CSE306 LAB REPORT

OFFLINE 1

GROUP - 03

4-bit ALU Simulation

Group Members
1705093
1705098
1705103
1705110
1705119

March 30, 2021

1 Introduction

An arithmetic logical unit (ALU) is a digital circuit used to perform arithmetic and logic operations. It represents the fundamental building block of the central processing unit (CPU) of a computer.

The purpose of the ALU is to perform mathematical operations such as addition, subtraction, multiplication and division. Additionally, the ALU processes basic logical operations like AND/OR calculations.

Unless otherwise stated, we assume that the inputs A and B are signed, two's complement number when they are presented to the input of the ALU. The operations performed by an ALU are controlled by a set of operation-select inputs.Logical operations take place on the bits that comprise a value (known as bitwise operations), while arithmetic operations treat inputs and outputs as two's complement integers. Errors must be detected by the ALU.If an addition results in overflow or a multiplication results in a value that cannot be shortened to 4 bits, the Overflow flag must be enabled.

2 Problem Specification

We are going to design an ALU for the following specifications

Table 1: Design Spec for Group-03

Cs2	Cs1	Cs0/Cin	Function
0	0	0	Subtract with borrow
0	0	1	Subtract
0	1	0	Transfer A
0	1	1	Increment A
1	0	x	OR
1	1	x	AND

3 Truth table and K-Maps

Table 2: Truth Table for X_i, Y_i, Z_i

Cs2	Cs1	Cs0	X_i	Y_i	Z_i
0	0	0	A	B′	0
0	0	1	A	B′	1
0	1	0	A	0	0
0	1	1	A	0	1
1	0	x	$A \vee B$	0	0
1	1	x	AB	0	0

3.1 K-Maps for X_i

Cs2Cs1 \ AB	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	0
10	0	1	1	1

So the final equation for X_i ,

$$
\begin{aligned}
&X_i \\
&= AB + Cs_2Cs_1'B + Cs_1'A + Cs_2'A \\
&= A(B + Cs_2' + Cs_1') + BCs_2Cs_1'
\end{aligned}
$$

