CSE 306
Computer Architecture Sessional

Assignment-1: 4-bit ALU Simulation

Lab Section - A1
Group - 06

15 December, 2023

Members of the Group:

i. 2005020 - Mostafa Rifat Tazwar

ii. 2005025 - Most. Sonia Khatun

iii. 2005027 - Swastika Pandit

iv. 2005029 - MD. Minhajul Islam Fuad

v. 2005030 - Fairuz Mubashwera

1 Introduction

The Arithmetic Logic Unit (ALU) serves as the computational core of a computer's central processing unit (CPU). Responsible for executing arithmetic and logic operations, the ALU performs essential tasks such as addition, subtraction,bitwise logical operations, negation and many more. Its speed and versatility make it a vital component in processing instructions efficiently. ALUs come in various architectures, adapting to different computational demands across applications. To understand modern computing devices, knowing about the ALU – its role and design – is vital.

The ALU we designed consisted of three control signals, through which we facilitated six operations: decrement, subtract with borrow, subtract, negation, bitwise AND, and XOR operations. The arithmetic unit was responsible for managing the arithmetic operations, while our logical unit executed the rest. The control signals supervised the entire ALU operations.

Creating an ALU demands creative thinking from the designer. We employed decoders and multiplexers to link the control signals to our specified operations. Additionally, we repurposed specific ICs to perform operations beyond their intended functions. Our focus was on maximizing the utilization of the resources (ICs) at our disposal, ensuring an optimized design.

We also incorporated 4 status flags , Carry(C), Sign(S), Overflow(V) and Zero(Z) flags, in our design. These flags followed the rules of Assembly Language with some flexibility. They imply :

a) **C** :- C or Carry Flag is the carry out C_{out} of the adder in the ALU. It depends on the carry output of arithmetic operations. It is reset during logical operations.

b) **S** :- S or the Sign bit is the MSB of the output, O_3.

c) **V** :- V is the overflow flag. If the output of an n-bit ALU exceeds the range of -2^{n-1} to $2^{n-1}-1$ then an overflow has occured. V is set when the addition of two positive numbers gives negative output and vice-versa. It is always reset during logical operations.

$$O_3 = X_3 \oplus Y_3 \oplus C_3$$
$$\Rightarrow O_3 \oplus C_3 \oplus O_3 = X_3 \oplus Y_3 \oplus C_3 \oplus C_3 \oplus O_3$$

$$C_3 = X_3 \oplus Y_3 \oplus O_3 \tag{1}$$

$$V = C_3 \oplus C_{\text{out}} \tag{2}$$

$$\therefore V = X_3 \oplus Y_3 \oplus O_3 \oplus C_{out}$$

where C_{out} is the output carry, O_3 is the MSB of the output, X_3 and Y_3 are the MSB of the first and second input of the adder respectively.

d) **Z** :- Z is the Zero flag, which is set when the output of the ALU is zero and reset otherwise.

$$Z = \overline{O_3 \vee O_2 \vee O_1 \vee O_0}$$

2 Problem Specification with Assigned Instructions

Design a 4-bit ALU with three selection bits cs_2 , cs_1 and cs_0 for performing the following operations:

Control Signals			**Function**	**Description**
cs_2	cs_1	cs_0		
0	X	0	Decrement A	A - 1
0	0	1	Subtract with borrow	A + $\overline{B}$ (i.e. A - B - 1)
0	1	1	NEG A	$\overline{A}$ + 1
1	0	0	Subtract	A + $\overline{B}$ + 1 (i.e. A - B)
1	0	1	AND	A $\wedge$ B
1	1	X	XOR	A $\oplus$ B

