CSE 306
Computer Architecture Sessional

Assignment-1: 4-bit ALU Simulation

Section - A1
Group - 01

Members of the Group:

i 1905001 - Mohammad Sadat Hossain

ii 1905002 - Nafis Tahmid

iii 1905004 - Asif Azad

iv 1905005 - Md. Ashrafur Rahman Khan

v 1905008 - Shattik Islam Rhythm

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(flages) 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:

CF: CF 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.

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

VF: 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 VF is set. After any logical operation, it is cleared.

$$\begin{aligned} S_3 &= A_3 \oplus I_3 \oplus C_3 \\ \implies C_3 &= S_3 \oplus A_3 \oplus I_3 \end{aligned} \tag{1}$$

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

Combining 1 and 2,

$$V = A_3 \oplus I_3 \oplus S_3 \oplus C_{out}$$

Here, I_3 is either B_3 from input (MSB of B), or c depending on selected operation, S_3 is the MSB of the adder output and C_{out} is the output carry of the adder.

SF: 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	Description
cs2	cs1	cs0		
0	X	0	Add	$A + B$
0	0	1	AND	$A \cap B$
0	1	1	Transfer A	Output is A
1	0	0	Decrement A	$A - 1$
1	0	1	XOR	$A \oplus B$
1	1	X	Add with Carry	$A + B + 1$

Table 1: Problem Specification

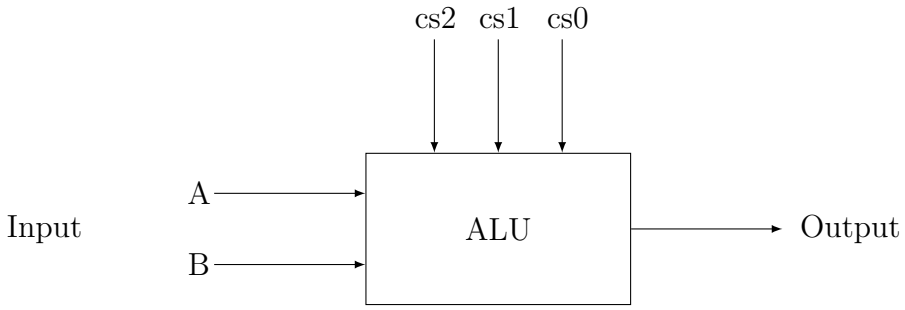


Figure 1: 4-bit ALU

3 Detailed Design Steps with K-maps

3.1 Design Steps

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

2. For the adder, the first input A is fixed; the second input is B or c which is selected by the multiplexer having S_0 selection bit. For Add and Add with Carry operations, B is selected whereas for Transfer A and Decrement A operations, c is selected. $c = 0000$ for Transfer A and $c = 1111$ for Decrement A.

3. In the arithmetic unit, the adder adds A, B with $C_{in} = 0$ and $C_{in} = 1$ for Add, Add with Carry operations respectively. $c = 0000, C_{in} = 0$ for transfer operation, so adder outputs $A + 0 + 0 = A$. During decrement operation, $c = 1111$, $C_{in} = 0$ that is adder adds 1111 to A which is basically equivalent to $A - 1 + 0 = A - 1$ in signed representation.

4. 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_1 (0 for XOR, 1 for AND).

5. A third multiplexer selects the final output of the ALU. For selection bit $S_2 = 0$, the output of the arithmetic unit is the final output and for $S_2 = 1$, the output of the logical unit is the final output.

6. The overflow flag, VF and carry flag, CF is computed from the arithmetic unit. During logical operations, $C_{in} = 0$ and $c = 0000$ is selected as the second input of the adder. So there is no chance of overflow and carry keeping $VF = 0, CF = 0$ for logical operations.

7. Zero flag, ZF is computed by adding the 4 output bits using 3 OR gates and then inverting $O_0 + O_1 + O_2 + O_3$ by 1 XOR gate($X \oplus 1 = \overline{X}$).

3.2 K-maps

We will be following Table 2 to construct the K-maps for intermediate selection bits.

