CSE 210

Computer Architecture Sessional
Assignment-1: 4-bit ALU Simulation

Section: B1
Group: 01

Team Members:
2105068- Anika Morshed
2105069 - Sheikh Iftikharun Nisa
2105081 - Diganta Saha Tirtha
2105082 - Jahedul Islam Nayeem
2105084 - Dibya Jyoti Sarkar

Report Prepared by:
2105068 - Anika Morshed

September 30, 2024

1 Introduction

An Arithmetic Logic Unit (ALU) is a key part of a CPU, executing arithmetic and logical operations like addition, subtraction, AND, and OR. It processes binary data (0s and 1s) to perform calculations and decisions, directly influencing a computer's performance.

The ALU consists of two main units: Arithmetic and Logic. It operates on 4-bit inputs, with three control inputs selecting the operation. The output includes 4 bits and four status flags—Carry (C), Zero (Z), Overflow (V), and Sign (S)—indicating specific results described below:

C: C is the c_{out} of the adder of the ALU which denotes the overflow in unsigned operations. $C = 1$ indicates output carry being 1 and 0 otherwise.

S: The most significant bit of the output of the adder. For signed operation, the sign of the result can be determined from this.

Z: If the result of the adder output 0 i.e. all 4 bits are 0 then Z bit becomes 1.

V: V denotes the overflow in signed operation. $V = 1$ means the adder result exceeds the range of the 4 bits.We can find V using,

$$V = C_3 \oplus C_{out}$$

C_3 is the carry generated during the third bit addition and C_{out} is the output carry of the adder.

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	Description
cs2	cs1	cs0		
X	0	0	Add	A + B
0	0	1	Sub	A - B
X	1	0	Transfer A	A
0	1	1	OR	$A \vee B$
1	0	1	Increment A	A + 1
1	1	1	NEG A	- A

Table 1: Problem Specification

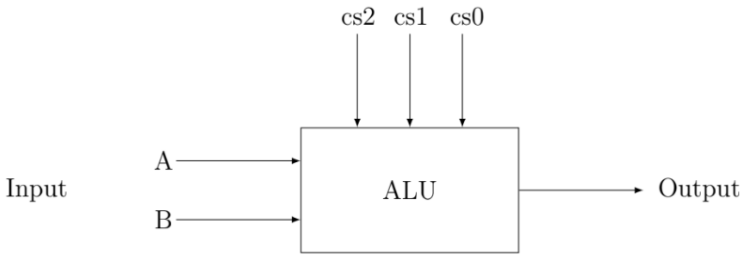


Figure 1: 4-bit ALU

3 Design steps with K-maps

3.1 Design Steps

1. The arithmetic unit computes 5 arithmetic operations with the help of a 4-bit full adder and a multiplexer.

2. For the adder, the first input A is fixed to be A_i except for the combinations $cs_2cs_1cs_0 = 111$ and $cs_2cs_1cs_0 = 011$. For the first combination, A_i' is selected, and for the latter, $A_i \vee B_i$ is selected.A 2x1 multiplexer is used for each bit to select from the two options.

3. To implement the flags, it is observed that the overflow flag requires previous carry from the adder which can not be directly found from a adder IC. So the adder is split in to 2 parts where one adder adds only the 3 bits and another adder only adds the MSB's. Dividing the adder into two parts it only a design choice. The carry generated in the first adder is used as the carry in for the second adder.

4. The overflow bit, V is determined from $C_3 \oplus C_{out}$.

5. Zero bit, Z is computed by adding the 4 output bits using 3 OR gates and then inverting it using XOR gate to use the leftover gates in the IC.

6. Carry bit, C is simply the C_{out} of the second adder which is none other than the second output bit of it.

7. The sign fbit, S can be found from O_4.

3.2 K-maps

3.2.1 K-map for S_1

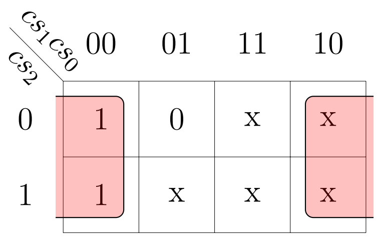
S_1 is the selection bit for the multiplexer that selects the first input of the adder.

