Assignment No- 01

Title : 4 bit ALU Simulation and Hardware Implementation

Course No: CSE 306

Course Name: Computer Architecture Sessional

Submitted By -

Group B2 - Group 01

1805091

1805093

1805105

1805106

1805111

Date : June, 2022

## 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 logical operations. The ALU has a number of selection lines to select a particular operation in the unit. The selection lines are decoded within, so that K selection variables can specify upto 2K distinct operations.

In our experiment, we have three selection variables (cs2, cs1, cs0) which enable us to perform 8 distinct operations. A combinational circuit is prepared to modify the data inputs (A and B) to produce the inputs (X, Y, Z) for the parallel adders to get the desired output bits (F).

In our demonstration, cs2 signifies the carry input and cs1 acts as the mode selector.

A 4bit status register is used to understand changes during arithmetic operations. They are C (carry), S (sign), V (overflow) and Z (zero). These 4 bits indicates -

(i) C : Set when output (unsigned) carry is 1, unset otherwise.

(ii) S : Set when the MSB of the output is set and unset otherwise.

(iii) V : Set when the output/result overflows. This is the signed overflow which we got by XOR-ing the two most significant carries.

(iv) Z : Set when the result is zero, unset otherwise.

## Problem Specifications with assigned instructions

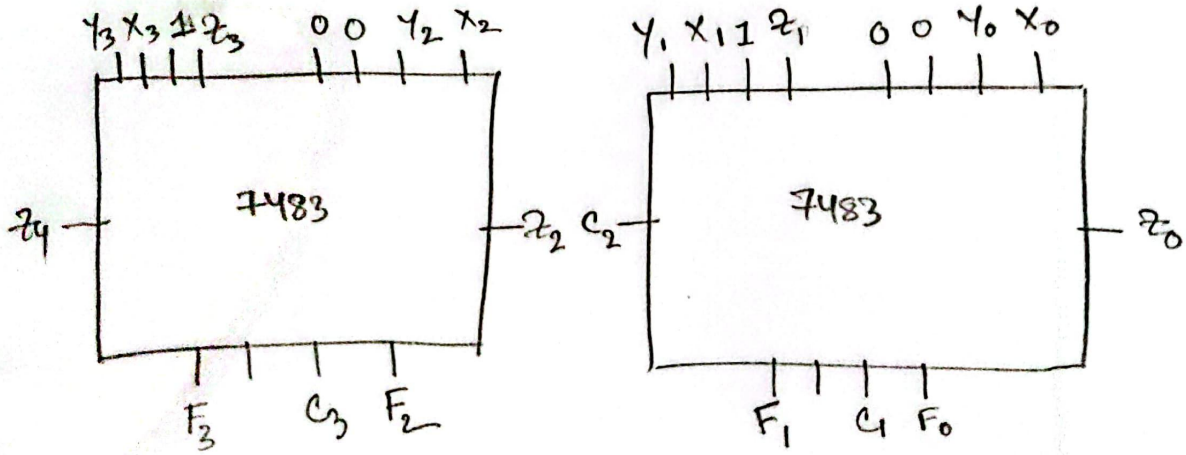
Design and demonstration of a 4bit ALU with three selection bits which performs the following operations:

| cs2(cin) | cs 1 | cs0 | function |
|---|---|---|---|
| 0 | 0 | 0 | Subtract with borrow |
| 0 | 0 | 1 | Decrement A |
| 1 | 0 | 0 | Subtract |
| 1 | 0 | 1 | Transfer A |
| X | 1 | 0 | AND |
| X | 1 | 1 | XOR |

## Detailed Design Steps with KMap

① For a 4bit ALU, a 4bit parallel adder is to be used. IC 7483 serves that purpose but it cannot be directly used as the carry propagation happens internally and we do not have access from outside to change it. The reason we need to change it is for logical operations we want to keep them off.