Table 1: Problem Specification

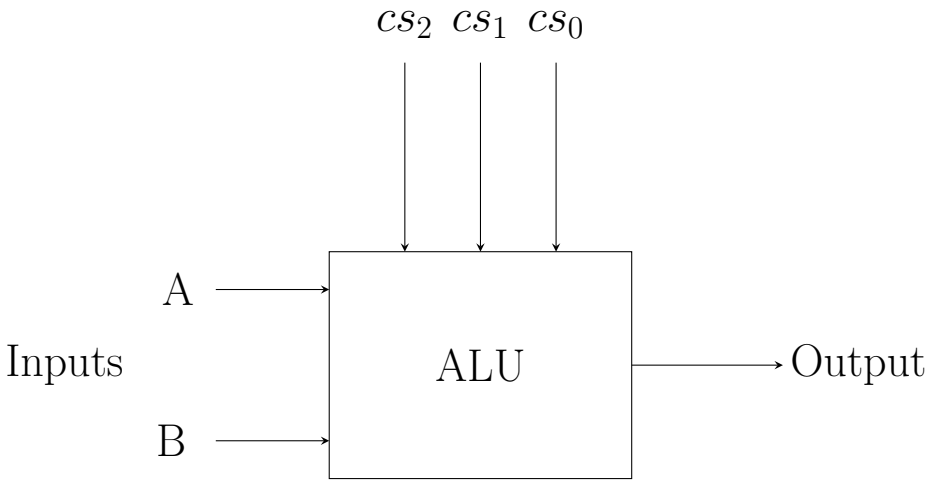

Figure 1: 4-bit ALU

3 Detailed Design Steps with K-maps

3.1 Design Steps

i) Our design comprises 4 parts: arithmetic unit, logical unit, selection control, and status flags.

ii) The arithmetic unit computes 4 arithmetic operations (decrement A, subtract with borrow, negate A and subtract) with the help of a 4-bit parallel adder, basic gates and a multiplexer.

iii) In the adder, the first input is A or $\overline{A}$ which is obtained using 1 XOR IC which complements A when S_3 is set. The second input is $\overline{B}$ or P which is selected by the multiplexer having S_2 selection bit. For negate A, A is complemented; A remains unchanged for rest of the operations. For subtract and subtract with borrow operations, $\overline{B}$ is selected, for decrement A, P = 1111 is selected and for negate A operation, P = 0000 is selected. for Transfer A and c = 1111 for Decrement A.

iv) The adder adds A, $\overline{B}$ with $C_{in} = 0$ and $C_{in} = 1$ for subtract with borrow and subtract operations respectively. P = 0000, $C_{in} = 1$ and $\overline{A}$ is used for negate A, so adder outputs $\overline{A}$ + 0 + 1 = -A. During decrement operation, P = 1111, $C_{in} = 0$, that is adder adds 1111 to A which is equivalent to A - 1 + 0 = A - 1.

v) The logical unit performs 2 logical operations AND, XOR using 1 AND IC, 1 XOR IC, and the output (AND, XOR) of the logical unit is selected by a multiplexer having selection bit S_3 (0 for AND,1 for XOR).

vi) The selection control uses the control signals as inputs, an active low decoder, and basic gates to output the selection variables, C_{in} and intermediate variable P.

vii) A third multiplexer selects the final output of the ALU. For selection bit S2 = 0, the output of the logical unit is the final output, and for S2 = 1, the output of the arithmetic unit is the final output.

viii) The overflow flag, V and carry flag, C is computed from the arithmetic unit. During logical operations, $C_{in} = 0$ and P = 0000 is selected as the second input of the adder. So there is no chance of overflow and carry, which ensures V = 0 & C = 0 for logical operations.

ix) Zero flag, Z is computed from the 4 output bits using 3 OR gates and then inverting $O_3 \vee O_2 \vee O_1 \vee O_0$ by 1 XOR gate($Z \oplus 1 = \overline{Z}$).

x) The sign flag S and carry flag C are obtained from O_3 and C_{out} respectively.

3.2 K-maps

3.2.1 K-map for S_3

S_3 is the selection bit which complements the first input A_i using a XOR IC.

cs_2 \ cs_1,cs_0	00	01	11	10
0	0	0	1	0
1	0	0	0	0

So, there is one minterm. We will be using a decoder (active low output) to implement the minterm.

$$S_3 = \overline{D_3}$$

3.2.2 K-map for S_2

S_2 is the selection bit for the second input of the adder in the arithmetic unit. It chooses either $\overline{B_i}$ or P. The former is chosen when S_2 is reset. The later P is chosen when S_2 is set.

cs_2 \ cs_1,cs_0	00	01	11	10
0	1	0	1	1
1	0	1	1	1

Hence, we get

$$S_2 = D_4 \wedge D_1$$

3.2.3 K-map for S_1

S_1 is the selection bit for logical operation.It is set during xor and reset during and operation.