$cs_2 \backslash cs_1cs_0$	00	01	11	10
0	0	0	$\boxed{1}$	0
1	0	0	0	0

$$S_1 = cs_2' cs_1 cs_0$$

3.2.2 K-map for S_2

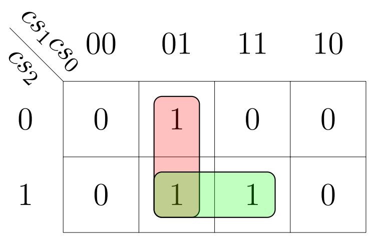
S_2 is the selection bit for the MUX that selects the second input of the adder.



$$S_2 = cs_0'$$

3.2.3 K-map for C_{in}

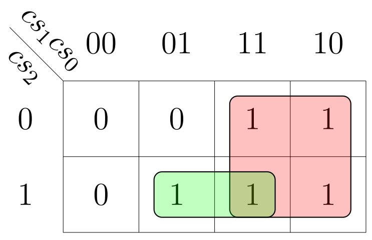
C_{in} is the input carry input of the adder.



$$C_{in} = cs_2\, cs_0 + cs_0\, cs_1' = cs_0\,(cs_2' cs_1)'$$

3.2.4 K-map for en_2

en_2 is the enable bit for the multiplexer.



$$en_2 = cs_1 + cs_0 cs_2$$

4 Truth Table

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

$cs2$	$cs1$	$cs0$	Function	X_i	S_1	Y_i	S_2	en_2	C_{in}
X	0	0	Add	A_i	0	B_i	1	0	0
0	0	1	Sub	A_i	0	B_i'	0	0	1
X	1	0	Transfer A	A_i	0	0	x	1	0
0	1	1	OR	$A \vee B$	1	0	x	1	0
1	0	1	Increment A	A_i	0	0	x	1	1
1	1	1	NEG A	A_i'	0	0	x	1	1