3.2.1 K-map for S_0

S_0 is the selection bit for the multiplexer that selects the second input of the adder in the arithmetic unit. There are two alternatives for the second input of the adder B or c. B is the second user input which will be selected in case of add and add with carry operation. c can be 1111 or 0000 and will be used respectively for decrement and transfer operations. (c will also be used in case of logical operations to maintain the carry and overflow flags.)

cs2cs1 \ cs0	0	1
00	1	0
01	1	0
11	1	1
10	0	0

So, there are four minterms. We will be using decoders (active low output) to implement the minterms.

$$S_0 = D_0 + D_2 + D_6 + D_7 = \overline{\overline{D_0}.\overline{D_2}} + \overline{\overline{D_6}.\overline{D_7}}$$

3.2.2 K-map for S_1

S_1 is the selection bit for the multiplexer which selects between XOR and AND operation inside logic unit.

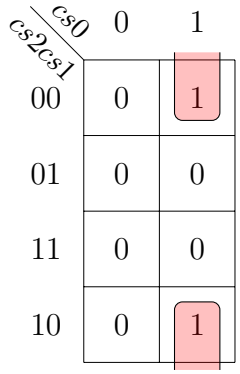
cs2cs1 \ cs0	0	1
00	X	1
01	X	X
11	X	X
10	X	0

We can easily express S_1 with only one minterm. But we will use the maxterm instead as our decoder is active low.

$$S_1 = \overline{D_5}$$

3.2.3 K-map for S_2

S_2 is the selection bit for the multiplexer that selects between the output of the arithmetic and logic unit which eventually is the ALU output.

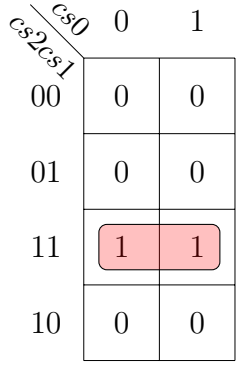


We will do similar treatment as S_0.

$$S_2 = D_1 + D_5 = \overline{\overline{D_1}.\overline{D_5}}$$

3.2.4 K-map for C_{in}

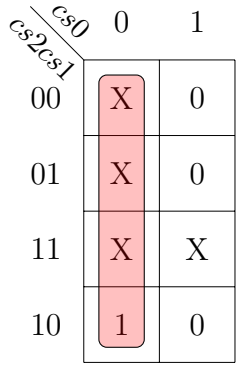
It is the input carry bit of the adder used inside arithmetic unit. It will be 1 only at the time of Add with carry operation. (At the time of logical operation, we will design the selection bits to select transfer operation in arithmetic unit to ensure the flags are in correct state. So, C_{in} will be 0 in those cases too.)



$$C_{in} = D_6 + D_7 = \overline{\overline{D_6}.\overline{D_7}}$$

3.2.5 K-map for c

c is the alternative input for the second input of the adder inside Arithmetic unit. it will be 1 in case of decrement operator and 0 in case of transfer operation. (c will be input to all 4 bits of second input of the adder when selected.)



As there can be only one bit in the simplified expression, we do not need decoder output in this case.

$$c = \overline{cs0}$$

4 Truth Table

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

cs2	cs1	cs0	Function	c	C_{in}	S_0	S_1	S_2
0	0	0	Add	X	0	1	X	0
0	0	1	AND	0	0	0	1	1
0	1	0	Add	X	0	1	X	0
0	1	1	Transfer A	0	0	0	X	0
1	0	0	Decrement A	1	0	0	X	0
1	0	1	XOR	0	0	0	0	1
1	1	0	Add with Carry	X	1	1	X	0
1	1	1	Add with Carry	X	1	1	X	0

