BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

DHAKA, BANGLADESH

4 Bit ALU Design

Course No: CSE306

Couse Title: Computer Architecture Sessional

Session: July,2022

Submitted By:

1905042- Rakibul Hasan Rafi

1905046- Niaz Rahman

1905047- Rakib Abdullah

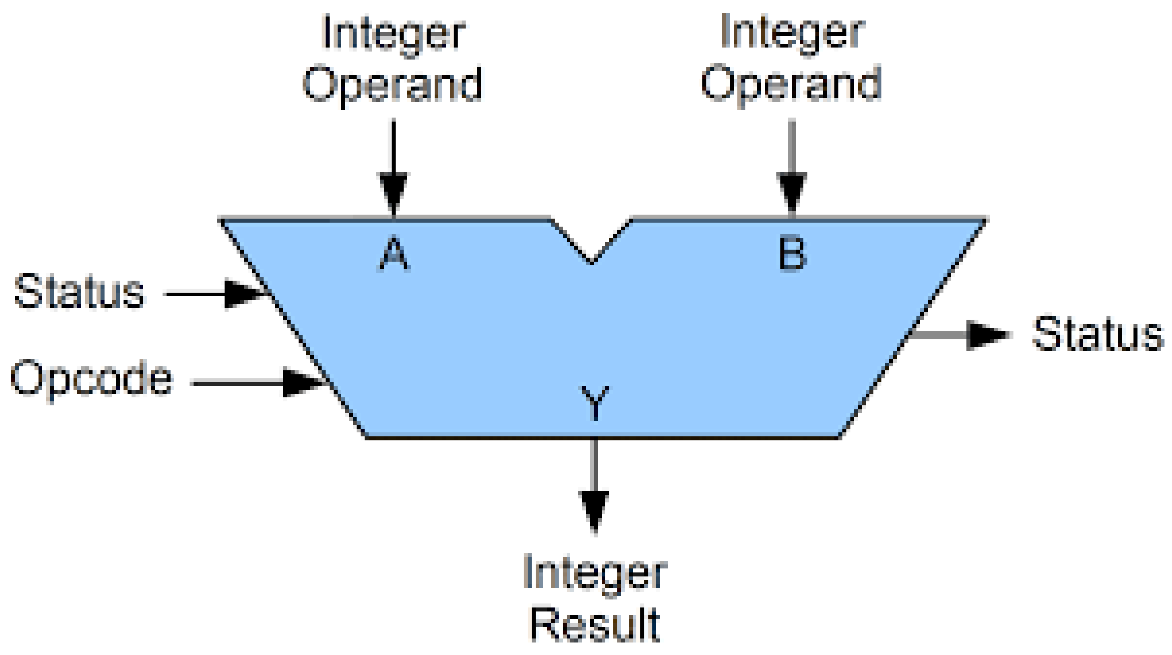
1905048- Md. Al-Amin Sany

1905052- Bijoy Ahmed Saiem

Introduction:

AlU is a major component of CPU

An arithmetic logic unit is a major component of a computer's central processing unit which performs arithmetic and logic operation. It has control unit which defines which operation is to be done. If there are 'k' selection bits we can perform 2^k distinct operation. An operation may be implemented in a processor either with a single operation or a sequence of operation. The control routes the source information from registers into the inputs of ALU. Then ALU performs the operation.



The main component of ALU is a parallel binary adder. By controlling different inputs, it is possible to obtain different outputs of operation. ALU can perform two types of information. It can perform arithmetic as well as logic operation. Logic operations are bitwise where arithmetic operations are sequential. But both the operation can be performed using adder by controlling selection bits and input of adder. There is a combinational part which controls all the operation and input. Decoder, MUX, AND, OR circuits are used to make the combinational part.

Steps involved implementing the ALU:

1. Designing the circuit depending on the operation.
2. Hardware implementation based on the simulation.

Problem Specifications:

In our ALU, there are two inputs and three selection bits. Both the input are four bits. By controlling three bits, six operations are to be performed. By performing, the output will be shown which is also four bits. Again, four flags will be displayed which are Carry(C), Overflow(V), Sign(S) and Zero(Z) flag. Flags will be affected as per the rules of assembly language. C and V will be cleared after the logical operation. S and Z will be changed according to the output. Minimum possible ICs are used to design.

