

Bangladesh University of Engineering and Technology  
CSE 209  
Computer Architecture Sessional



Assignment-1  
4-bit ALU Design

Section A1  
Group 01

Group Members:

1. Khalid Hasan Tuhin - 2105002
2. K.M. Mehemud Azad - 2105014
3. Arnob Biswas - 2105015
4. Zaki Rehnoom Unmona - 2105016
5. Gourave Roy - 2105017

# 1 Introduction

An arithmetic logic unit (ALU) is a multi-operation, combinational-logic digital function. It can perform a set of basic arithmetic operations and a set of logic operations. The ALU has a number of selection lines to select a particular operation in the unit. The selection lines are decoded within the ALU so that  $k$  selection variables can specify up to  $2^k$  distinct operations.

Figure 1 shows the block diagram of a 4-bit ALU. The four data inputs from A are combined with the four inputs from B to generate an operation at the F outputs. The mode-select input  $s_2$  distinguishes between arithmetic and logic operations. The two function-select inputs  $s_1$  and  $s_0$  specify the particular arithmetic or logic operation to be generated. With three selection variables, it is possible to specify four arithmetic operations (with  $s_2$  in one state) and four logic operations (with  $s_2$  in the other state). The input and output carries have meaning only during an arithmetic operation. The input carry in the least significant position of an ALU is quite often used as a fourth selection variable that can double the number of arithmetic operations. In this way, it is possible to generate four more operations, for a total of eight arithmetic operations.

The input carry in the least significant position of an ALU is quite often used as a fourth selection variable that can double the number of arithmetic operations. In this way, it is possible to generate four more operations, for a total of eight arithmetic operations.

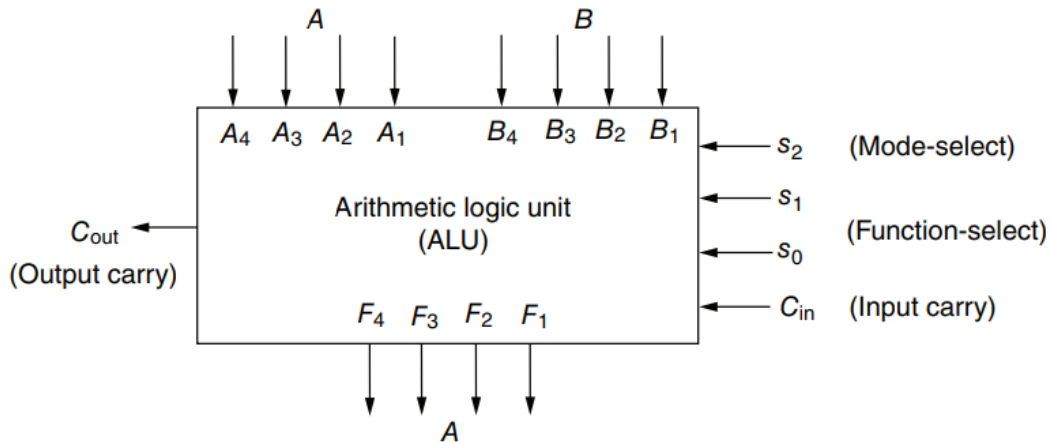


Figure 1: Block Diagram of a 4-bit ALU

The design of a typical ALU will be carried out in three stages. First, the design of the arithmetic section will be undertaken. Second, the design of the logic section will be considered. Finally, the arithmetic section will be modified so that it can perform both arithmetic and logic operations.

There are also 4 status outputs (flags) in ALU. They are denoted by C(Carry Flag), Z(Zero Flag), V(Overflow Flag), S(Sign Flag). Their representations carry out the following meanings:

- C(Carry Flag): This flag is set when there is a carry out of the most significant bit of the result.
- Z(Zero Flag): This flag is set when the result of the operation is zero.
- V(Overflow Flag): This flag is set when the result of the operation is too large to be represented in the given number of bits.
- S(Sign Flag): This flag is set when the result of the operation is negative.

## 2 Problem Specification with assigned instructions

Design a 4-bit ALU with three selection bits CS2, CS1, CS0 that can perform the following operations:

| CS2 | CS1 | CS0 | Functions       |
|-----|-----|-----|-----------------|
| 0   | 0   | 0   | Add             |
| 0   | 0   | 1   | AND             |
| 0   | 1   | X   | Sub with borrow |
| 1   | 0   | 0   | Complement A    |
| 1   | 0   | 1   | OR              |
| 1   | 1   | X   | NEG A           |

Table 1: Control Signals and Functions of the 4-bit ALU

Here, CS2, CS1, and CS0 are the control signals. X means that the value of the signal is not important. The ALU should have 4 status outputs (flags) C, Z, V, S.

