply the C_{out} of the adder used in the ALU. So, any carry out results in it being 1, otherwise it is set to 0. Logical operations cause it to be 0.

Z: If the last operation of the ALU yielded an output of 0, **Z** is set, else it is 0.

V: If after any arithmetic operation, two positive numbers result in a negative output, or two negative numbers result in a positive output, that is an overflow and **V** is set. After any logical operation, it is cleared.

$$S_3 = A_3 \oplus I_3 \oplus C_3$$
$$\Rightarrow C_3 = S_3 \oplus A_3 \oplus I_3 \quad (1)$$
$$V = C_3 \oplus C_{out} \quad (2)$$

Combining 1 and 2

$$V = A_3 \oplus I_3 \oplus S_3 \oplus C_{out} \quad (3)$$

S: It reflects the output MSB.

2 Problem Specification with Assigned Instructions

Design a 4-bit ALU with three selection bits cs0, cs1 and cs2 for performing the following operations:

Control Signals			Functions	Boolean Expression
cs2	cs1	cs0		
X	0	0	AND	$A \cap B$
X	0	1	Decrement A	$A - 1$
0	1	0	Add with Carry	$A + B + 1$
0	1	1	Subtract	$A - B$
1	1	0	Complement A	A'
1	1	1	Transfer A	A

Table 1: Problem Specification

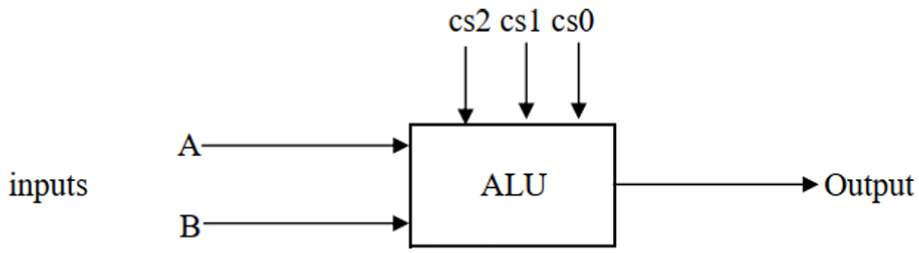


Figure 1: 4-bit ALU

3 Detailed Design Steps with K-maps

3.1 Design Steps

1. The arithmetic unit computes four arithmetic operations (Decrement A, Add with Carry, Subtract, and Transfer A) using a 4-bit full adder and a multiplexer.

2. For the adder, the first input A is fixed, while the second input is either B or c, selected by a multiplexer with the selection bit S_1. For the Decrement A and Transfer A operations, c is selected, where $c = 1111$ for Decrement A and $c = 0000$ for Transfer A. For Add with Carry and Subtract operations, B and B' are selected, respectively.

3. In the arithmetic unit, for the Decrement operation, the adder adds A with $c = 1111$ and $C_{\text{in}} = 0$, effectively resulting in $A - 1$ in signed representation. For the Transfer operation, $c = 0000$ and $C_{\text{in}} = 0$, so the adder outputs $A + 0 = A$. To perform Add with Carry and Subtract operations, the adder adds A with B and B', respectively, with C_{in} set to 1 in both cases.

4. The logical unit performs two logical operations: AND and Complement A. It uses one AND gate and one NOT gate. The output of the logical unit (AND or Complement) is selected by a multiplexer with the selection bit S_0 (0 for Complement, 1 for AND).

5. A multiplexer selects the final output of the ALU. When the selection bit $S_2 = 0$, the output of the arithmetic unit is selected as the final output, and when $S_2 = 1$, the output of the logical unit is selected.

6. The overflow flag, V, and carry flag, C, are computed from the arithmetic unit. During logical operations, $C_{\text{in}} = 0$ and $c = 0000$ are selected as the second input to the adder, so there is no chance of overflow or carry, keeping $V = 0$ and $C = 0$ for logical operations.

7. The zero flag, Z, is computed by adding the four output bits using three OR gates and then inverting $O_0 + O_1 + O_2 + O_3$ with an XOR gate, $X \oplus 1 = \overline{X}$.