CS2	CS1	CS0	Function	Mathematical Representation
0	0	0	Add	A+B
0	0	1	Add with Carry	A+B+1
0	1	X	AND	$A \wedge B$
1	0	0	XOR	$A \oplus B$
1	0	1	Transfer A	A
1	1	X	Decrement A	A-1

Truth Table and Required K-Maps:

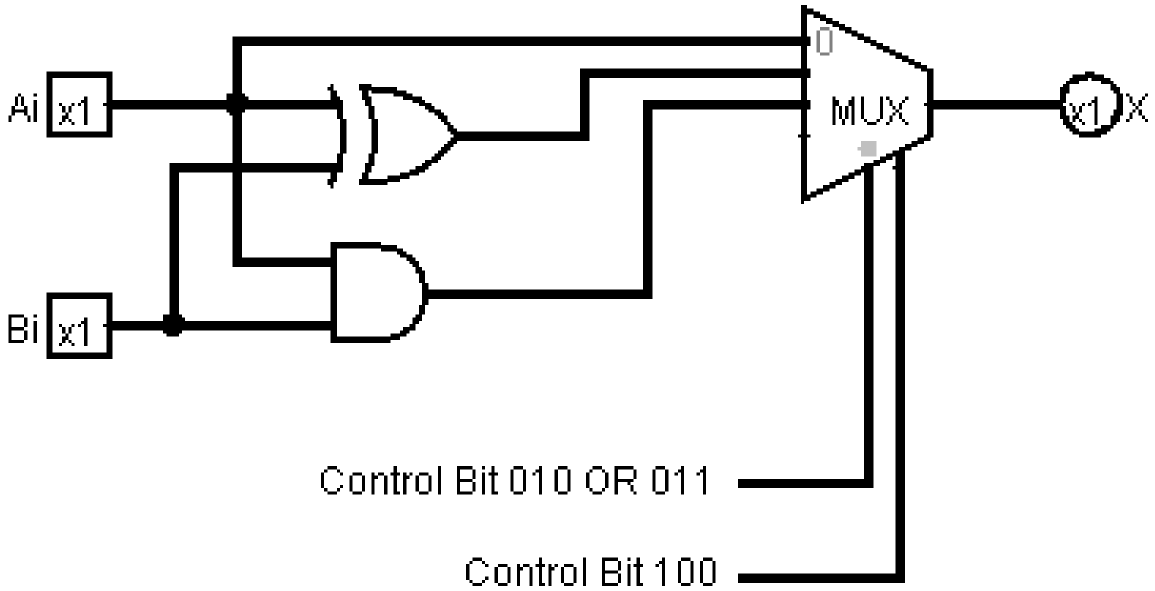
CS2	CS1	CS0	Function	X	Y	Z
0	0	0	A+B	A_i	B_i	0
0	0	1	A+B+1	A_i	B_i	1
0	1	0	A ∧ B	$A_i \wedge B_i$	0	0
0	1	1	A ∧ B	$A_i \wedge B_i$	0	0
1	0	0	A ⊕ B	$A_i \oplus B_i$	0	0
1	0	1	A	A_i	0	0
1	1	0	A-1	A_i	1	0
1	1	1	A-1	A_i	1	0

No K-maps Required.

Z=CS2’CS1’CS0

Y=CS2’CS1’B_i+CS2CS1

For X: Here, no logical equation is used. MUX is used to control the input.



Design Steps:

X, Y, Z are the input of the adder. We are using decoder for the control bit configuration. For the control 000 we have to add two bits A_i and B_i. So, the input of the adder is A_i, B_i, 0. Then we have to add with carry for the control 001. For this operation the input is A_i, B_i, 1. For transfer, the control is set to 101 and the input of the adder is A_i,0,0. Now, comes to the decrement. This operation will be done when control is set to 11X. X means don’t care. That means we have to perform decrement for 110 as well as 111. Decrement is done by subtracting 1. For four bits input the 2’s complement of 1 is 1111. That is why when decrementing input of the adder is A_i,1,0.

