# Assignment 4.1

## 16-bit ALU

### Introduction: -

**ALU_TOP** is the fundamental building block of the processor, which is responsible for carrying out different functions: -

- **Arithmetic** functions through **ARITHMETIC_UNIT** block.

- **Logic** functions through **LOGIC_UNIT** block.

- **Shift** functions through **SHIFT _UNIT** block.

- **Comparison** functions through **CMP_UNIT** block.

And **Decoder Unit** responsibles for enable which Function to operate according to the highest Most significant **2-bit** of the ALU_FUNC control bus **ALU_FUNC [3:2].**

### Block Diagram
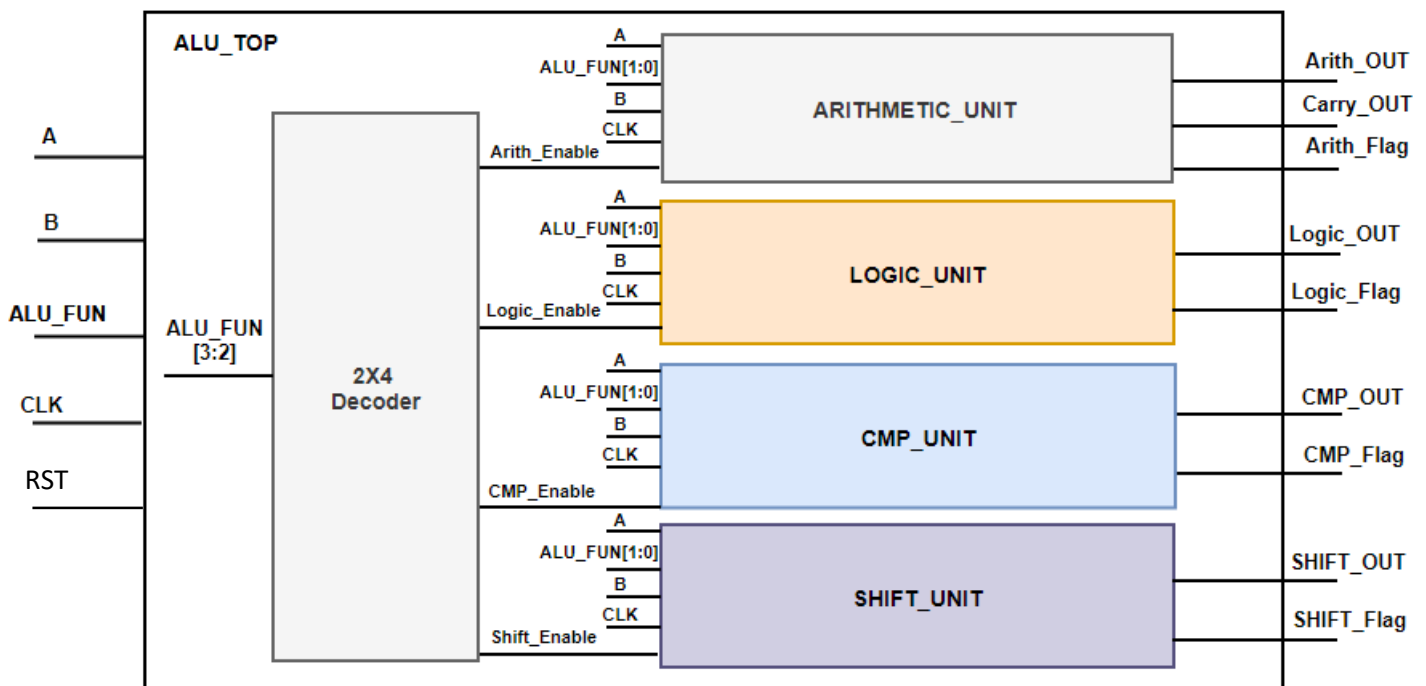
## TOP Module (ALU_TOP) Port Description:

| Signal Name | Width (bits) |
|---|---|
| A | parameterized |
| B | parameterized |
| ALU_FUNC | 4 |
| CLK | 1 |
| RST | 1 |
| Arith_OUT | parameterized |
| Carry_OUT | 1 |
| Arith_Flag | 1 |
| Logic_OUT | parameterized |
| Logic_Flag | 1 |
| CMP_OUT | parameterized |
| CMP_Flag | 1 |
| SHIFT_OUT | parameterized |
| SHIFT_Flag | 1 |

## Specifications: -

- All Outputs are registered.
- All registers are cleared using **Asynchronous active low reset**
- **Arith_flag** is activated "High" only when ALU performs one of the arithmetic operations (Addition, Subtraction, Multiplication, division), otherwise "LOW"
- **Logic_flag** is activated "High" only when ALU performs one of the Boolean operations (AND, OR, NAND, NOR), otherwise "LOW"
- **CMP_flag** is activated "High" only when ALU performs one of the Comparison operations (Equal, Greater than, less than) or NOP, otherwise "LOW"
- **Shift_flag** is activated "High" only when ALU performs one of the shifting operations (shift right, shift left), otherwise "LOW"

- The ALU function is carried out according to the value of the **ALU_FUN** input signal stated in the following table

## ALU_FUN Table

| ALU_FUN | Operation | ALU_OUT |
|---------|-----------|---------|
| 0000 | Arithmatic : **Addition** | |
| 0001 | Arithmatic : **Subtraction** | |
| 0010 | Arithmatic : **Multiplication** | |
| 0011 | Arithmatic : **Division** | |
| 0100 | Logic : **AND** | |
| 0101 | Logic : **OR** | |
| 0110 | Logic : **NAND** | |
| 0111 | Logic : **NOR** | |
| 1000 | **NOP** | **Equal to 0** |
| 1001 | CMP: **A = B** | **Equal to 1** else **Equal to 0** |
| 1010 | CMP: **A > B** | **Equal to 2** else **Equal to 0** |
| 1011 | CMP: **A < B** | **Equal to 3** else **Equal to 0** |
| 1100 | SHIFT: **A >> 1** | |
| 1101 | SHIFT: **A << 1** | |
| 1110 | SHIFT: **B >> 1** | |
| 1111 | SHIFT: **B << 1** | |

**Hint:** Use Case statement to describe the behavior of this table and use default case if needed.

**Hint:** You can use if statement inside case branches.

**Note:** Arith_Enable, Logic_Enable, SHIFT_Enable and CMP_Enable are called block enable which responsible for enabling the function of the block or not.

## Decoder Truth Table

| ALU_FUNC[3:2] | Arith_En | Logic_En | CMP_EN | SHIFT_EN |
|:---:|:---:|:---:|:---:|:---:|
| 00 | 1 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | 1 |

**Hint:** How to use the enable signal inside the code of each block.

```
always @(*)
  begin
   if(Arith_Enable)
    begin
     case(ALU_FUN)
       2'b00 : {ALU_Carry, ALU_Arith } = A + B ;

       ………..
      endcase
    end
   else
    begin
      ALU_Arith = 16'b0 ;
    End
 End
```

## Requirements: -

1. Write a Verilog Codes of the following **6 modules**
   - **ARITHMETIC_UNIT**
   - **LOGIC_UNIT**.
   - **SHIFT _UNIT**
   - **CMP_UNIT**
   - **Decoder Unit**
   - **ALU_TOP**

2. Write a testbench to test all the ALU functions with operating clock frequency 100 KHz with duty cycle 40% low and 60% high



60%          40%