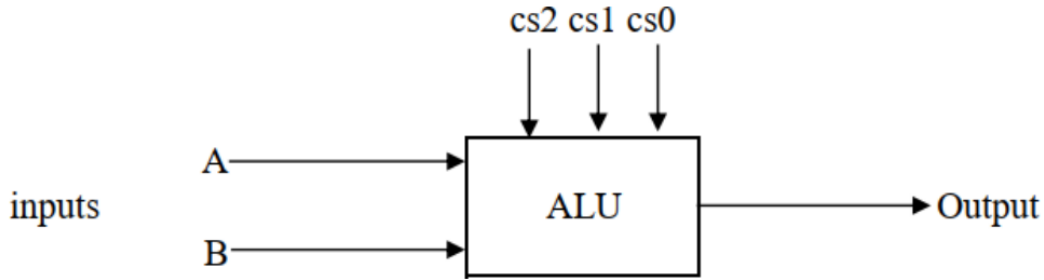


Figure 2: Block Diagram of a 4-bit ALU

## 3 Detailed design steps with k-maps

### 3.1 Design Steps

### 3.2 K-maps

We will be following table 3 and 4 to construct the K-maps for selection bits, enable bit of multiplexers and  $C_{in}$ .

The IC of parallel adder takes  $X_i$  and  $Y_i$  and  $C_{in}$  as input. We need  $X_i$  as  $A_i$  or its complement or its logical changes with B. These values are received as  $X_i$  (output of the 4 to 1 multiplexer) and the selection bits for the multiplexer are  $S_{11}$  and  $S_{10}$ . For  $Y_i$  we want  $B_i$  or its complement as the output of 2 to 1 multiplexer. The selection and enable bit of this (2 to 1 MUX) multiplexer are  $S_2$  and  $\overline{E_2}$  respectively. The k-map for the selection bits, enable bit of the multiplexer and carry input  $C_{in}$  are as follows:

### 3.2.1 K-map for $S_{11}$

| $c_{s2}c_{s1} \backslash c_{s0}$ | 0 | 1 |
|----------------------------------|---|---|
| 00                               | 0 | 0 |
| 01                               | 0 | 0 |
| 11                               | 1 | 1 |
| 10                               | 1 | 1 |

We can easily express  $S_{11}$  as following:

$$S_{11} = C_{S_2}$$

### 3.2.2 K-map for $S_{10}$

| $c_{s2}c_{s1} \backslash c_{s0}$ | 0 | 1 |
|----------------------------------|---|---|
| 00                               | 0 | 1 |
| 01                               | 0 | 0 |
| 11                               | 0 | 0 |
| 10                               | 0 | 1 |

There are two minterms:

$$S_{10} = \overline{C_{S_1}}C_{S_0}$$

### 3.2.3 K-map for $S_{10}$

$S_2$  is the selection bit for the multiplexer that selects B and  $\bar{B}$  as input of a 2 to 1 MUX .

| $\begin{matrix} cs0 \\ cs2cs1 \end{matrix}$ | 0 | 1 |
|---------------------------------------------|---|---|
| 00                                          | 0 | X |
| 01                                          | 1 | 1 |
| 11                                          | X | X |
| 10                                          | X | X |

The simplified form for  $S_2$  will be:

$$S_2 = C_{S_1}$$

### 3.2.4 K-map for $\overline{E}$

$\overline{E}$  is the enable bit for the 2 to 1 multiplexer that decides when the multiplexer will be functioning( will select one of its inputs) or not(in ground state).

| $\begin{matrix} cs0 \\ cs2cs1 \end{matrix}$ | 0 | 1 |
|---------------------------------------------|---|---|
| 00                                          | 0 | 1 |
| 01                                          | 0 | 0 |
| 11                                          | 1 | 1 |
| 10                                          | 1 | 1 |

$$\overline{E} = C_{S_2} + \overline{C_{S_1}}C_{S_0}$$

### 3.2.5 K-map for $C_{in}$

It is the input carry bit of the adder used inside arithmetic unit.