Table 2: Truth Table for Intermediate I/O

5 Block Diagram

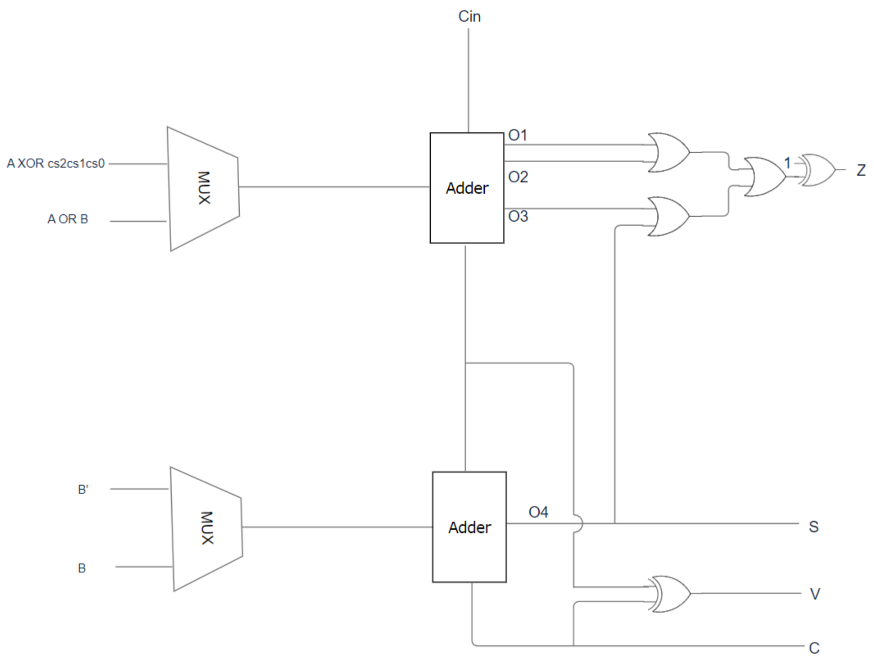


Figure 2: Block diagram for 4-bit ALU

6 Complete Circuit Diagram

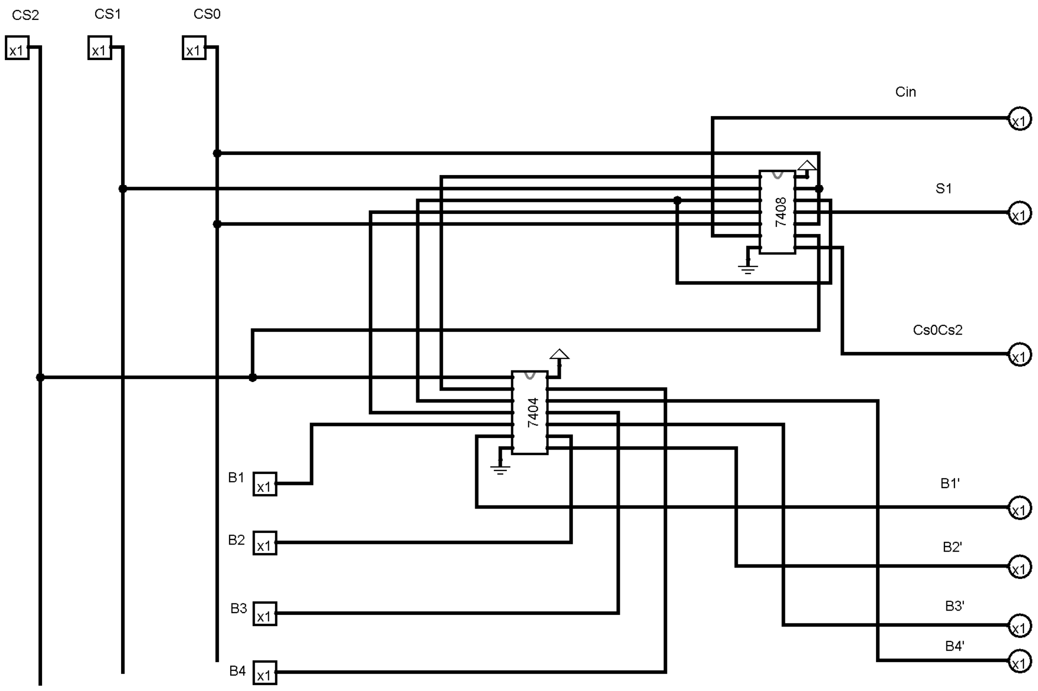


Figure 3: Selection Bit Circuit

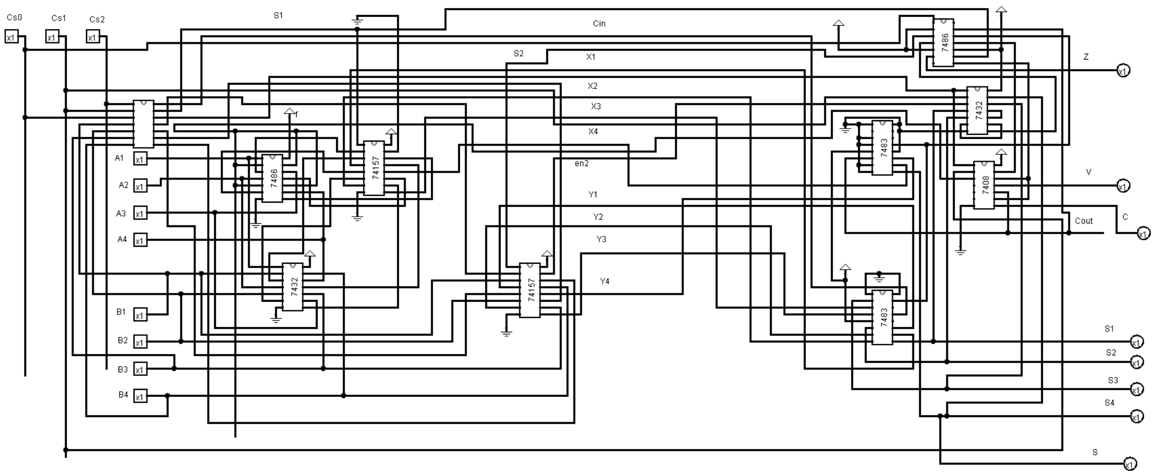


Figure 4: 4-bit ALU

7 ICs Used with Count as a Chart

IC	Quantity
IC74157	2
IC7432	2
IC7408	2
IC7486	2
IC7404	1
IC7483	2
Total	11

Table 3: IC count

8 The Simulator Used along with The Version Number

Simulator used: Logisim-win-2.7.1

9 Discussion

The final design of the Arithmetic Logic Unit (ALU) achieved a total IC count of 11, a result made possible through strategic and efficient design decisions. To further optimize the design and eliminate the need for an additional IC7404, the XOR gate's properties were utilized effectively. In many instances, intermediate values were strategically reused across different parts of the circuit, contributing to a reduction in both the number of ICs and the amount of wiring required.

The design process underwent multiple iterations to ensure optimal efficiency. Prior to commencing hardware implementation, the software model was rigorously tested multiple times by various team members to ensure the elimination of errors and the handling of edge cases. Hardware implementation was initiated only after confirming the full functionality of the software.

The hardware implementation phase required meticulous attention to details, especially regarding component functionality and connection integrity. Each component was individually tested before being integrated into the circuit. Connections were made with precision to maintain the circuit's integrity. Wire management was a key focus, with efforts directed at keeping the breadboard layout clean and easy to interpret. Each subpart of the circuit was thoroughly tested upon completion to verify its correctness.

Through repeated design iterations and efficient strategies, the resulting ALU represents a highly optimized implementation for the intended task.

10 Contribution of Each Member

2105068: contributed in the software designing, calculated Y_i, en_2, S_2 for IC optimisation, partially contributed in other calculations, testing of software implemetation and hardware implementation, testing, debugging.

2105069: contributions in partial implementations of hardware, calculation of X_i, partial calculations of status bits, IC optimisation, testing of both the software and hardware implementations.

2105081: contributions in calculation of C_{in}, IC optimisation, software implementation testing and hardware implementation, testing and debugging

2105082: contributed in the hardware implementation, the calculation of x_{in}, software testing and hardware implementation, testing and debugging.

2105084: contribution in the software design implementation, software modularization and testing of both software and harware.