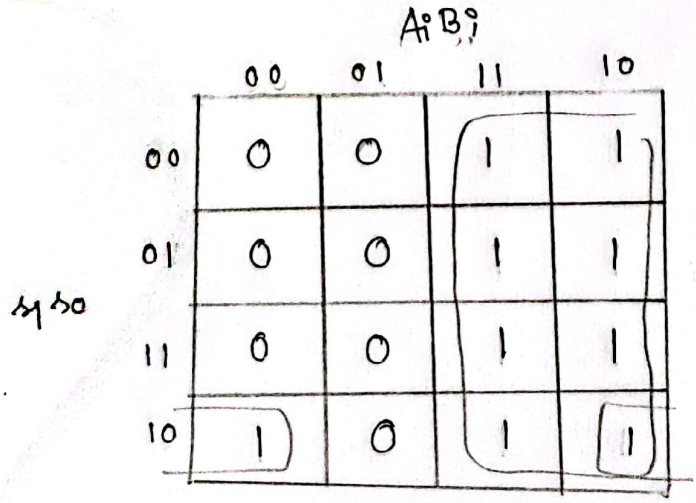
As a solution, two 7483 were used to simulate a 4 bit adder with access to the carry propagation. How that works is demonstrated below.



(11) The combinational circuit is designed on the basis of the truth table formulated in the next section of the report. The truth table minimizes the equation for $X_i$, $Y_i$, $Z_i$, the adder inputs of 4bit ALU. All of them are functions of $A_i$, $B_i$, $C_i$, cs2, cs1 and cs0.

| cs2 (Cin) | cs1 | cs0 | function | $X_i$ | $Y_i$ | $Z_i$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | A-B-1 | $A_i$ | $B_i'$ | $C_i$ |
| 0 | 0 | 1 | A-1 | $B_i A_i$ | ~~$B_i'$~~ 1 | $C_i$ |
| 1 | 0 | 0 | A-B | $A_i$ | 1 $B_i'$ | $C_i$ |
| 1 | 0 | 1 | A | $A_i$ | 1 | $C_i$ |
| X | 1 | 0 | ~~A~~B | $A_i + B_i'$ | $B_i'$ | 0 |
| X | 1 | 1 | A ⊕ B | $A_i$ | $B_i$ | 0 |

K-Map for $X_i$ :




$X_i = s_1 s_0' B_i' + A_i$

K Map for $Y_i$ :

| $B_i$ \ $s_1s_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |

$$Y_i = s_1's_0 + s_0B_i' + s_0B_i$$

$$= \overline{s_0 \oplus B_i} + s_1's_0$$

$$= s_0 \oplus B_i' + s_1's_0$$

K Map for $Z_i$ :