| $\begin{matrix} CS0 \\ CS2CS1 \end{matrix}$ | 0 | 1 |
|---------------------------------------------|---|---|
| 00                                          | 0 | 0 |
| 01                                          | 0 | 0 |
| 11                                          | 1 | 1 |
| 10                                          | 0 | 0 |

$$C_{in} = C_{S_2}C_{S_1}$$

## 4 Truth Table

| CS2 | CS1 | CS0 | Functions        | $X_i$       | $Y_i$       | $Z_i$ | Cin |
|-----|-----|-----|------------------|-------------|-------------|-------|-----|
| 0   | 0   | 0   | Add              | $A_i$       | $B_i$       | $C_i$ | 0   |
| 0   | 0   | 1   | AND              | $A_iB_i$    | 0           | $C_i$ | 0   |
| 0   | 1   | X   | Sub with borrow  | $A_i$       | $\bar{B}_i$ | $C_i$ | 0   |
| 1   | 0   | 0   | Complement $A_i$ | $\bar{A}_i$ | 0           | $C_i$ | 0   |
| 1   | 0   | 1   | OR               | $A_i+B_i$   | 0           | $C_i$ | 0   |
| 1   | 1   | X   | NEG A            | $\bar{A}_i$ | 0           | $C_i$ | 1   |

Table 2: Truth Table of  $X_i, Y_i, Z_i, Cin$

| CS2 | CS1 | CS0 | Functions        | $X_i$       | S11 | S10 |
|-----|-----|-----|------------------|-------------|-----|-----|
| 0   | 0   | 0   | Add              | $A_i$       | 0   | 0   |
| 0   | 0   | 1   | AND              | $A_iB_i$    | 0   | 1   |
| 0   | 1   | X   | Sub with borrow  | $A_i$       | 0   | 0   |
| 1   | 0   | 0   | Complement $A_i$ | $\bar{A}_i$ | 1   | 0   |
| 1   | 0   | 1   | OR               | $A_i+B_i$   | 1   | 1   |
| 1   | 1   | X   | NEG A            | $\bar{A}_i$ | 1   | 0   |

