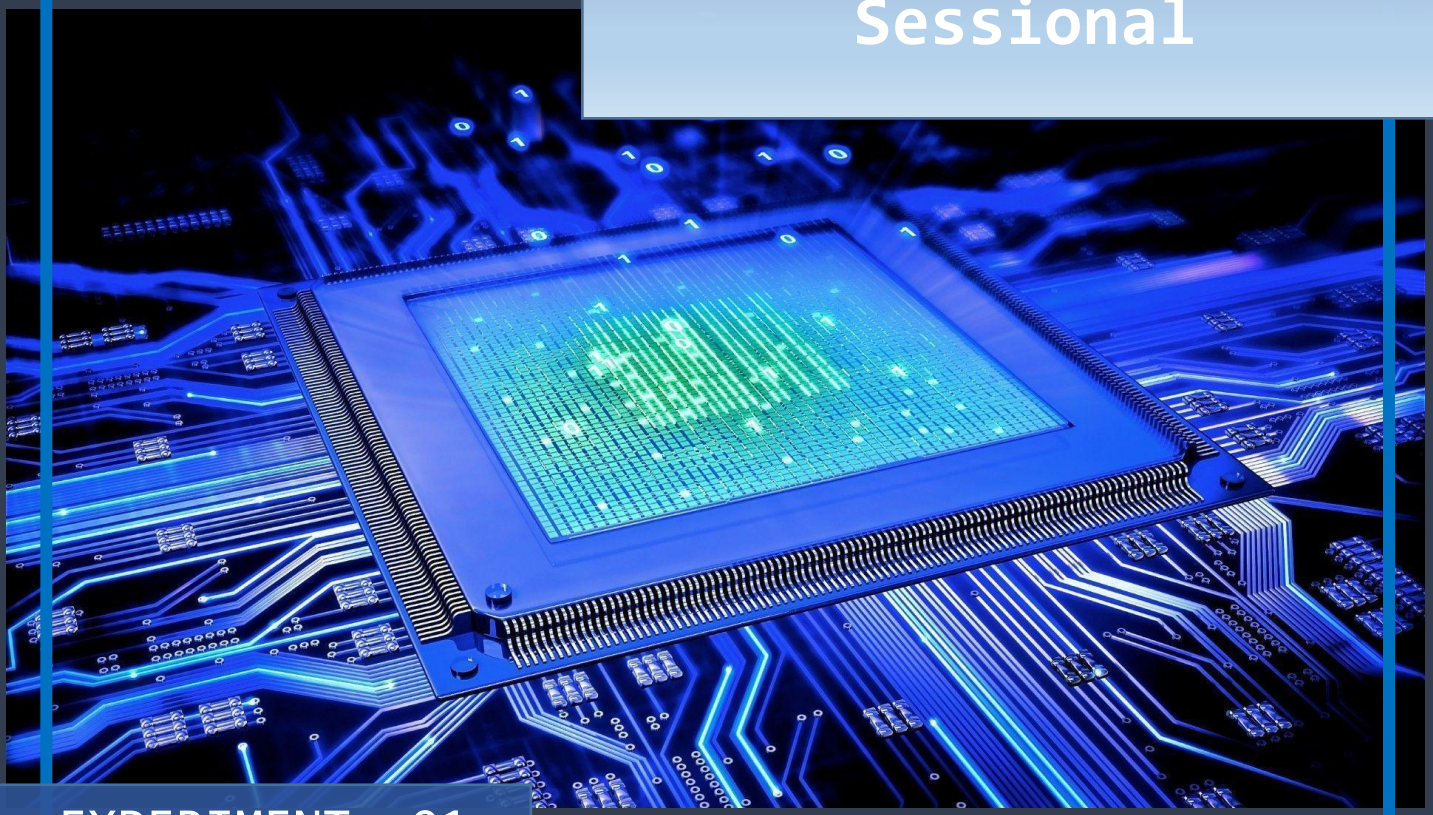
CSE 306

Computer Architecture Sessional



EXPERIMENT: 01

NAME OF EXPERIMENT:

4-bit ALU simulation

GROUP NO: 04

Section: A2

Department: CSE

Group Members:
1705041
1705047
1705049
1705050
1705052

Date of Submission: 27-03-2021

Introduction:

ALU or Arithmetic Logical Unit is a digital circuit to do arithmetic operations like addition, subtraction, division, multiplication and logical operations like AND, OR, XOR, NAND, NOR etc.