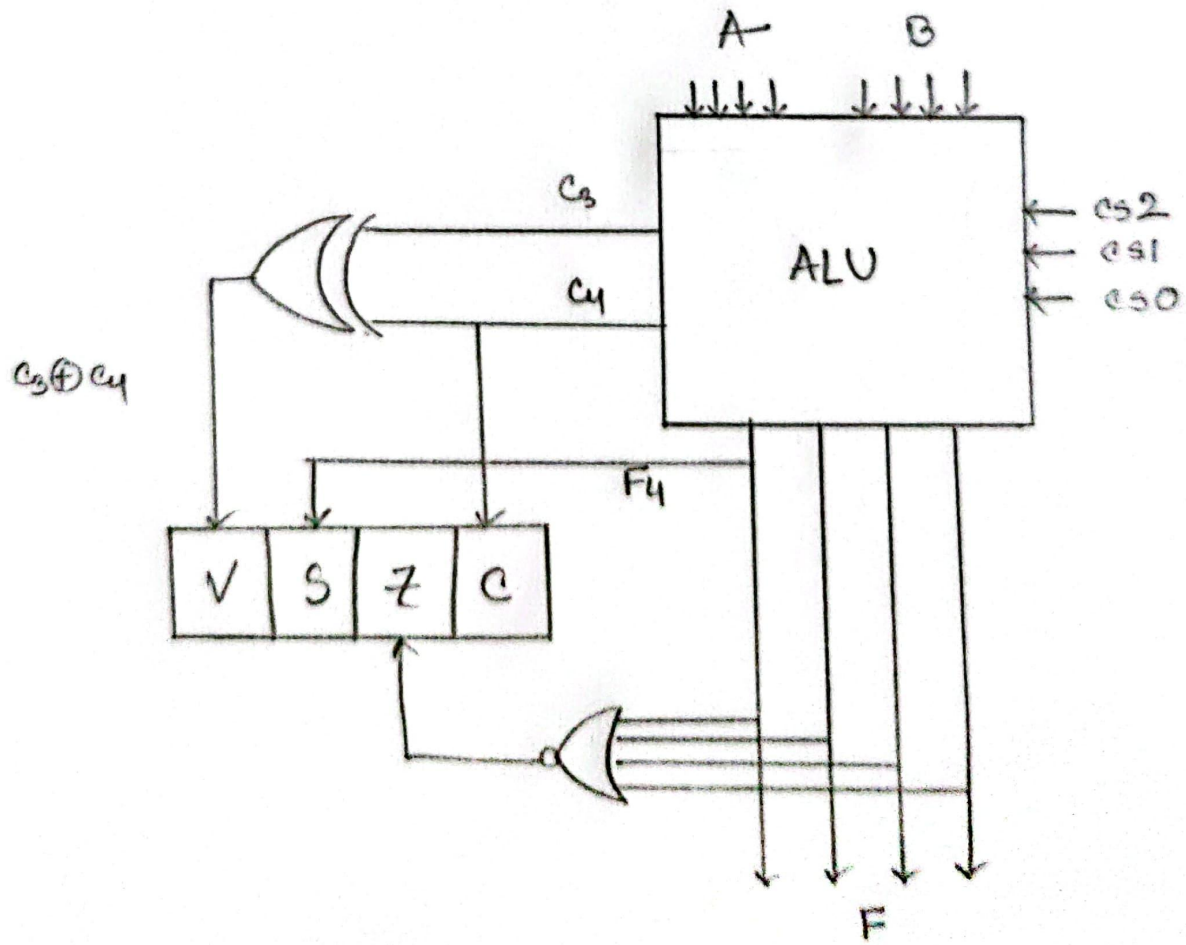
| $C_i$ \ $s_1s_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 |

$$Z_i = s_1'C_i$$

## Truth Table :

| $cs_2(C_{in})$ | $cs_1$ | $cs_0$ | $X_i$ | $Y_i$ | $Z_i$ | $F_i$ | Description |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | $A_i$ | $B_i'$ | $C_i$ | $A-B-1$ | Subtract with borrow |
| 0 | 0 | 1 | $A_i$ | 1 | $C_i$ | $A-1$ | Decrement A |
| 1 | 0 | 0 | $A_i$ | $B_i'$ | $C_i$ | $A-B$ | Subtract |
| 1 | 0 | 1 | $A_i$ | 1 | $C_i$ | A | Transfer A |
| X | 1 | 0 | $A_i+B_i'$ | $B_i'$ | 0 | $A \cdot B$ | AND |
| X | 1 | 1 | $A_i$ | $B_i$ | 0 | $A \oplus B$ | XOR |

## Block Diagram:



## ICs used with count as chart:

| IC | Gate | Count |
|---|---|---|
| 7404 | NOT | 1 |
| 7408 | AND | 3 |
| 7432 | OR | 3 |
| 7483 | 4bit Adder | 2 |
| 7486 | XOR | 2 |

## Simulator Used

Logisim - generic - 2.7.1

## Discussion

While implementing the circuit we had to change our design several times to reduce the number of ICs used. We reused some outputs so that we don't waste any I.C.

Implementing the 4bit Adder with 2 7483 needed a bit of calculation and research.