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: -

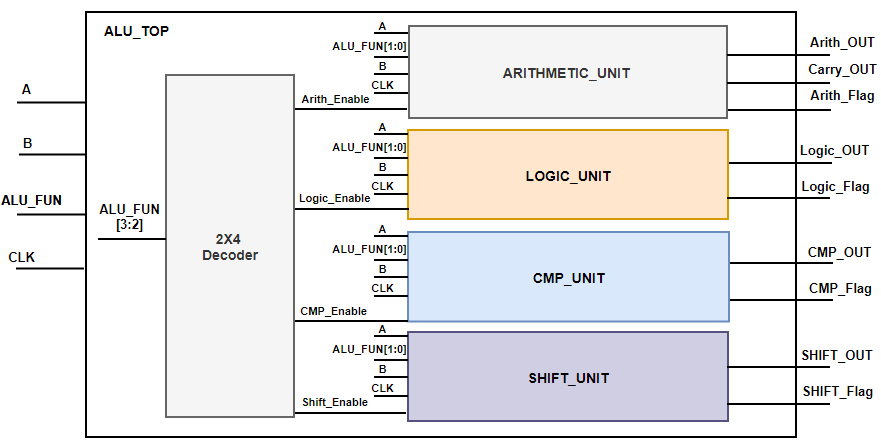
**- Arithmetic** functions through **ARITHMETIC\_UNIT** block.

**- 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**:

|  |  |
| --- | --- |
| Width (bits) | Signal Name |
| parameterized | A |
| parameterized | B |
| 4 | ALU\_FUNC |
| 1 | CLK |
| parameterized | Arith\_OUT |
| 1 | Carry\_OUT |
| 1 | Arith\_Flag |
| parameterized | Logic\_OUT |
| 1 | Logic\_Flag |
| parameterized | CMP\_OUT |
| 1 | CMP\_Flag |
| parameterized | SHIFT\_OUT |
| 1 | SHIFT\_Flag |

**Specifications: -**

* All Outputs are registered.
* **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\_OUT | Operation | ALU\_FUN |
|  | Arithmatic : **Addition** | 0000 |
|  | Arithmatic : **Subtraction** | 0001 |
|  | Arithmatic : **Multiplication** | 0010 |
|  | Arithmatic : **Division** | 0011 |
|  | Logic : **AND** | 0100 |
|  | Logic : **OR** | 0101 |
|  | Logic : **NAND** | 0110 |
|  | Logic : **NOR** | 0111 |
| Equal to 0 | **NOP** | 1000 |
| Equal to 1 | CMP: **A = B** | 1001 |
| Equal to 2 | CMP: **A > B** | 1010 |
| Equal to 3 | CMP: **A < B** | 1011 |
|  | SHIFT: **A >> 1** | 1100 |
|  | SHIFT: **A << 1** | 1101 |
|  | SHIFT: **B >> 1** | 1110 |
|  | SHIFT: **B << 1** | 1111 |

**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**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SHIFT\_EN** | **CMP\_EN** | **Logic\_En** | **Arith\_En** | **ALU\_FUNC[3:2]** |
| **0** | **0** | **0** | **1** | **00** |
| **0** | **0** | **1** | **0** | **01** |
| **0** | **1** | **0** | **0** | **10** |
| **1** | **0** | **0** | **0** | **11** |

**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

