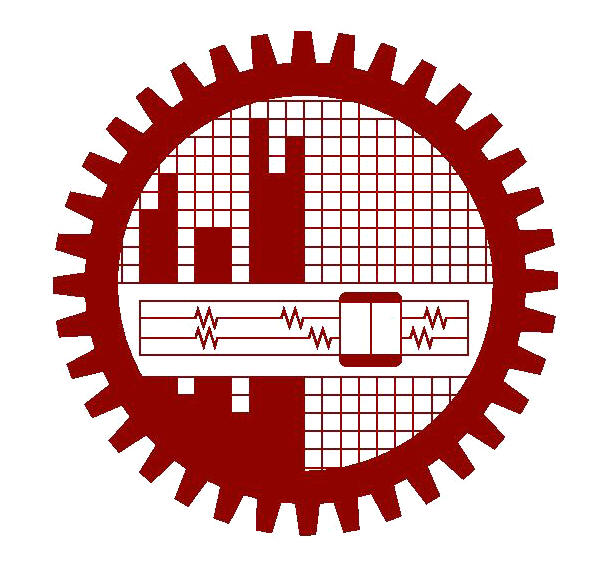
**Bangladesh University of Engineering and Technology**

CSE 306

Offline 1

**4-BIT ALU DESIGN & SIMULATION**

**GROUP : 01**

**SECTION : A2**

**Prepared By:**

**1605031**

**1605033**

**1605038**

**1605041**

**1605042**

*DEPARTMENT of COMPUTER SCIENCE and ENGINEERING*

**Introduction:**

In this assignment we have to design a 4-bit ALU.

An arithmetic logic unit (ALU) is a multioperation, combinational logic digital function. It can perform a set of basic arithmetic operation 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 2k operations.

**Problem Specification:**

|  |  |  |  |
| --- | --- | --- | --- |
| **cs2** | **cs1** | **cs0** | **Functions** |
| 0 | 0 | 0 | Transfer A |
| 0 | 0 | 1 | Add |
| 1 | 0 | 0 | Increment A |
| 1 | 0 | 1 | Add with carry |
| x | 1 | 0 | OR |
| x | 1 | 1 | XOR |

**Instructions Given:**

1. Required Flags:

* Carry
* Sign
* Overflow
* Zero

1. Flags will be affected as per the rules of Assembly Language.
2. Any SSI (AND, OR, NOT, XOR etc.) and MSI (MUX, Decoder, Adder etc.) chip can be used.

d) Emphasis should be given on efficiency of design and

minimization of ICs used.

**Truth Table:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **cs2** | **cs1** | **cs0** | **Xi** | **Y­i** | **Cin** | **Operation** | **Required Operation** |
| 0 | 0 | 0 | A | 0 | 0 | F = A | Transfer A |
| 0 | 0 | 1 | A | B | 0 | F = A + B | Add |
| 0 | 1 | 0 | Same as 110 | | | F = A + B | OR |
| 0 | 1 | 1 | Same as 111 | | | F = A ⊕ B | XOR |
| 1 | 0 | 0 | A | 0 | 1 | F = A + 1 | Increment A |
| 1 | 0 | 1 | A | B | 1 | F = A + B + 1 | Add with carry |
| 1 | 1 | 0 | A+B | 0 | 0 | F = A + B | OR |
| 1 | 1 | 1 | A⊕B | 0 | 0 | F = A ⊕ B | XOR |

**K-maps:**

1. **Determining X –**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | A | A | A⊕B | A+B |
|  | A | A | A⊕B | A+B |

Simplifying, we obtain:

X = + (+) + ⊕)

= + + + +

= + ( + ) + ( + )

= + ( + ) + ( + )

= [ + ( + )] + ( + )

= ( + + ) + ( + )

Therefore, X = ( + + ) + ( + )

1. **Determining Y –**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | B | 0 | 0 |
|  | 0 | B | 0 | 0 |

Therefore, Y =

1. **Determining Cin –**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 0 | 0 | 0 |
|  | 1 | 1 | 0 | 0 |

Therefore, Cin  =

**Block Diagram:**

When cs1 = 0 (Arithmetic Part)