cs_2 \ cs_1,cs_0	00	01	11	10
0	1	1	1	1
1	1	0	1	1

So the expression is,

$$S_1 = D_5$$

3.2.4 K-map for S_0

S_0 is the selection bit for choosing between arithmetic and logical operations. For arithmetic operations, S_0 is set and reset for the other.

cs_2 \ cs_1,cs_0	00	01	11	10
0	1	1	1	1
1	1	0	0	0

As we are implementing this using active low decoder(using maxterms) we get,

$$S_0 = D_5 \wedge D_6 \wedge D_7$$

3.2.5 K-map for P

P is the alternative second input of the adder for arithmetic unit. It will be set during decrement, reset for negation and logical operations.

cs_2 \ cs_1,cs_0	00	01	11	10
0	1	1	0	1
1	1	0	0	0

Although the expression is a product of 4 maxterms, we already get AND operation of 3 out of 4 terms using S_0. Hence, we optimize and get,

$$S_0 = D_3 \wedge D_5 \wedge D_6 \wedge D_7 = D_3 \wedge S_0$$

3.2.6 K-map for C_{in}

C_{in} is the carry input of the adder for arithmetic unit. It will be set only during negation and subtraction.

cs_2 \ cs_1,cs_0	00	01	11	10
0	0	0	1	0
1	1	0	0	0

At this point, we have one gate of OR IC and one of NOT IC remaining in our ALU design. We already have $\overline{D}_3$ from one not gate of that IC. So for optimization, we use ***De Morgan's law*** and get

$$C_{in} = \overline{D_4 \wedge D_3} = \overline{D_4} \vee \overline{D_3}$$

4 Truth Table

cs_2	cs_1	cs_0	X_i	Y_i	C_{in}	S_3	S_2	S_1	S_0	P
0	X	0	A_i	1	0	0	1	1	1	1
0	0	1	A_i	$\overline{B_i}$	0	0	0	1	1	1
0	1	1	$\overline{A_i}$	0	1	1	1	1	1	0
1	0	0	A_i	$\overline{B_i}$	1	0	0	1	1	1
1	0	1	$A_i \wedge B_i$	0	0	0	1	0	0	0
1	1	X	$A_i \oplus B_i$	0	0	0	1	1	0	0