Now comes to the logic part. We can use different setup for arithmetic and logic part but it is easier for us to perform this operation with the same adder. So, earlier we have done the operation and we give the output of logic part as the input of adder. That means we have to perform the AND and XOR and transfer the output. That is why the input of the adder is $A_i \wedge B_i$,0,0 for AND operation and $A_i \oplus B_i$,0,0 for XOR operation. Here, we have to perform XOR for control 100 and perform AND for 01X. It implies like previous that we have to perform AND for control 010 and 011.

Now, we have derived the equation of Y (input B of adder) and Z (Cin of Adder) from truth table using simple Boolean logic and we got this equation. So, we got this equation,

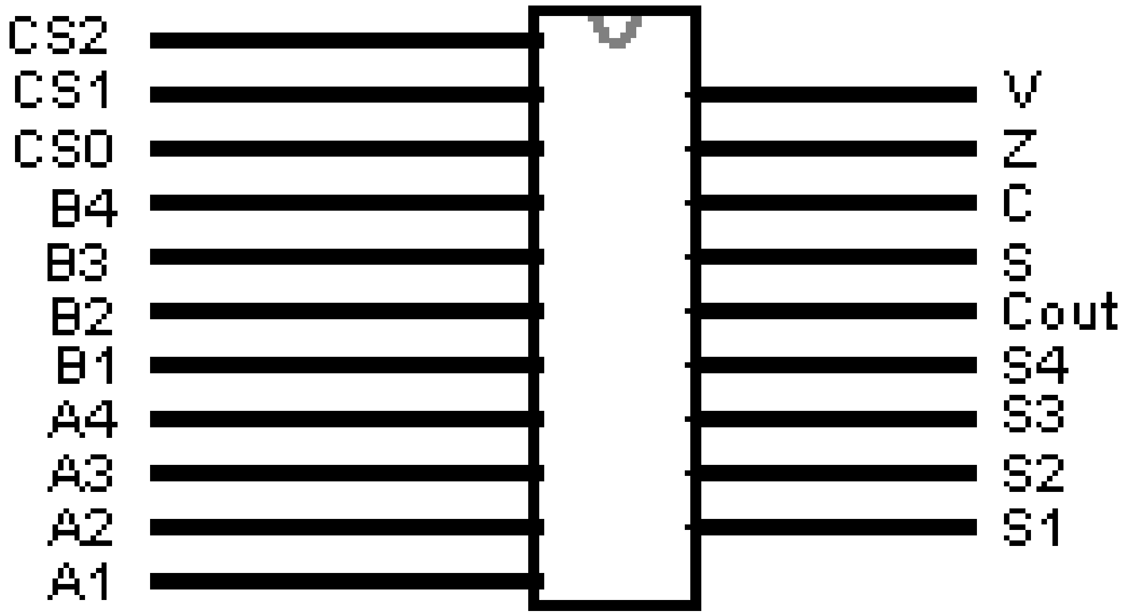
Z=CS2’CS1’CS0

Y=CS2’CS1’B_i+CS2CS1

Now for X, we have to look in the truth table. Here three input is possible A_i, $A_i \wedge B_i$, $A_i \oplus B_i$. So, for the control 100 we will get output 1 from decoder and all the other input is 0. So, for the selection 01, the output of the MUX is $A_i \oplus B_i$. Again, for the control 010 or 011, we get 1 from decoder and $A_i \wedge B_i$ will be gone to output of the MUX for the selection bit 10. And by default, when we configure other control bit we will get 00 as selection and A_i will go to the output of the MUX. Thus, X is controlled using MUX.

All the design is for one bit ALU. Now, now we have to perform this for four bit ALU.