3.2 K-Map for Y_i

B \ Cs2/Cs1	00	01	11	10
0	1	0	0	0
1	0	0	0	0

$$Y = Cs_2'Cs_1'B'$$

3.3 K-Map for Z_i

Cs2 \ Cs1/Cs0	00	01	11	10
0	0	1	1	0
1	0	0	0	0

$$Z = Cs_2'Cs_0$$

4 Block Diagram

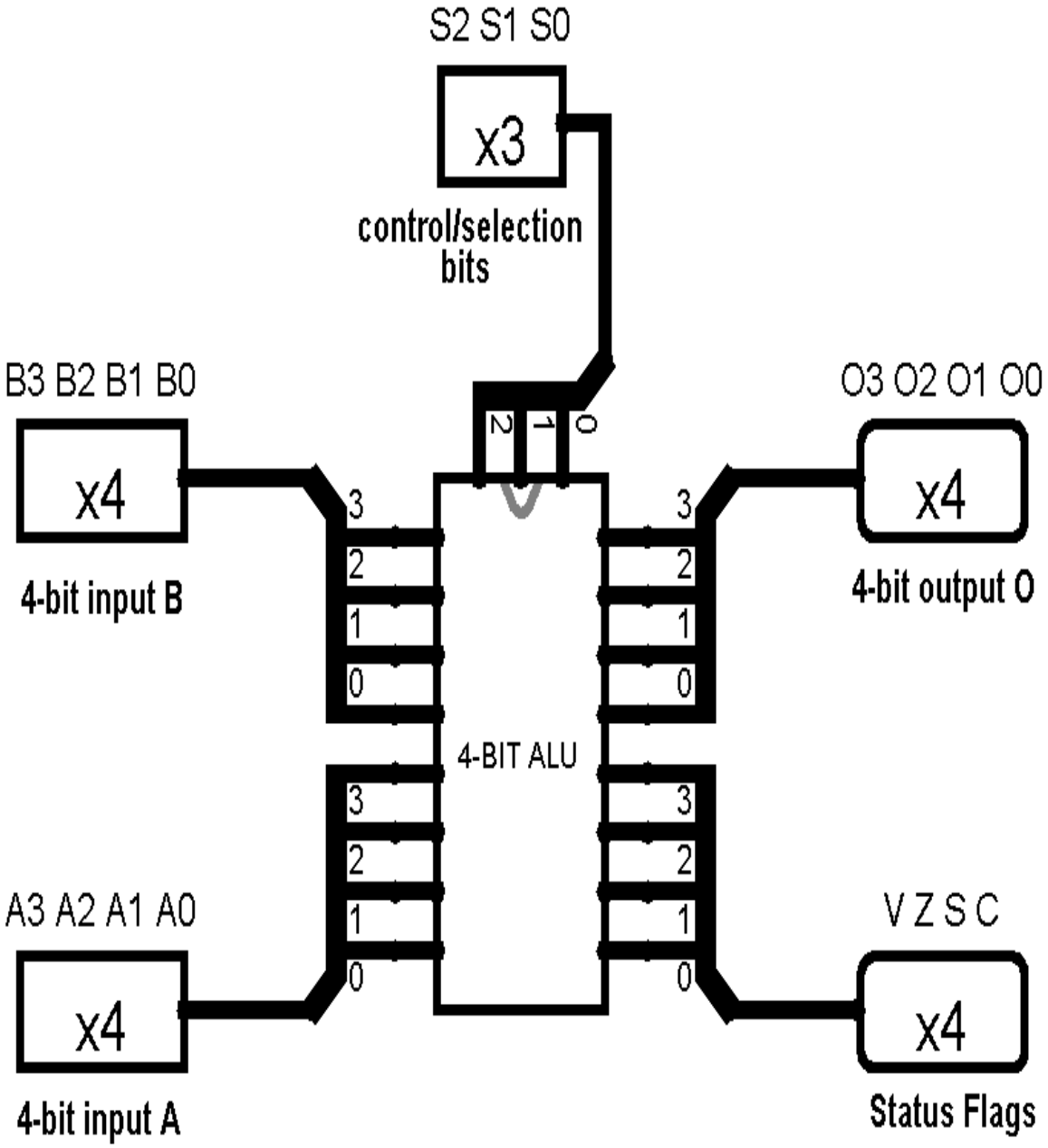

Figure 1: Block Diagram of a 4 bit ALU

5 Circuit Diagram

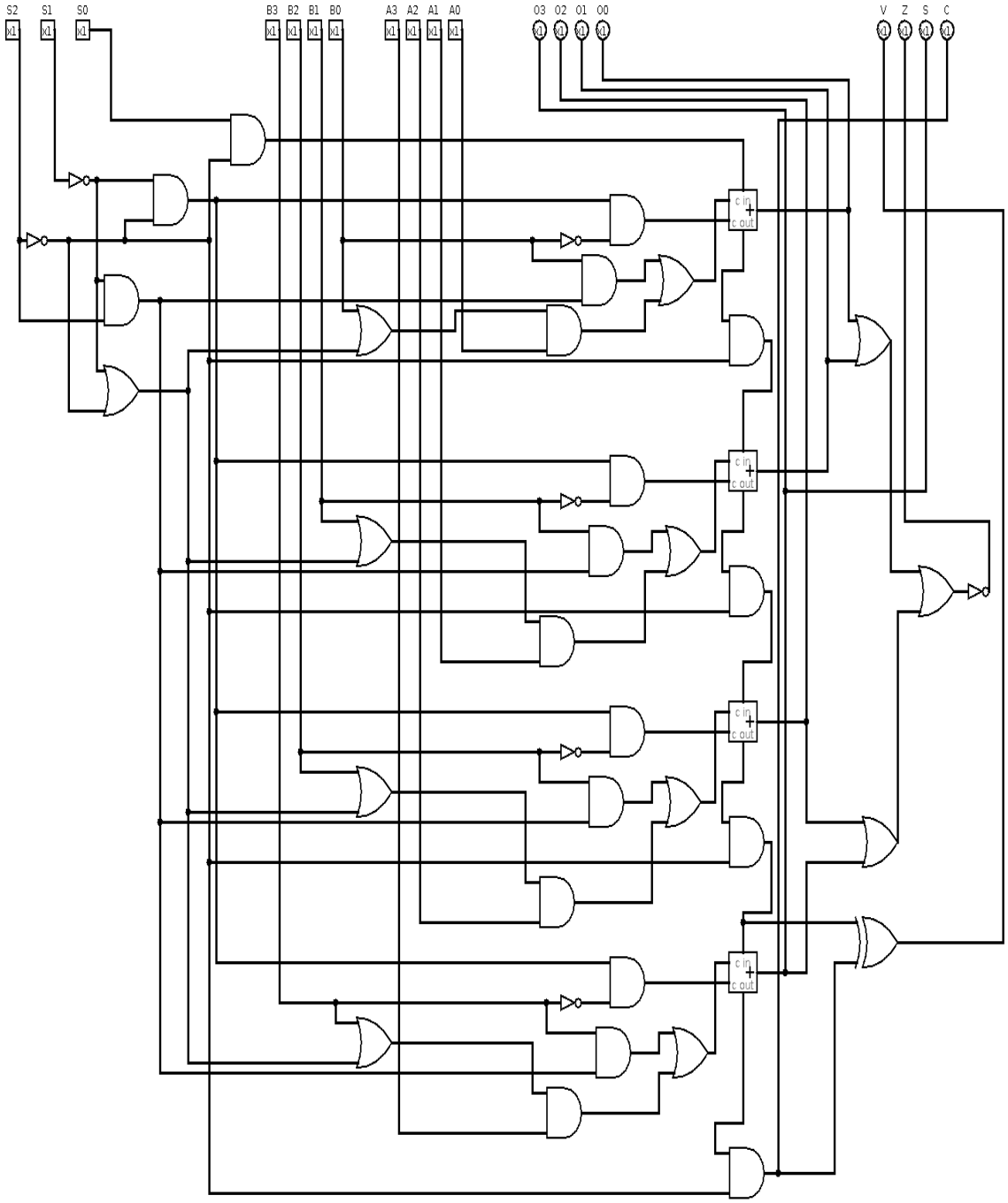

Figure 2: Circuit Diagram of A 4 Bit ALU

6 ICs used with count as a chart

Gate	Base Count	Count/Adder	Total	IC	IC Count
AND	3	4	19	7408	5
OR	4	2	12	7432	3
NOT	3	1	7	7404	2
XOR	1	-	1	7486	1

Table 3: Gates and IC chart

7 Simulation Platform

Logisim-2.7.10

8 Discussion

We tried to make our implementation as efficient as possible. To achieve this at first we applied gate level minimization using K-Maps. From the resulting SOPs of the K-Maps, we applied some basic Boolean algebra to minimize the number of gates further.

When implementing our design, we identified parts of the circuitry that were common for each of the 4 bits. Then we reused the common parts to avoid inefficiency due to redundancy.

Even though we designed the circuit using basic gates, we tried to minimize the number of ICs that would be needed if this circuit were to be implemented in lab using the 7400 series ICs. We did this by keeping the following idea in mind while designing and implementing : "When a new operation is needed try to use empty slots on existing ICs instead of using a gate that would require a new IC"

While implementing we tried to place the ICs in a systematic manner so that the logic being implemented

by the gates is readily obvious to anyone looking at the circuit diagram.

Finally, we created a block diagram version of our ALU design and tested our design with various inputs. This level of abstraction made it easier to verify inputs/outputs without having to think about the underlying design each time.