Table 2: Truth Table for selection bits, intermediate variable and carry input

5 Block Diagram

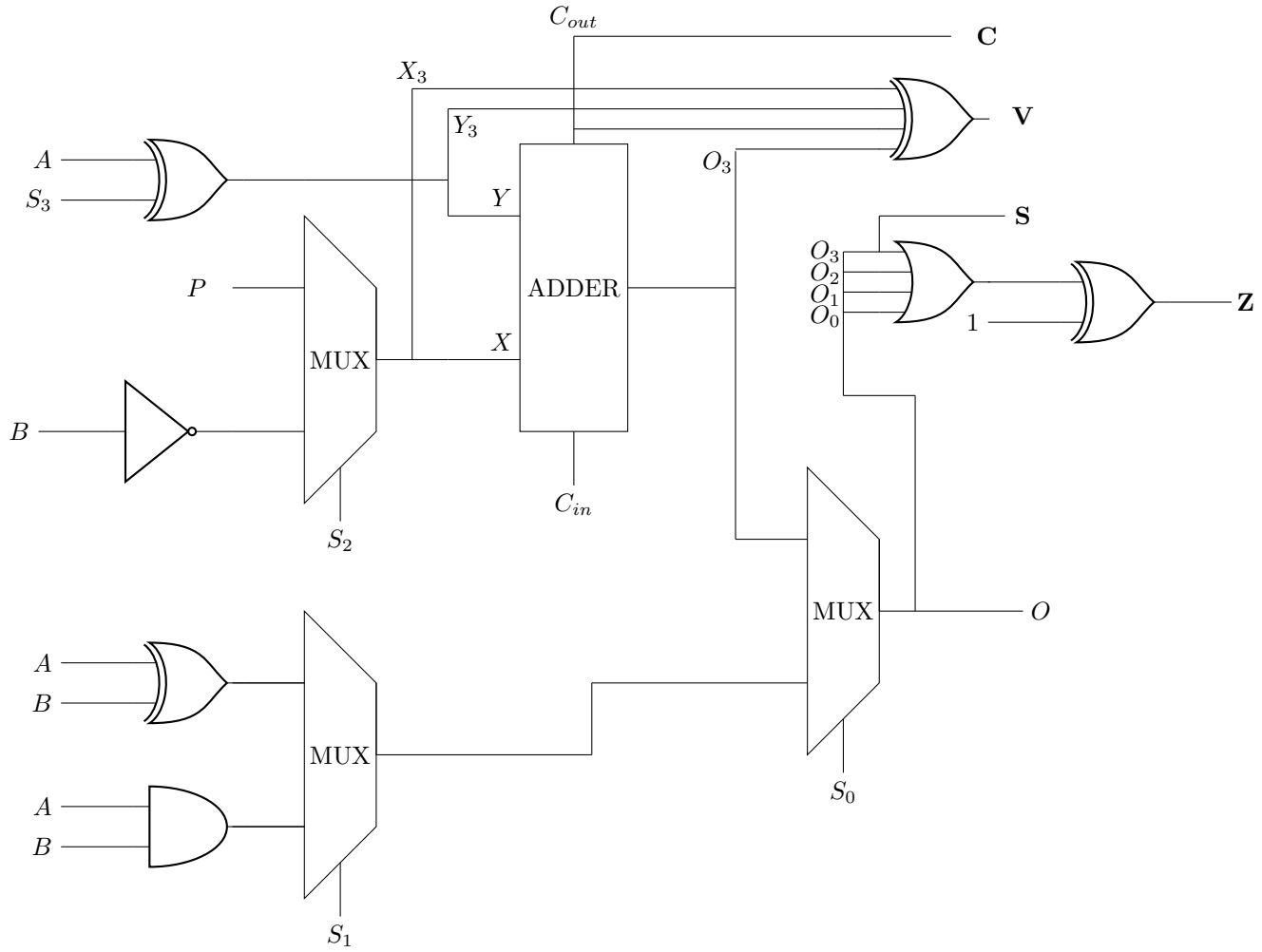

Figure 2: Block Diagram of ALU

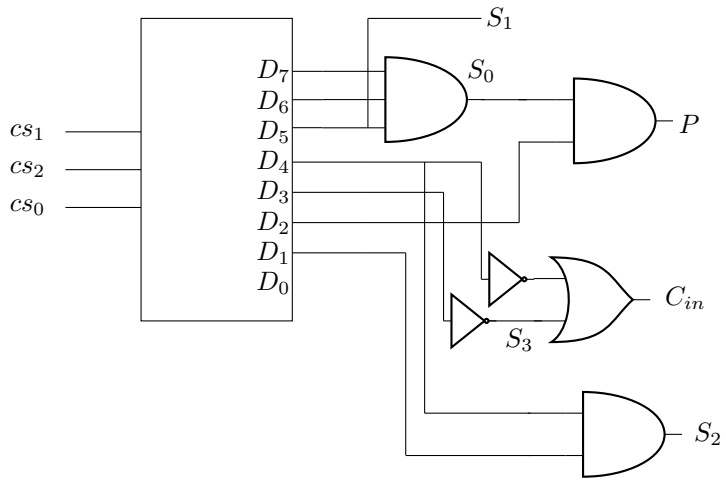

Figure 3: Selection and Intermediate Variables

6 Complete Circuit diagram

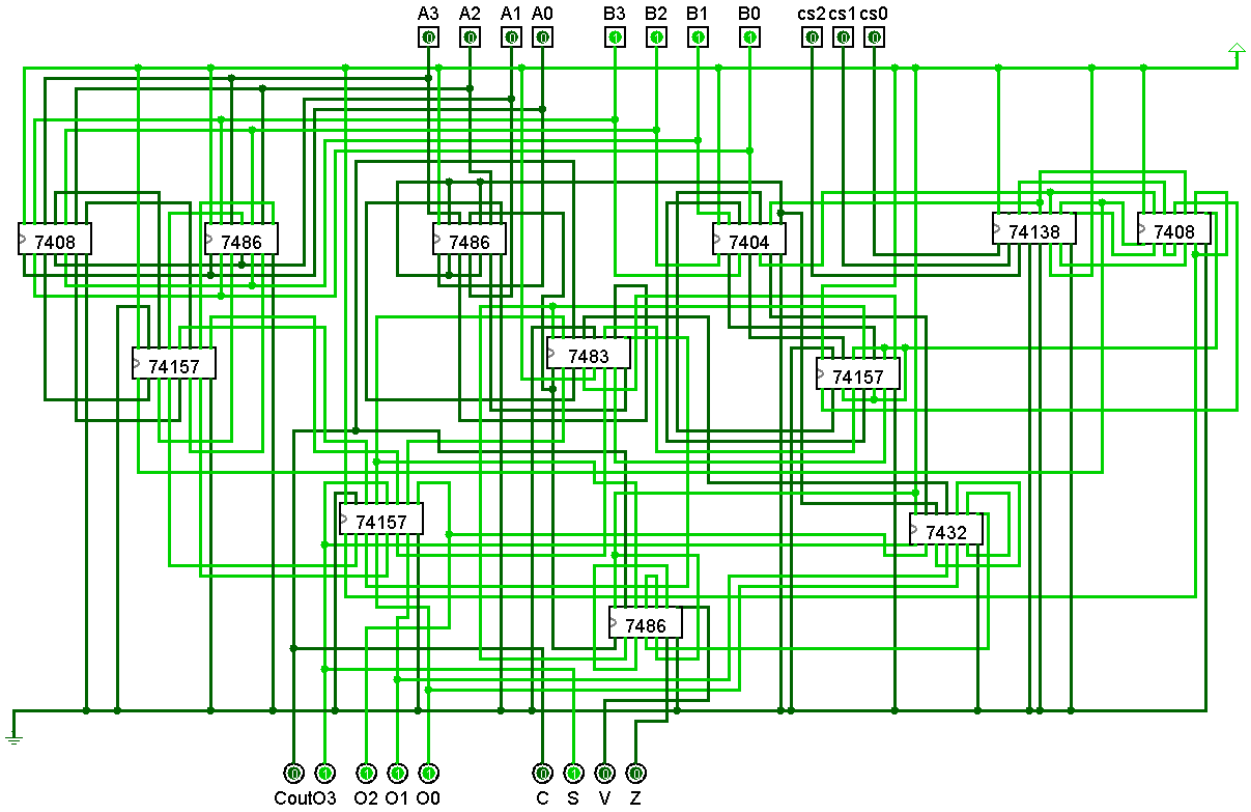

Figure 4: Complete ALU circuit

7 ICs used with count as a chart

IC	Number of ICs
74138	1
7483	1
74157	3
7432	1
7486	3
7404	1
7408	2
Total	12

Table 3: ICs and their number

8 The Simulator Used along with the Version Number

Logisim - 2.7.1

9 Discussions

The entire project was a pedagogic experience. Several difficulties had to be faced throughout the process.

Firstly, our circuit design needed a certain degree of creativity. The complement of an input was needed depending on the control signals. So, instead of using a MUX and hex inverter, a XOR IC was used. In addition, an existing selection bit was reused to implement another selection bit. Moreover, techniques like De Morgan's law to simplify and complementing using remaining slot of XOR IC instead of another hex inverter were used. Thus, maximum utilization of the existing ICs were ensured.

Secondly, some difficulties did arise while software simulation using **Logisim**. Changes had to be made in the 7400-lib.circ. The C_{out} pin of the IC7483 in the library was faulty. This problem was corrected using a seperate IC7483.circ file in our final ALU software simulation.

Finally, we addressed challenges in the hardware implementation by correcting issues with LEDs, where the use of different types initially caused a power drop. Overheating problems due to connecting LEDs without resistances were resolved through the implementation of proper corrections. Initial misconnections of mini push button switches, providing opposite outcomes, were corrected by the team. The challenge of maintaining a stable power flow across six breadboards was successfully addressed through multiple correction attempts. In debugging the complex circuit, the impracticality of manual methods led us to correct the approach by employing logic design tools for effective resolution.

10 Contribution of Each Member

- 2005020 : Mostafa Rifat Tazwar
  - Circuit design and optimization
  - Software simulation of ALU circuit
  - Report writing
- 2005025 - Most. Sonia Khatun
  - Hardware implementation
- 2005027 - Swastika Pandit
  - Hardware implementation
- 2005029 - MD. Minhajul Islam Fuad
  - Hardware implementation
- 2005030 - Fairuz Mubashwera
  - Hardware implementation
  - Discussion of report