Block Diagram:



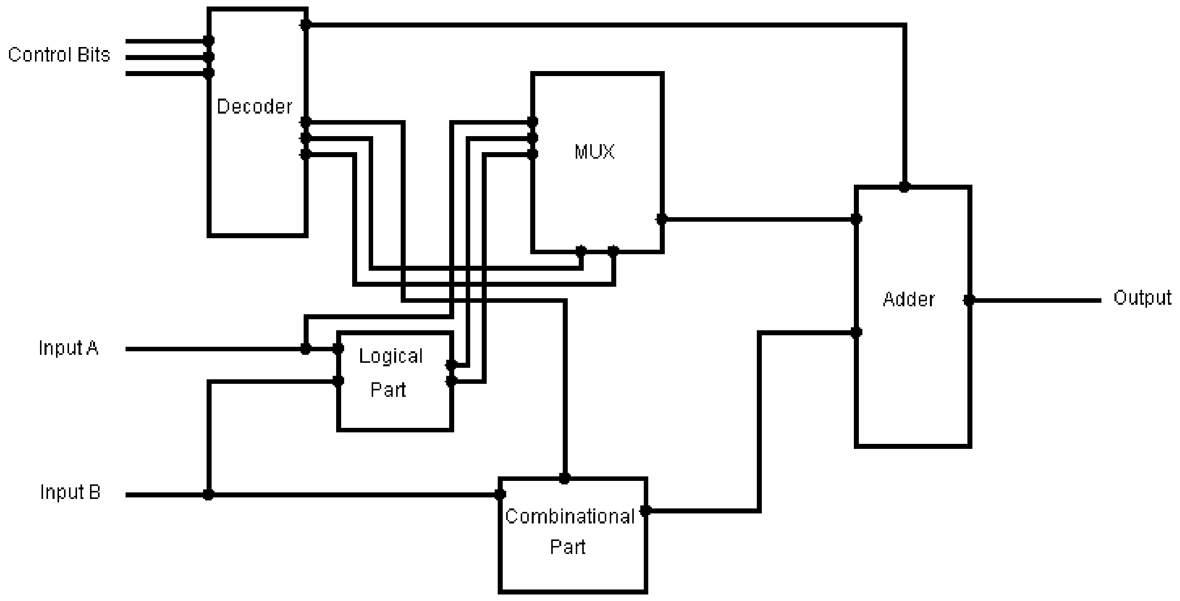
Here,

CS2, CS1, CS0 = Selection Bits

A4, A3, A2, A1 = 4 bit Input A and B4, B3, B2, B1 = 4 bit Input B

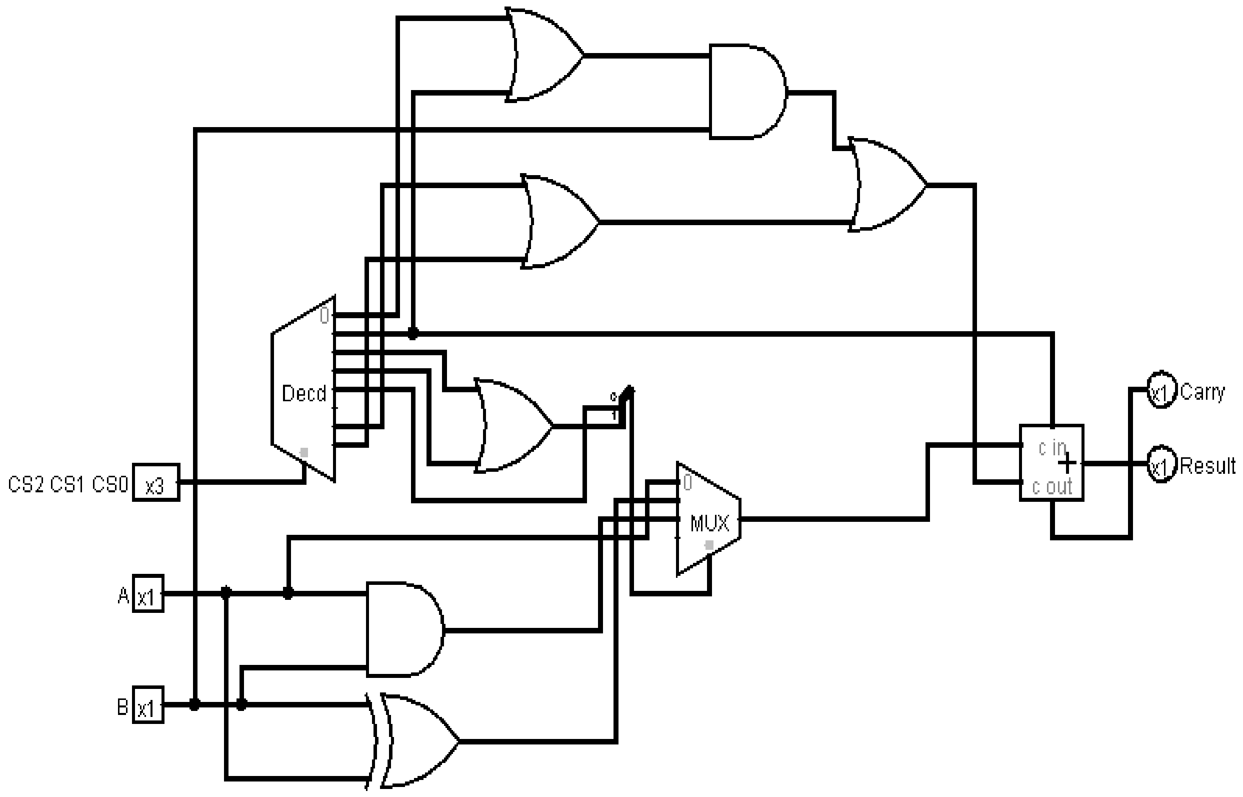
S4, S3, S2, S1 = 4 bit Output S and Cout = Carry output