The goal of this experiment is to understand the basics of building complex circuit from simple AND, OR, NOT and XOR logical gates and build an Arithmetic Logic Unit (ALU) from scratch, using these simple logic gates and other components. The ALU will take in two 4-bit values, and 3 control lines. Depending on the value of the control lines, the output will be decrement A, the addition, transfer A, add with carry, complement A, or bitwise XOR of the inputs.

There are four status outputs as well as the main result: carry, sign, overflow and zero.

Problem Specification with assigned instructions:

Specification: An ALU is to be designed that takes two 4 bit inputs A (A_3, A_2, A_1, A_0) and B (B_3, B_2, B_1, B_0). ALU should perform different operations according to the input state of selection bits.

Instructions:

The flags mentioned below should be operational:

1. Carry
2. Sign
3. Overflow
4. Zero

Additional Directives:

• Flags need to be affected as per the rules of Assembly Language.

• Any SSI (AND, OR, NOT, XOR etc.) and MSI (MUX, Decoder, Adder etc.) chip can be used.

• Emphasis is to be given on efficiency of design and minimization of ICs used.

Table:

cin			Functions to be Implemented
cs2	cs1	cs0	
0	0	0	Decrement A
0	0	1	Add
1	0	0	Transfer A
1	0	1	Add with carry
x	1	0	Complement A
x	1	1	XOR

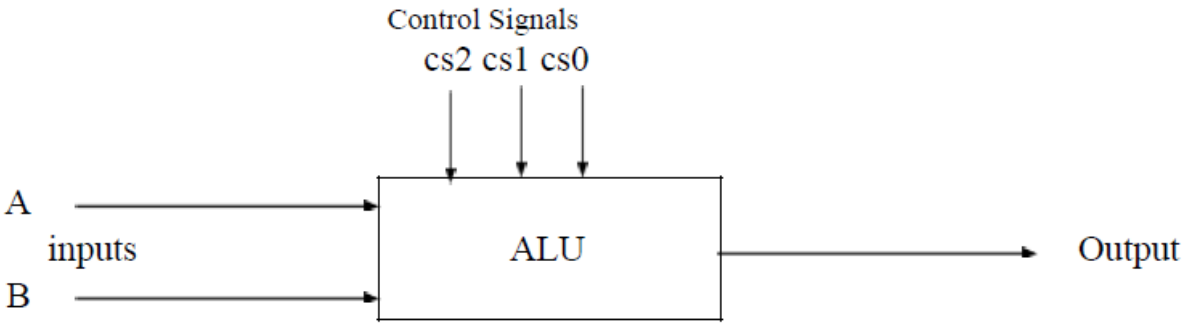


Figure 1: Intended Circuit Diagram

Truth Table:

CS2	CS1	CS0	Operation	F	Type of OP	X	Y	Cin
0	0	0	Decrement A	A-1	Arithmetic	A	1	0
0	0	1	Add	A+B	Arithmetic	A	B	0
1	0	0	Transfer A	A	Arithmetic	A	1	1
1	0	1	Add with Carry	A+B+1	Arithmetic	A	B	1
x	1	0	Complement A	A'	Logical	A	1	x
x	1	1	XOR	A⊕B	Logical	A	B	x

K-Maps:

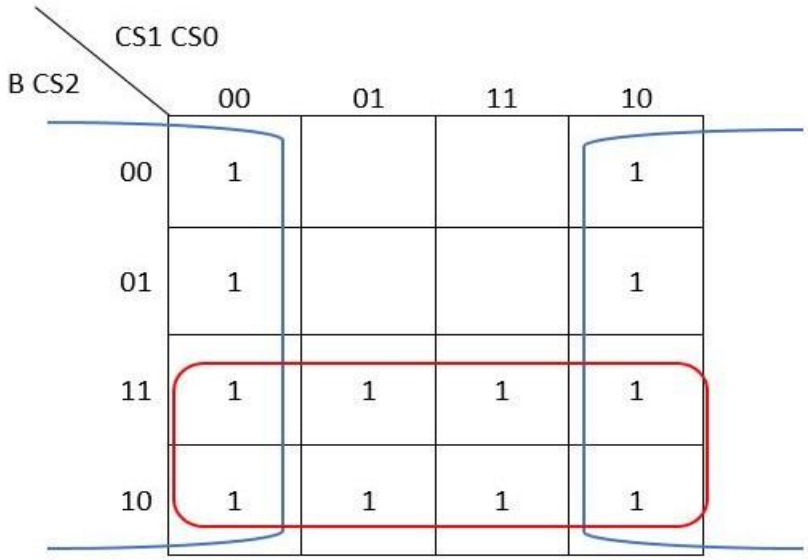


Figure 2: K-Map for finding Y

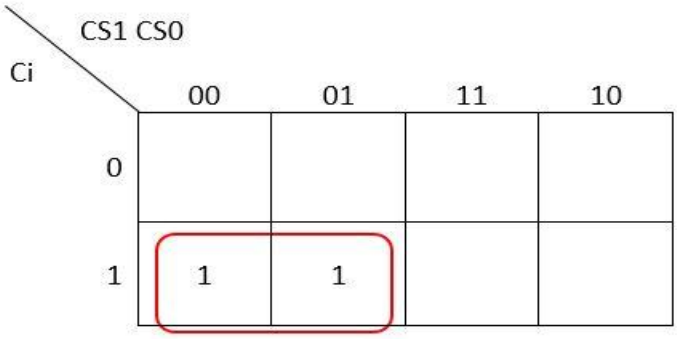


Figure 3: K -Map for Z_i

$X_i = A_i$

$Y_i = CS0` + B_i$

$Z_i = CS1`.C_i$

Circuit Diagram:

Complete Circuit Diagram:

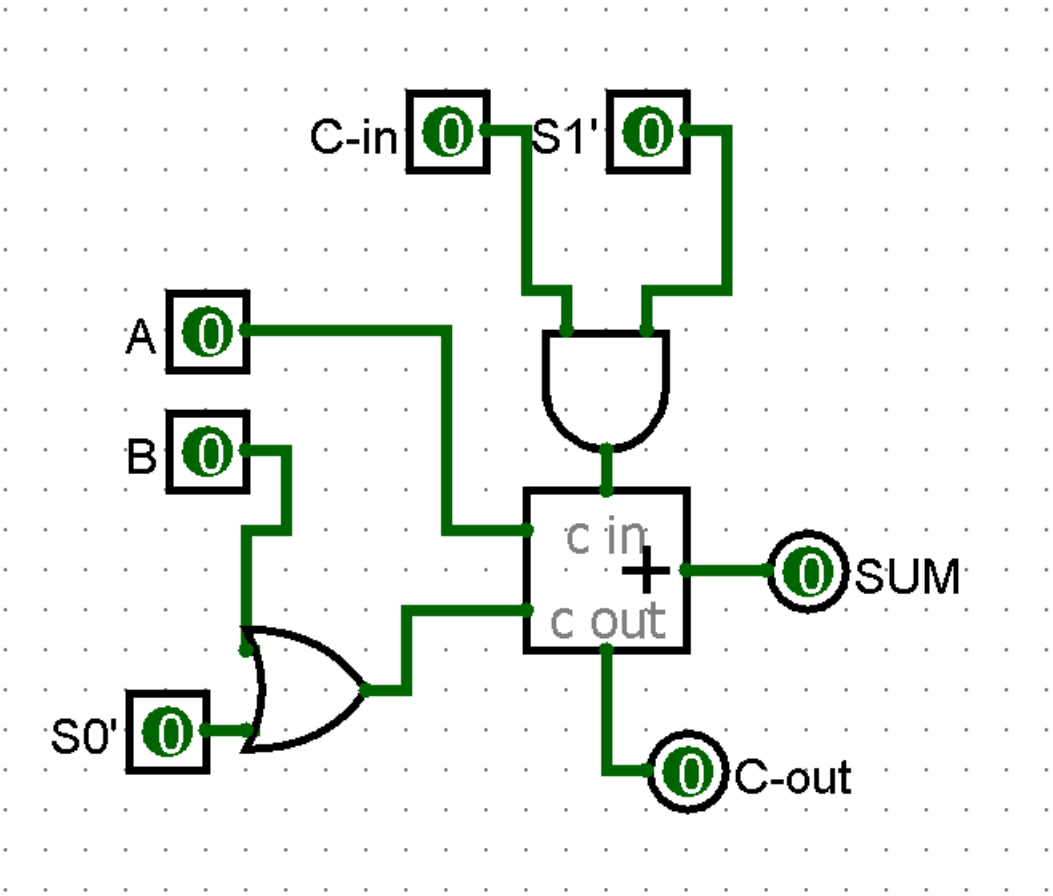


Figure 2: 1- bit ALU Design

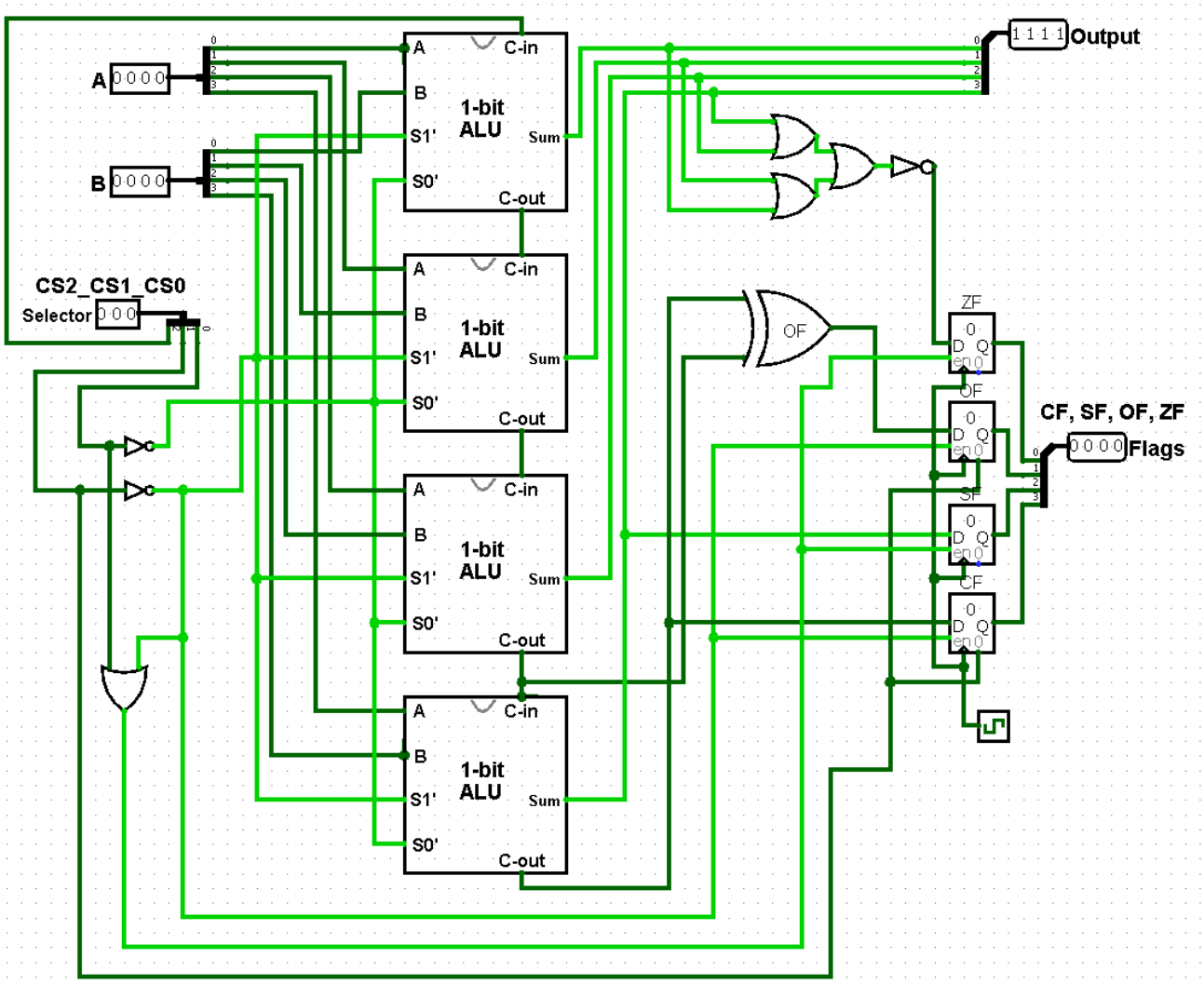


Figure 3: Complete Circuit Diagram of a 4-bit ALU

Block Diagram:

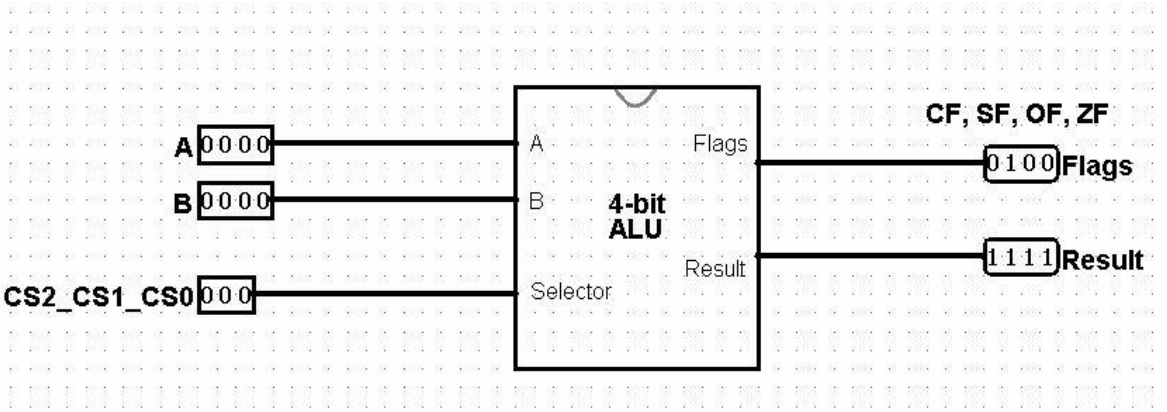


Figure 4: Block Diagram of 4-bit ALU Circuit

IC Count:

Used IC	Operation	Gates Used	Gates Per IC	IC Count
IC 74LS08	AND	4	4	1
IC 74LS32	OR	8	4	2
IC 74LS04	NOT	3	6	1
IC 74LS86	XOR	1	4	1
IC 74LS83	1-bit Full Adder	4 adders	4	1
IC 74LS74	Memory(D flip-flop)	4	2	2

Total IC used: 9

Simulator Used: Logisim

Version Number: 2.7.1

Discussion:

In this assignment, an Arithmetic Logic Unit was implemented using a virtual simulator. The ALU has two different modes of operation which is determined by a mode selection bit, which in our case was CS_1. The mode selection bit determined whether the ALU would perform an arithmetic operation or logical operation on the given inputs. For our group, we implemented designated arithmetic functional designs such as decrement A, addition of two Inputs A and B, transfer A, add A and B with Carry(which in our case was CS_2). Alongside we also implemented logical operations like complement A and XOR when mode selection bit was enabled. We added flags to our design as per specification and ensured that those flags were affected as per the rules of Assembly Language. During Logical operations, the carry bit CS_2 was given "don't care"(x) condition, because logical operations should perform only on two inputs.

For implementing and simulating these functional designs, we used **Logisim**. It is a useful software to make circuit and block diagram. During simulation it is necessary for the user to do 2 things- to ensure ticks enabled is checked and simulation enabled is checked.

For circuit designing, care was given so that minimum number of IC's are used. We used K-Maps and Boolean statement simplification for this purpose. Also we tried to make the design as clean and compact as possible for easier understanding.

From this assignment, the details of ALU implementation can be understood precisely and thoroughly.