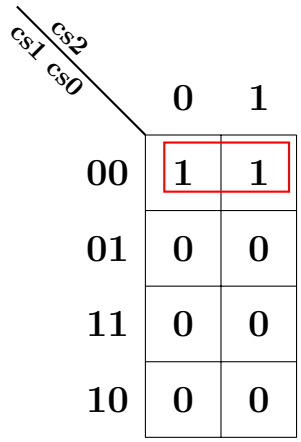
3.2 K-Maps

To determine the intermediate selection bits S_0, S_1, and S_2, we will use the following K-maps and their corresponding simplified equations:

3.2.1 Solving the K-map for S_0:

The bit S_0 serves as the selection input for the multiplexer, which determines the first input to the adder in the arithmetic unit. There are three possible choices for the adder's first input: AB, A, or $\overline{A}$. Here's how each selection works:

AB: Selected in case of the AND operation, using both inputs.
A: Selected for operations like Decrement A, Add with Carry, Subtraction, and Transfer.
$\overline{A}$: Selected in the Complement A operation.



Based on the K-map, there are two minterms for S_0, which we can simplify. Thus, the minimized equation for S_0 is:

$$S_0 = \overline{\mathrm{CS}_1}\,\overline{\mathrm{CS}_0}$$

This simplified equation represents the condition under which the multiplexer selects the input for the adder. To clarify how we select among the three options AB, A, and $\overline{A}$ using a 2×1 multiplexer:

In this setup, we send the following signals to the two inputs of the multiplexer:

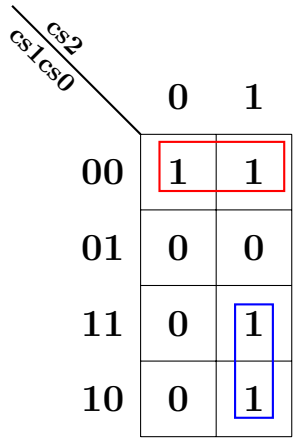
For the first input, we use the expression $A \oplus (CS_2CS_1\overline{CS_0})$. This means that when $CS_2 = 1$, $CS_1 = 1$, and $CS_0 = 0$, the expression evaluates to $\overline{A}$;

in all other cases, it evaluates to A. Thus, we can dynamically select between A and $\overline{A}$ based on the control signals. For the second input, we simply send AB. With this configuration, the multiplexer can choose between AB and the result of $A \oplus (CS_2CS_1\overline{CS_0})$ as the first input to the adder, allowing us to switch between AB, A, and $\overline{A}$ as needed for the different operations.

3.2.2 Solving K-map for S_1

The bit S_1 serves as the selection input for the multiplexer, which determines the second input to the adder in the arithmetic unit. There are four possible choices for the adder's second input: B,$\overline{B}$,0, or 1 . Here's how each selection works:

B: Selected in case of the Add with Carry operation.
$\overline{B}$: Selected for operations like Subtraction
0:Selected for And, Complement, and Transfer operation.
1:Selected for Decrement operation



Based on the K-map, there are four minterms for S_1, which we can simplify. Thus, the minimized equation for S_1 is:

$$S_1 = \overline{\mathrm{CS}_1}\,\overline{\mathrm{CS}_0} + CS_2CS_1$$

This simplified equation represents the condition under which the multiplexer selects the input for the adder. To clarify how we select among the four options B,$\overline{B}$,1 and 0 using a 2×1 multiplexer:

In this setup, we send the following signals to the two inputs of the multiplexer:

For the first input, we use the expression $(CS_1B \oplus CS_0)$. For the second input, we simply send 0. With this configuration, the multiplexer can choose between B and the result of $(CS_1B \oplus CS_0)$ as the second input to the adder, allowing us to switch between B,$\overline{B}$, 0, and 1 as needed for the different operations.

3.2.3 Solving K-map for S_2

The selection bit S_2 is conventionally used to choose between arithmetic and logical operations. However, in our design, we didn't implement it in this way because switching from arithmetic to logical operations was not necessary due to our design pattern. Our two logical operations, AND and Complement A, are handled directly by manipulating the first input. This approach helped us reduce the number of ICs in our design, making S_2 solving a mere formality. Nevertheless, the K-map for S_2 is provided here for completeness:

cs2 / cs1cs0	0	1
00	1	1
01	0	0
11	0	0
10	0	1

Based on the K-map, we identify three minterms for S_2, which we can simplify. The minimized equation for S_2 is therefore:

$$S_2 = \overline{\mathrm{CS}_1}\,\overline{\mathrm{CS}_0} + CS_2\overline{\mathrm{CS}_0}$$

4 Truth Table

For better interpretation of the variables used, refer to Figure 2.