Table 2: Truth Table for Intermediate I/0

5 Block Diagram

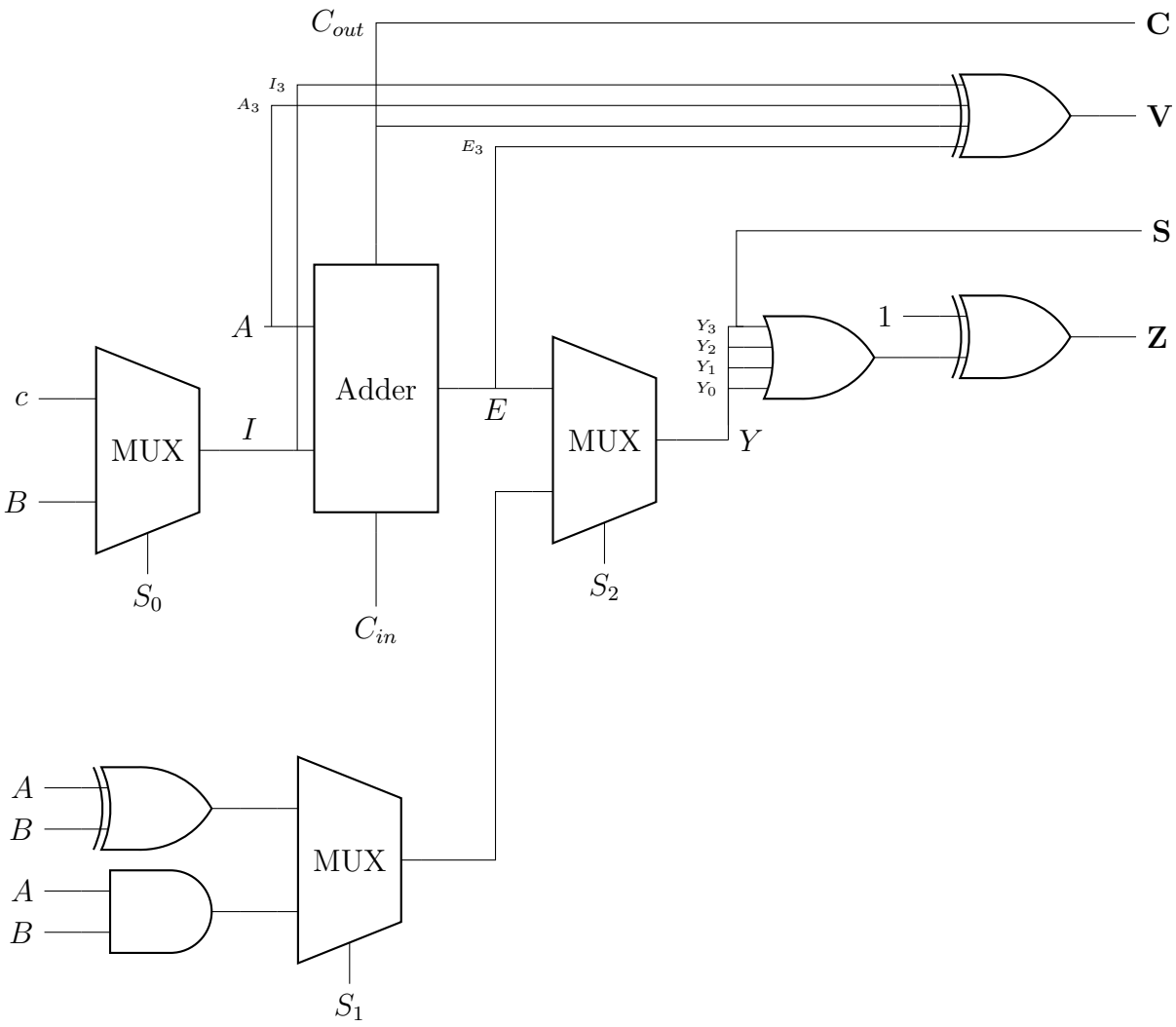


Figure 2: Block Diagram of ALU

6 Complete Circuit Diagram

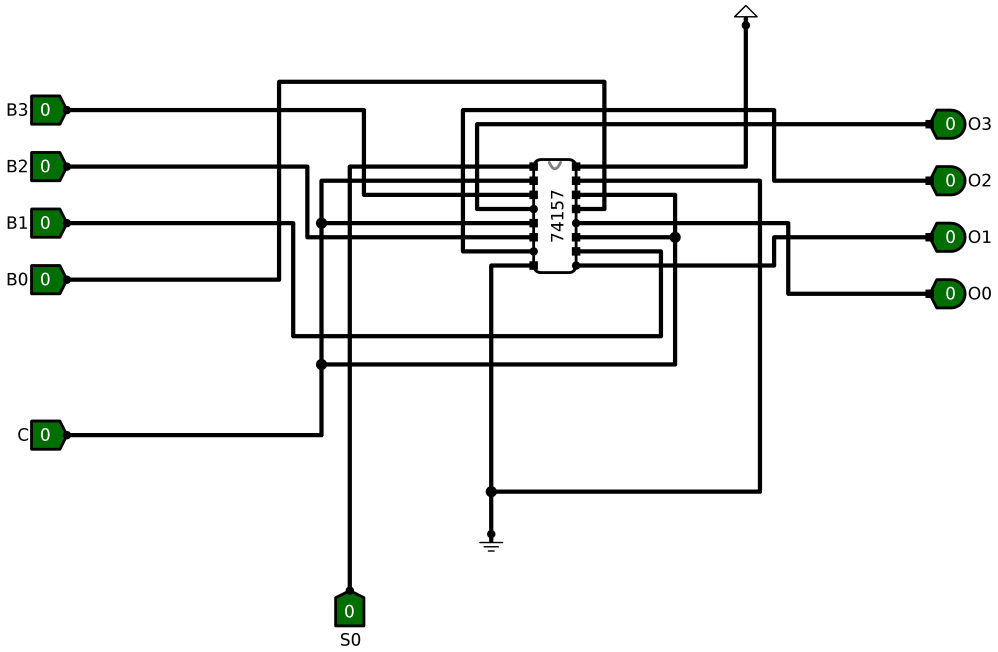


Figure 3: Input Processing for Arithmetic Unit