Table 3: Truth Table of MUX input for  $X_i$



## 6 Complete Circuit Diagram

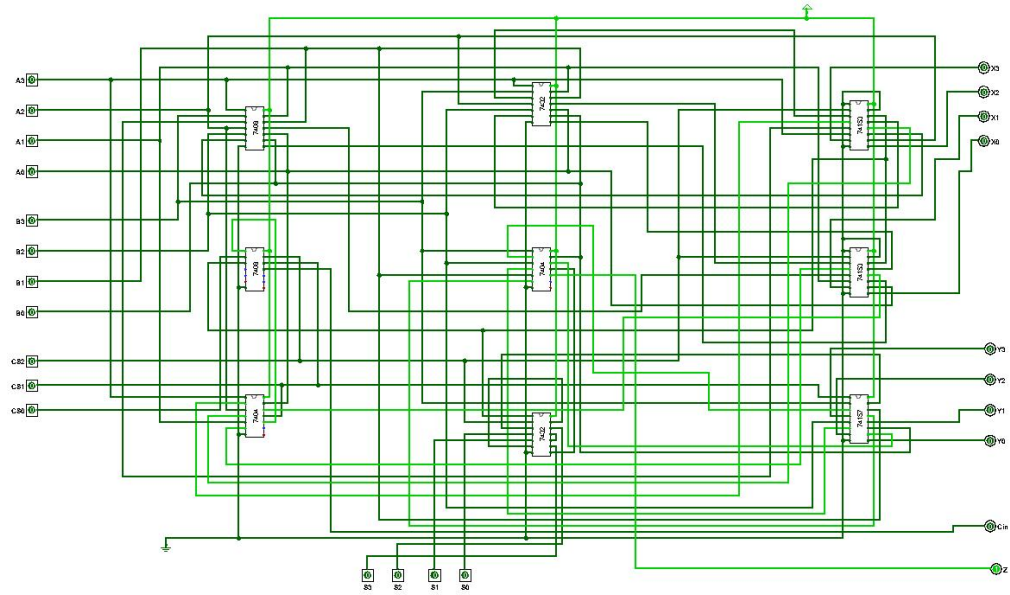


Figure 4: Input Processing Unit

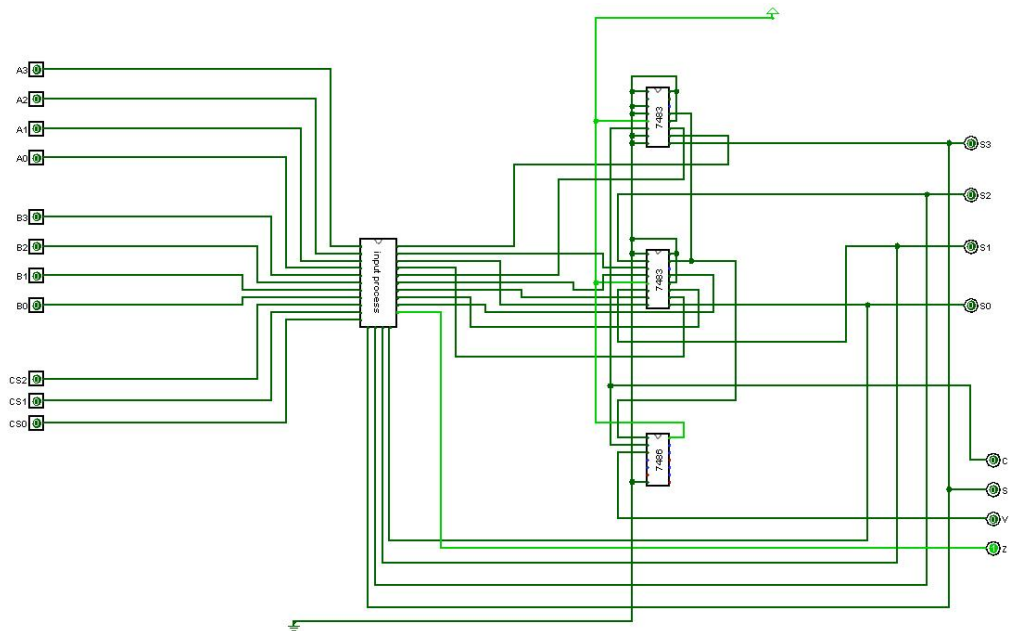


Figure 5: Arithmetic Logic Unit (ALU)



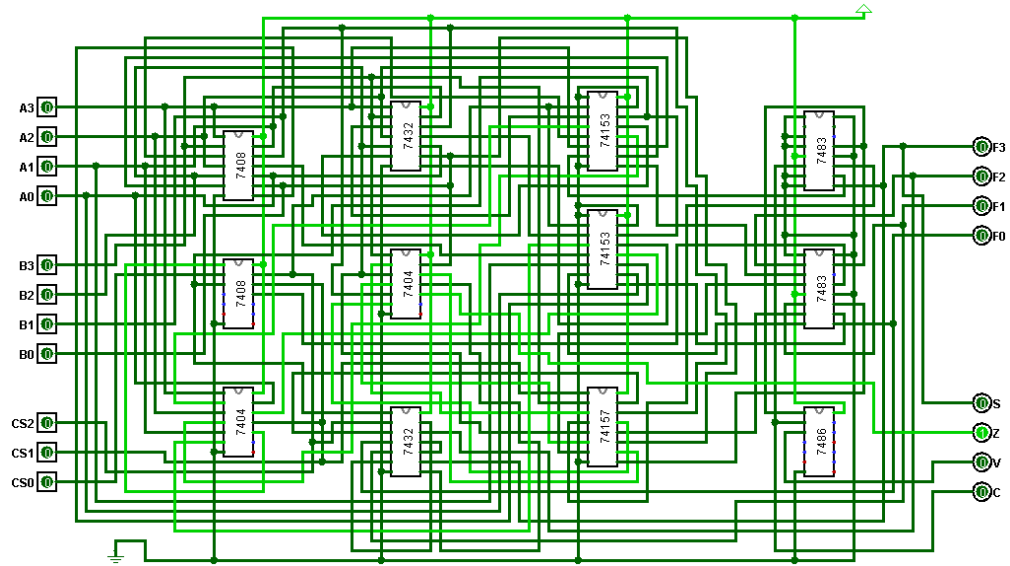


Figure 6: Complete Design

## 7 ICs used with count as a chart

| IC    | Count |
|-------|-------|
| 74153 | 2     |
| 74157 | 1     |
| 7408  | 2     |
| 7432  | 2     |
| 7404  | 2     |
| 7486  | 1     |
| 7483  | 2     |
| Total | 12    |

Table 5: ICs used with count

## 8 The simulator used along with the version number

Logisim - 2.7.1

## 9 Discussions

## 10 Contribution of Each Member