Assignment 4.0

16-bit ALU

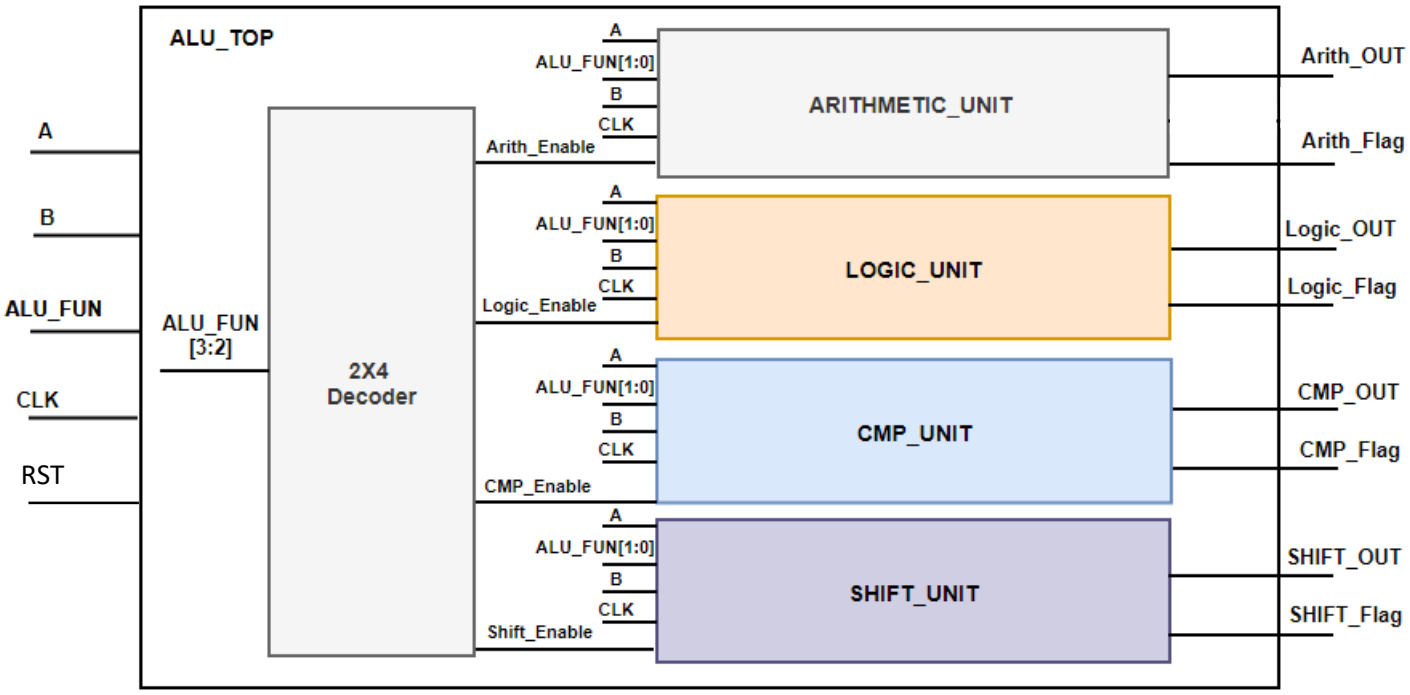
Introduction: -

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

- **Signed 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
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 (Signed Addition, Signed Subtraction, Signed Multiplication, Signed 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 : **Signed Addition**	
0001	Arithmatic : **Signed Subtraction**	
0010	Arithmatic : **Signed Multiplication**	
0011	Arithmatic : **Signed 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**	

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**

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

Hints: -

1- Use Case statement to describe the behavior of this table and use default case if needed. You can also use if statement inside case branches.

2- How to use the enable signal inside the code of each block.

```
always @(*)
  begin
    if(Arith_Enable)
     begin
       case(ALU_FUN)
         2'b00 : ALU_Arith = A + B ;
         ……..
        endcase
     end
   else
     begin
       ALU_Arith = 16'b0;
     end
 end
```

3- How to do signed arithmetic operations.

- **Operands must be defined as signed data types**

```
module ARITHMETIC_UNIT (
input  wire  signed  [IN_DATA_WIDTH-1:0]  A,
input  wire  signed  [IN_DATA_WIDTH-1:0]  B,
…..  );
```

4- How to deal with negative value in Testbench?

1- **Declare TB signals responsible for arithmetic operations as signed**

```
/*****************************************************************************/
/******************************** TB Signals *********************************/
/*****************************************************************************/

   reg    signed [OP_DATA_WIDTH_TB-1:0]     A_TB          ;
   reg    signed [OP_DATA_WIDTH_TB-1:0]     B_TB          ;
   wire   signed [Arith_OUT_WIDTH_TB-1:0]   Arith_OUT_TB  ;
```

2- **Apply the negative values using its 2's complement or use negative decimal value as shown for the below test case**

```
    $display ("*** TEST CASE 1 -- Addition -- NEG + NEG ***");

// A_TB = -'d4
A_TB = 'shFFFC;
// B_TB = -'d10
B_TB = 'shFFF6;
ALU_FUNC_TB = 4'b0000;

#CLK_PERIOD

   if (Arith_OUT_TB == -'d14 && Flags == 4'd8)
      $display ("Addition %0d + %0d IS PASSED = %0d",A_TB,B_TB,Arith_OUT_TB) ;
   else
      begin
       $display ("Addition %0d + %0d IS FAILED = %0d",A_TB,B_TB,Arith_OUT_TB) ;
      end
```

FFFC is the 2's complement of -4
&
FFF6 is the 2's complement of -10

3- **Expected Output**

```
# *** TEST CASE 1 -- Addition -- NEG + NEG ***
# Addition -4 + -10 IS PASSED = -14
```

Note: Use below online website for decimal to 2's complement conversion

https://www.rapidtables.com/convert/number/decimal-to-hex.html

Requirements: -

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

2. Write a testbench to test all the ALU functions to include the following 28 test cases: -
   - Signed Arithmetic Addition: A is Negative & B is Negative
   - Signed Arithmetic Addition: A is Positive & B is Negative
   - Signed Arithmetic Addition: A is Negative & B is Positive
   - Signed Arithmetic Addition: A is Positive & B is Positive
   - Signed Arithmetic Subtraction: A is Negative & B is Negative
   - Signed Arithmetic Subtraction: A is Positive & B is Negative
   - Signed Arithmetic Subtraction: A is Negative & B is Positive
   - Signed Arithmetic Subtraction: A is Positive & B is Positive
   - Signed Arithmetic Multiplication: A is Negative & B is Negative
   - Signed Arithmetic Multiplication: A is Positive & B is Negative
   - Signed Arithmetic Multiplication: A is Negative & B is Positive
   - Signed Arithmetic Multiplication: A is Positive & B is Positive
   - Signed Arithmetic Division: A is Negative & B is Negative
   - Signed Arithmetic Division: A is Positive & B is Negative
   - Signed Arithmetic Division: A is Negative & B is Positive
   - Signed Arithmetic Division: A is Positive & B is Positive
   - Logical Operations (AND, NAND, OR, NOR)
   - Compare Operations (Equal, Greater, Less)
   - Shift Operations (A shift right, A shift left, B shift right, B shift left)
   - NOP

3. Simulate your design with operating clock frequency 100 KHz with duty cycle 40% low and 60% high