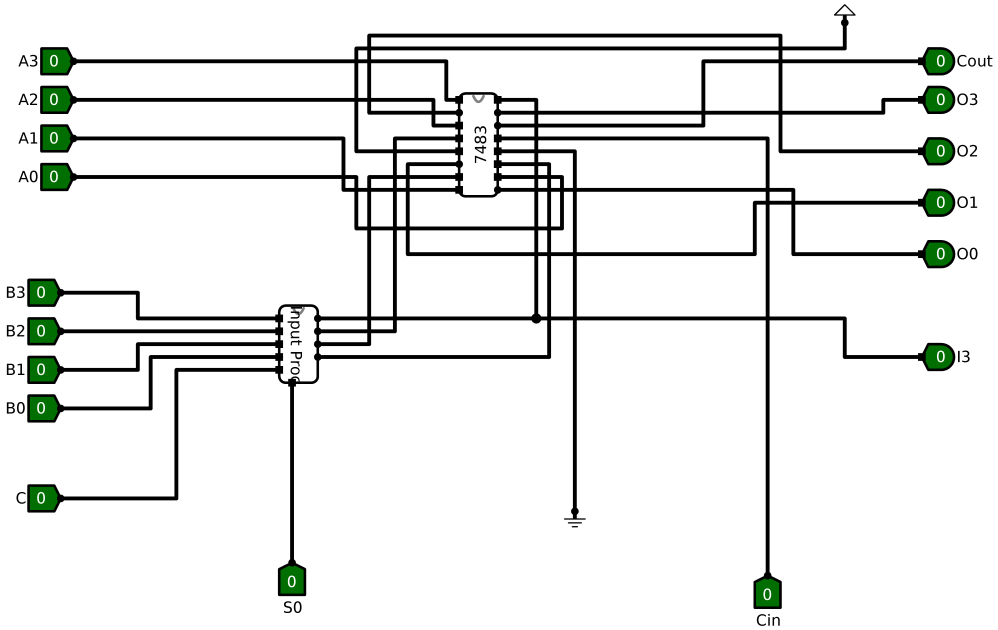


Figure 4: Arithmetic Unit

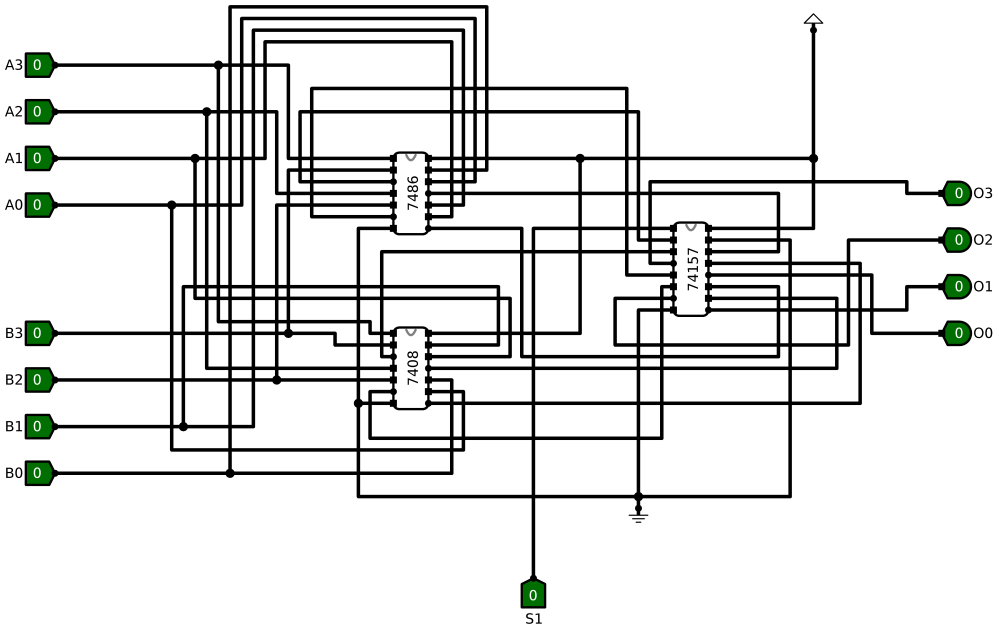


Figure 5: Logic Unit

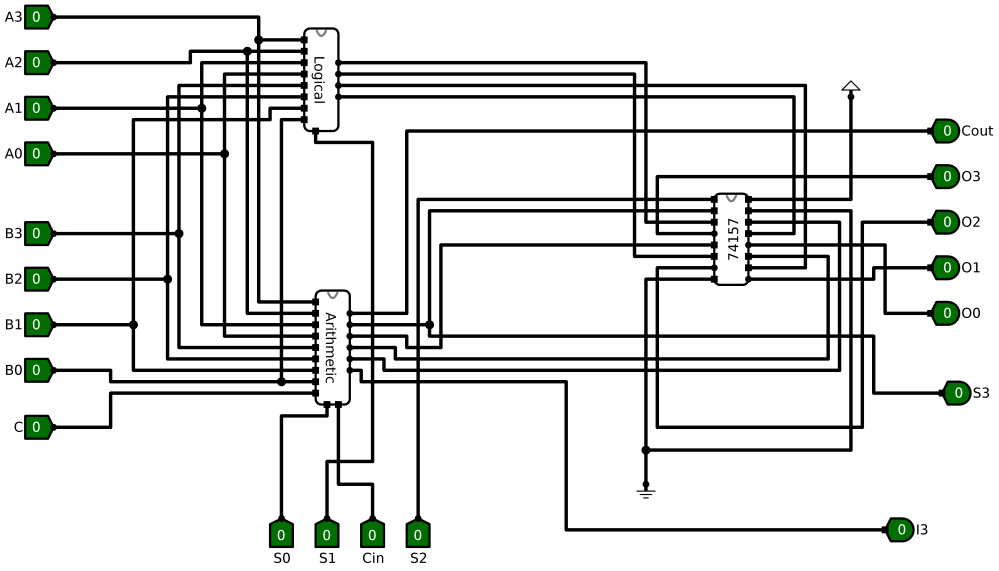


Figure 6: Multiplexed Arithmetic and Logic Unit

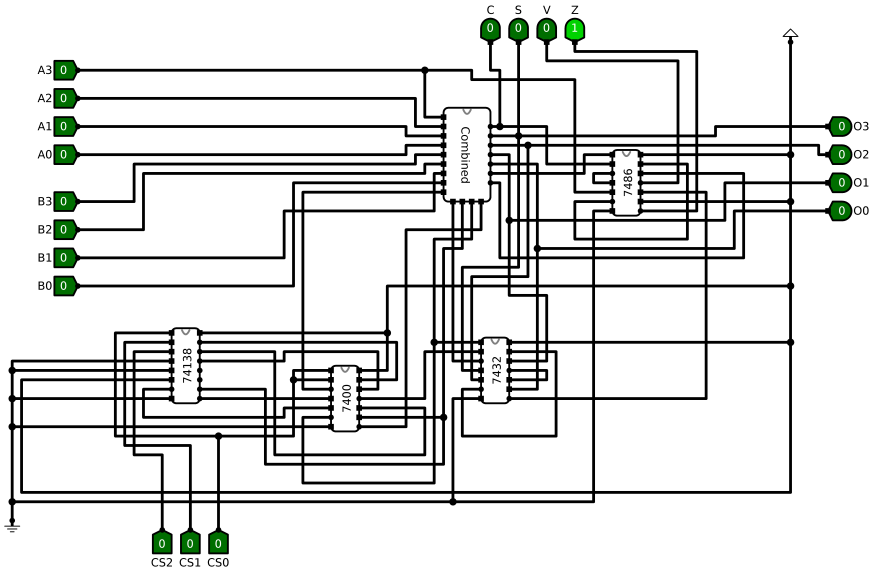