cs2	**cs1**	**cs0**	**Function**	$\mathbf{C_{in}}$	$\mathbf{S_0}$	$\mathbf{S_1}$	$\mathbf{S_2}$
X	0	0	AND	0	1	1	1
X	0	1	Decrement A	0	0	0	0
0	1	0	Add with carry	1	0	0	0
0	1	1	Subtraction	1	0	0	0
1	1	0	Compliment A	0	0	1	1
1	1	1	Transfer A	0	0	1	0

Table 2: Truth Table for Intermediate I/O

5 Block Diagram

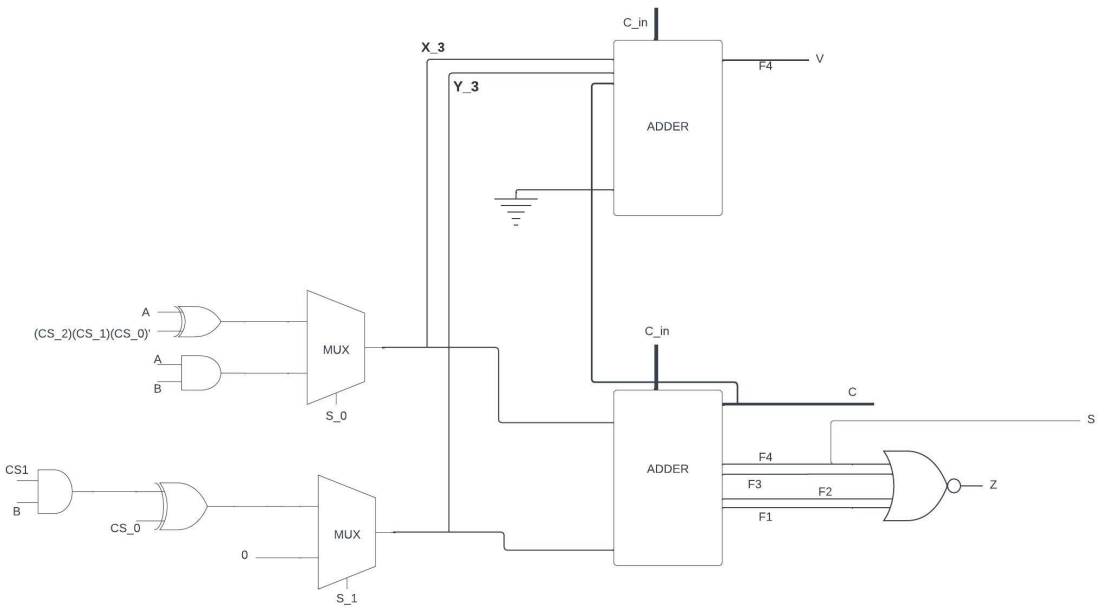


Figure 2: Block Diagram of the circuit

6 Complete Circuit Diagram

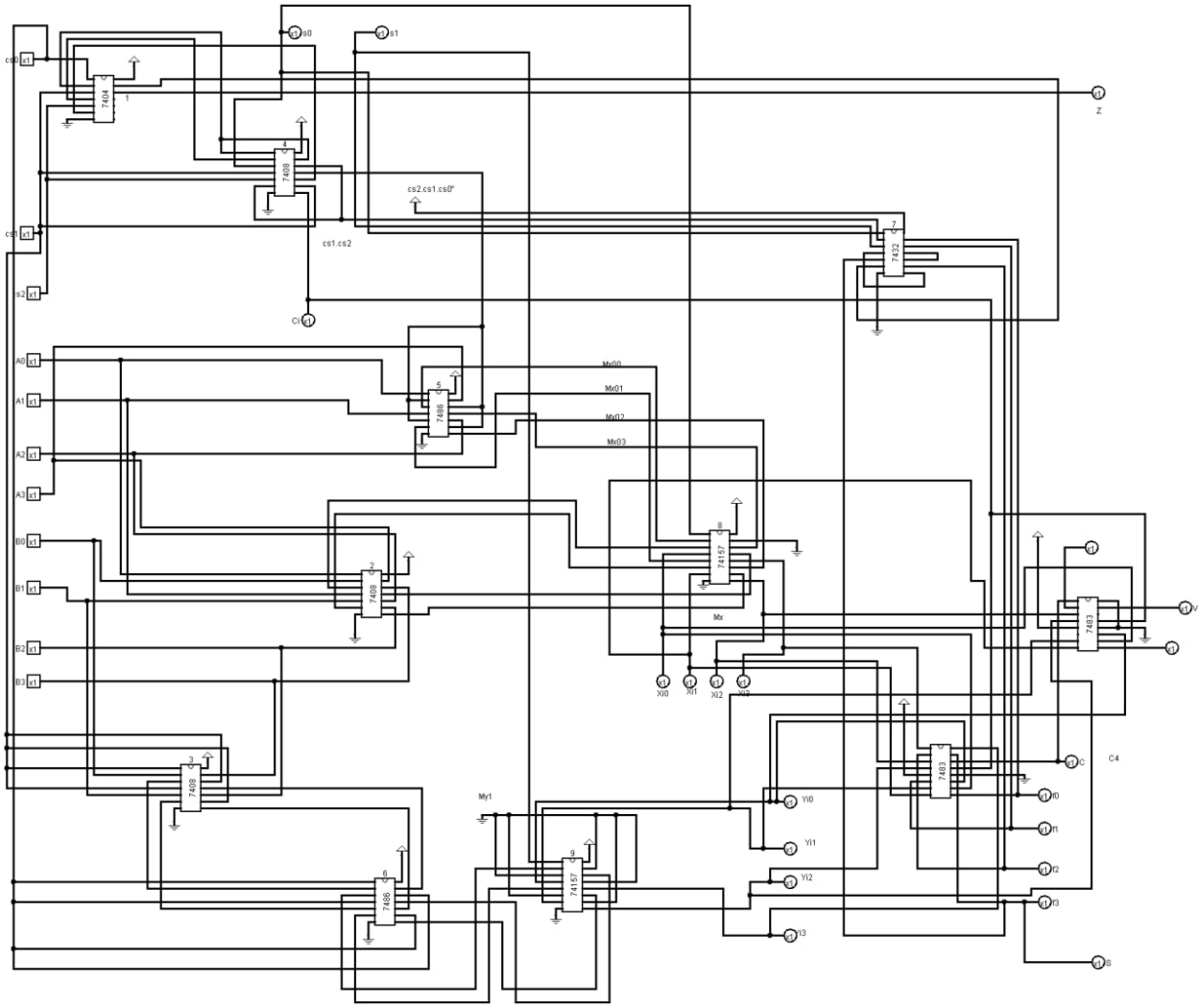


Figure 3: Complete Circuit Diagram

7 ICs Used with Count as a Chart

IC	**Quantity**
IC 7404	1
IC 7408	3
IC 7432	1
IC 7483	2
IC 7486	2
IC 74157	2
Total	11

Table 3: Integrated Circuits Used and Their Counts

8 The Simulator Used along with the Version Number

Logisim- 2.7.1

9 Discussion

In this assignment, we were tasked to implement a 4-bit ALU which performs 4 arithmetic and 2 logical operations.

Through rigorous scrutiny, we had to strive hard to obtain the design with the minimum number of ICs. In achieving this, we had to make optimizations like performing XOR operation for getting **V** flag,instead of using a single operation in 7486 IC,we did it in between our extra used adder,7483 IC's extra one port by giving C_{out} in it's A4 pin, 0 in B4 pin and by giving X1,X2,X3 and Y1,Y2,Y3 in previous we got C3.And such,we were able to do $C3 \oplus C_{out}$ and reduce one IC to take place to use. They were results of multiple runs of redoing the design.

The hardware implementation also posed challenges like planning the placement of different modules. To keep the hardware design clean, we had to connect the wires so that they cross as little length as possible between the connections. Extra effort had to be invested to make the hardware not just working, but aesthetically pleasing. Power and ground connection were done

with caution to prevent IC or other components from getting damaged. After checking all the boxes, we are hopefully successful in implementing the 4-bit Arithmetic and Logic Unit (ALU) with minimum number of ICs.

10 Contribution of Each Member

- **Circuit Design:** Imdadul Hasan Hamim (2105160) and Mezba (2105172) led the design and derivation of the circuit, providing the primary framework. Additional support was provided by team members Siam (2105155) and Alif (2105158).
- **Circuit Assembly and Arrangement:** Siam (2105155) and Hamim (2105160) handled the input components, while Alif (2105158) and Mezba (2105172) were responsible for the output and flag sections of the circuit.
- **Report Writing:** The project report was written by Alif (2105158) and Siam (2105155).