C = Carry Flag, Z = Zero Flag, V = Overflow Flag, S = Sign Flag

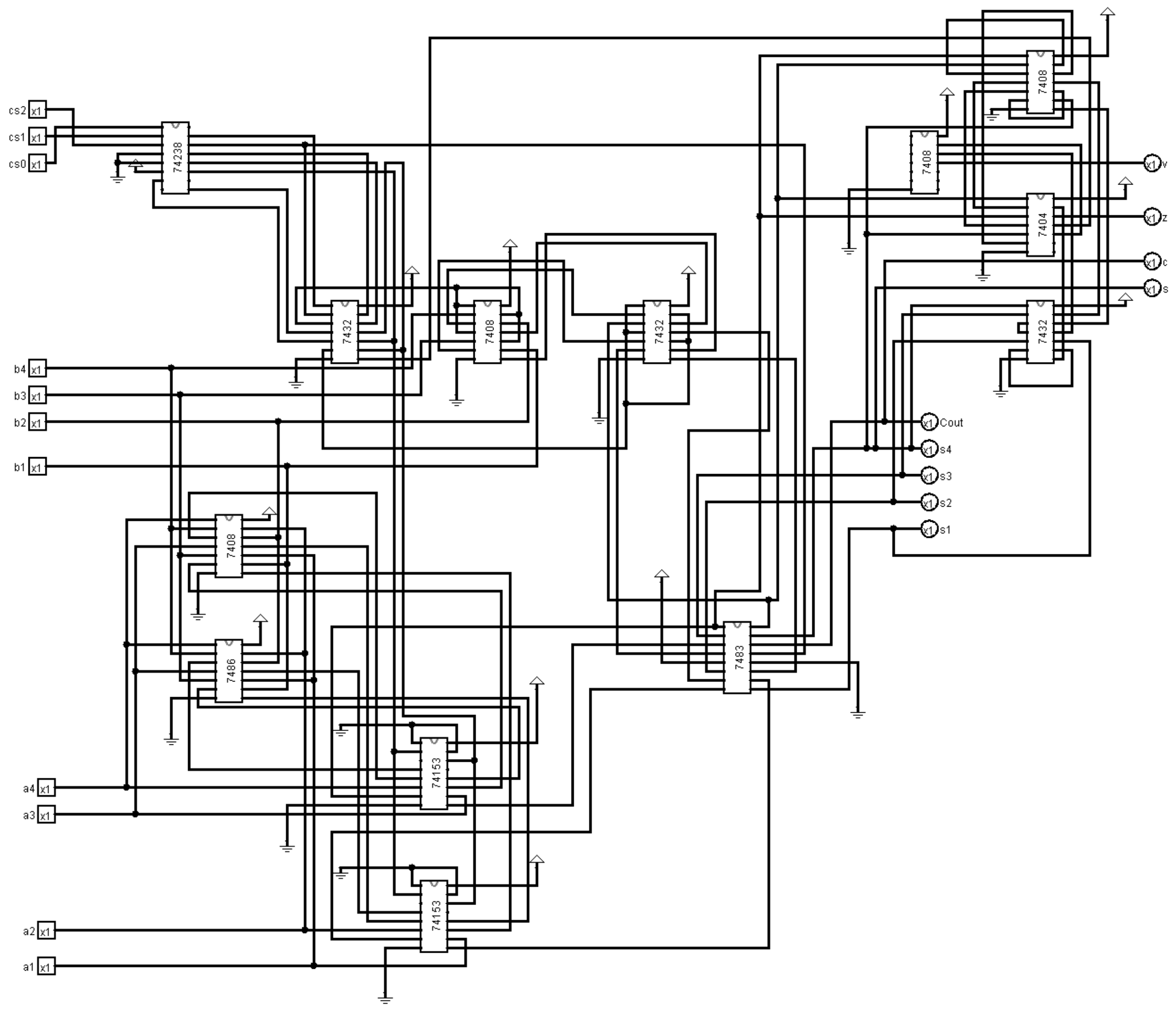