1. **Transfer A:**

A 0

Parallel Adder

Cout Cin = 0

F = A

1. **Add:**

A B

Parallel Adder

Cout Cin = 0

F = A + B

1. **Increment:**

A 0

Parallel Adder

Cout Cin = 1

F = A + 1

1. **Add with carry:**

A B

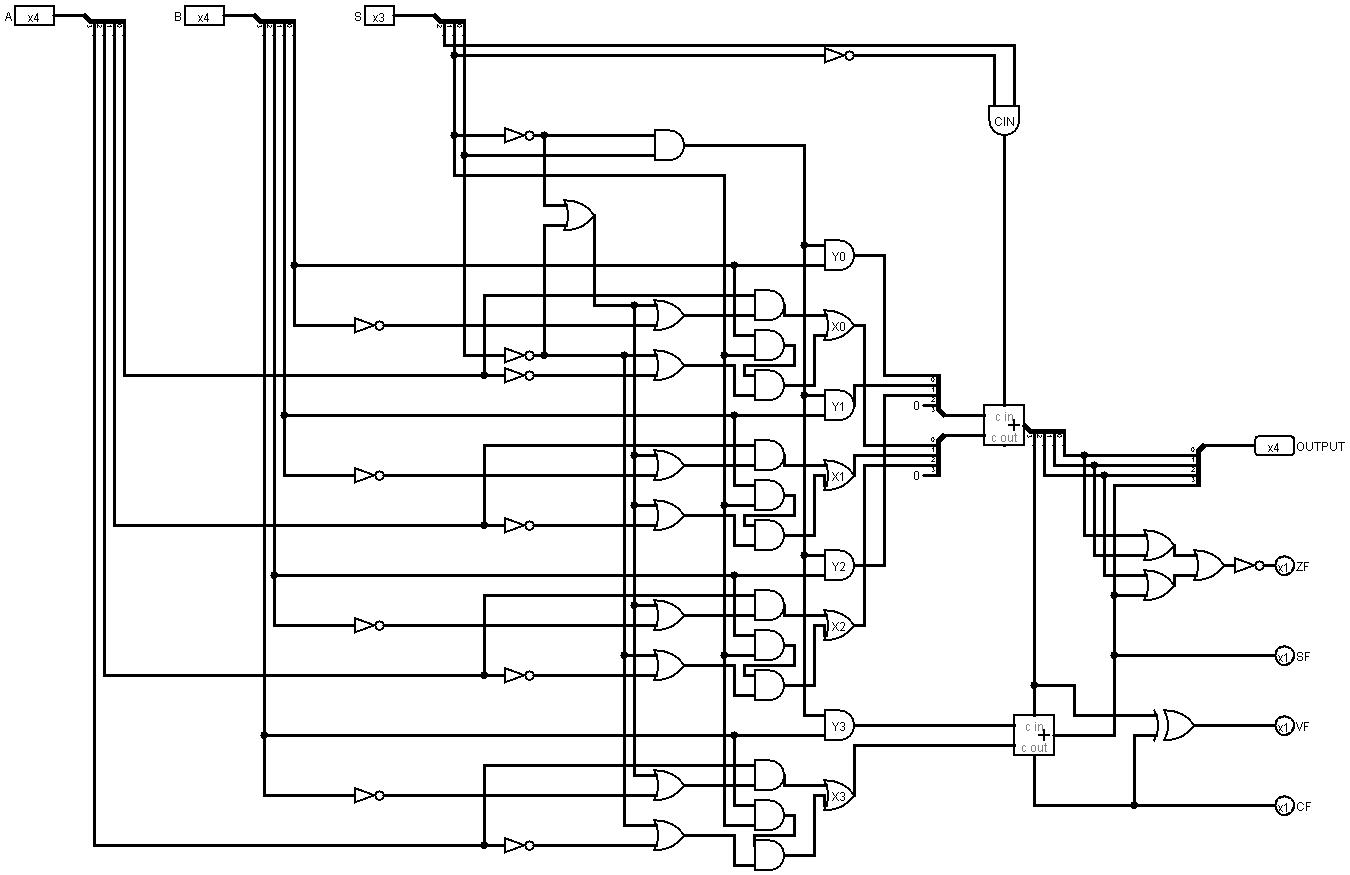
Parallel Adder

Cout Cin = 1

F = A + B + 1

When cs1 = 1 (Logical Part):

When cs1 = 1, the logical part is activated and cs2 is considered to be a don’t-care. We can force Cin to be 0 by making Cin = . Similarly, we force the cascaded input carries to the subsequent adders to be 0 as well.

****

**Fig: Circuit diagram**

**ICs used with count:**

|  |  |
| --- | --- |
| AND IC’s | 5 (and gates = 18. ICs = 18/4 = 4.5) |
| OR IC’s | 4 (or gates = 16; ICs = 16/4 = 4) |
| NOT IC’s | 2 (not gates = 12; ICs = 12/6 = 2) |
| 4-Bit Adder | 1 |
| XOR IC’s | 1 |
| 1-Bit Adder | 1 |
| **Total** | **14** |

**Simulator used:**

Logisim 2.7.1

**Discussion:**

While preparing our design we tried to use the minimum number of ICs we thought was possible. The design is as clean and easy-to-understand as we could make it. We used the simulation software mentioned above to test the circuit with various input data and verified that it works correctly. The required flags have also been properly incorporated into our design.To implement the assigned functions, we only required 6 of the total 8 possible selection-bit-combinations that were available to us. The two remaining selection-bit-combinations were ignored.