Figure 7: The ALU

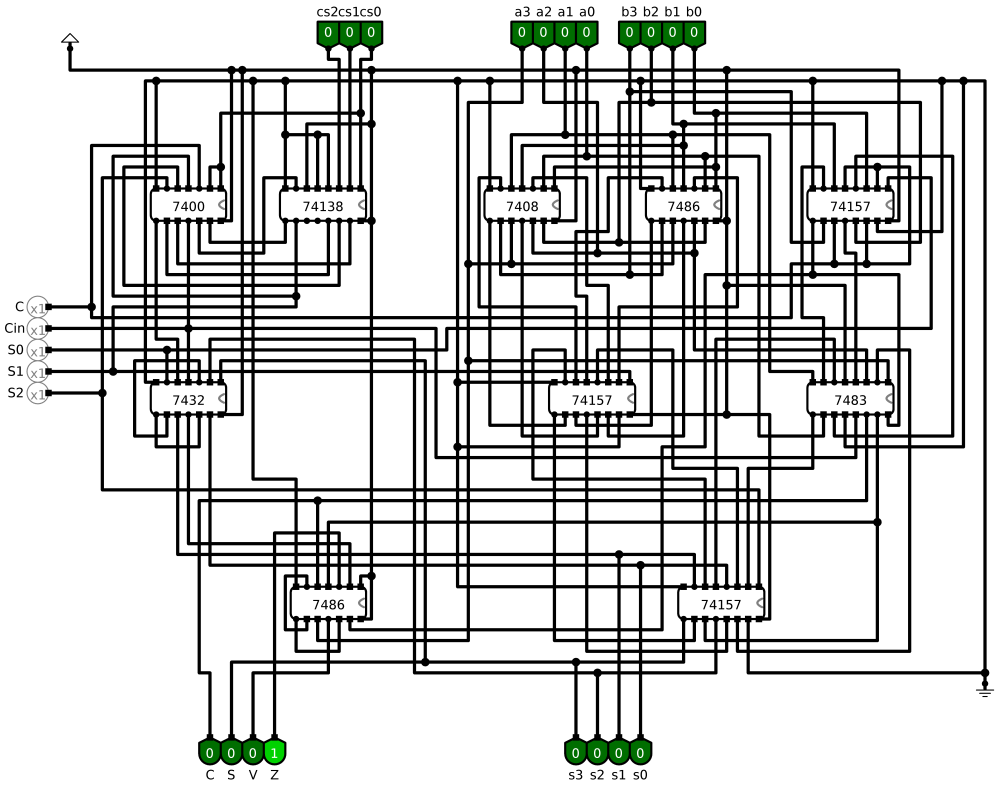


Figure 8: The Complete Design

7 ICs Used with Count as a Chart

IC	Quantity
IC 7400	1
IC 7408	1
IC 7432	1
IC 7483	1
IC 7486	2
IC 74138	1
IC 74157	3
Total	10

Table 3: ICs Used with Quantity

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 the logical NOT operation with IC7486 (XOR) or manipulating one of the IC7483 (Adder) inputs to achieve various arithmetic operations. 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.