Complete Circuit Diagram:

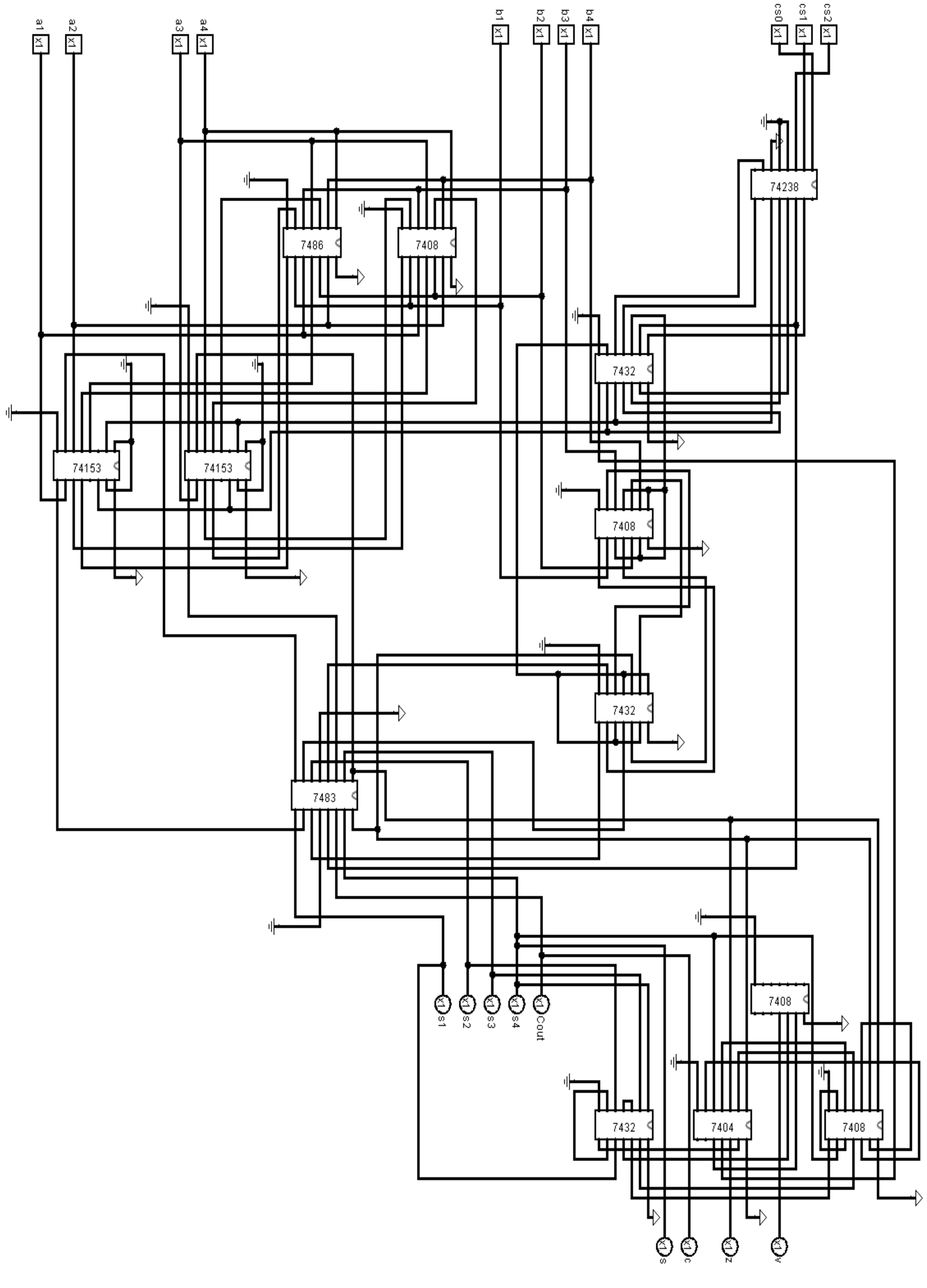
1 bit ALU:



4 bit ALU:


cs2
cs1
cs0
74238
b4
b3
b2
b1
7432
7408
7432
7408
7486
a4
a3
a2
a1
74153
74153
7483
7408
7408
7404
7432
Cout
s4
s3
s2
s1
v
z
c
s
x1

4 bit ALU: (Landscape View)



IC Count:

IC number	IC Name	Count
7404	Logical NOT	1
7408	Logical AND	4
7432	Logical OR	3
7483	4 bit Binary Adder	1
7486	Logical X-OR	1
74153	4x1 MUX	2
74238	Decoder	1

The Simulator Used:

Logisim V2.7.1

Discussion:

There are two phases in our ALU implementation. First part is software implementation. Logisim is used for this part. We used 7400 library file to use IC (Integrated Circuit) for our design. We use decoder to control which operation is to done. So, the input of the decoder is selection bits. Then we design the input of the adder. There are three types of input in adder; two numbers and carry. We design one number and carry using logical gates and use mux for another number. So, after getting output, we have to set the flags. Carry Flag is the “Cout” of the adder. When all the bits are zero, the Zero flag will be given 1. Besides, the MSB is the Sign Flag. Last, if we add two numbers of same sign bit and get the number of opposite sign bit, the Overflow Flag will be given 1. If we add two numbers of different sign, there will be no overflow.

Now, it comes the hardware implementation. It is the hardest part of this assignment. We faced some major problems while implementing it. Some LEDs and ICs were faulty. So, we need a huge time to debug it. The wires become messy while connecting and sometimes we had to connect two long distance and that was one of the most difficult parts. So, we faced a vast of problems while implement the hardware part.

In the end, we successfully completed the implementation. Now, this ALU can perform all the operation. Suppose, if we set CS2=0, CS1=0, CS0=1, then it will perform add with carry. Suppose, A is 0011 and B is 0001, the result will be 0101. If we set CS2=0, CS1=1, CS0= (no matter whether it is 0 or 1), it will perform logical AND. Suppose A is 1011 and B is 1110 then the result is 1010. Thus, all the operation will be performed.

According to the operation, the flags will be set or reset. Suppose, the result is 0000 then Z=1. If A=1011 and B=1011 and we perform add, then the result is 0110 with carry 1. In this case, C=1 and V=1. As we add two numbers of same sign and we got a number with different sign, so the V will be 1. If the MSB of the result is 1, then S=1. That is how the flag will work.