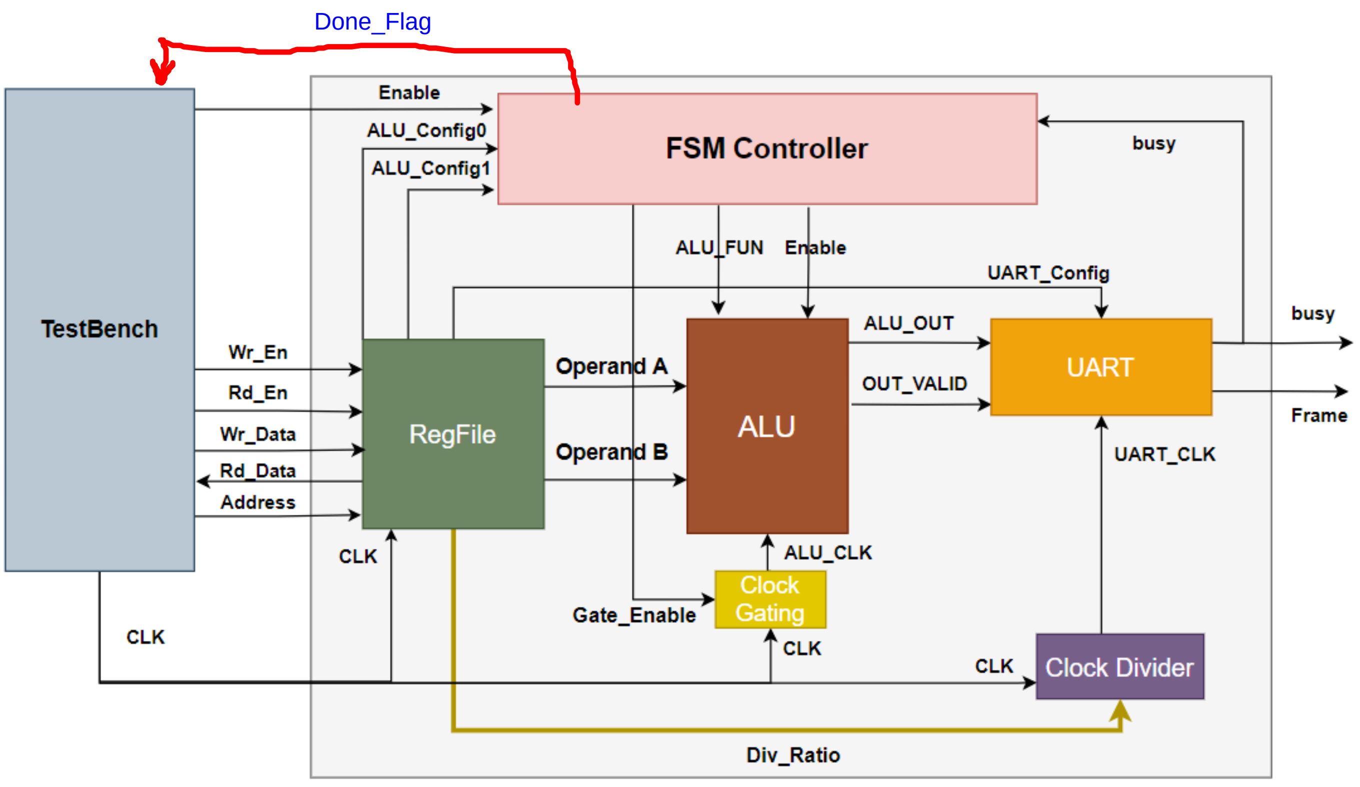
Final System

Introduction: -

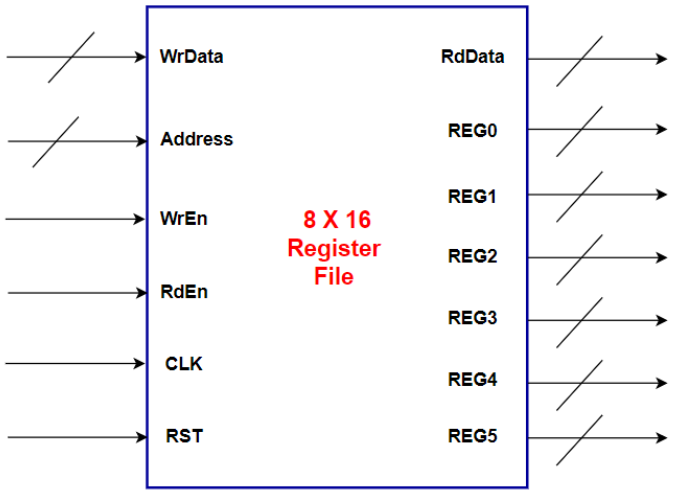
- **This System Is Our Final Target: -**



- **Description: -**
  - **This system contains 6 blocks: -**
    - **RegFile**
    - **ALU**
    - **UART**
    - **FSM Controller**
    - **Clock Divider**
    - **Clock Gating**

1) RegFile: -

- **Block Interface: -**



- **Signal Description: -**

Port	Direction	Width	Description	Connected to
CLK	IN	1	Clock Signal	TOP Input Port
RST	IN	1	Active Low Async Reset	TOP Input Port
Address	IN	Parameterized (default : 4 bits)	Address bus	TOP Input Port
WrEn	IN	1	Write Enable	TOP Input Port
RdEn	IN	1	Read Enable	TOP Input Port
WrData	IN	Parameterized (default : 8 bits)	Write Data Bus	TOP Input Port
RdData	OUT	Parameterized (default : 8 bits)	Read Data Bus	TOP output Port
REG0	OUT	Parameterized (default : 8 bits)	Register at Address 0x0	ALU (A)
REG1	OUT	Parameterized (default : 8 bits)	Register at Address 0x1	ALU (B)
REG2	OUT	Parameterized (default : 8 bits)	Register at Address 0x2	FSM Controller (ALU Config0)
REG3	OUT	Parameterized (default : 8 bits)	Register at Address 0x3	FSM Controller (ALU Config1)
REG4	OUT	Parameterized (default : 8 bits)	Register at Address 0x4	UART (UART_Config)
REG5	OUT	Parameterized (default : 8 bits)	Register at Address 0x5	Clock Divider (Div_Ratio)

- **Reserved Registers Description: -**

1) **REG0 (Address: 0x0)** ALU Operand A

2) **REG1 (Address: 0x1)** ALU Operand B

3) **REG2 (Address: 0x2)** ALU Config0

   1. **Each bit value represents enable certain ALU function in case of value = "1" and disable the function in case of value = "0"**

      REG2[0]: Arithmetic Adding
      REG2[1]: Arithmetic Subtraction
      REG2[2]: Arithmetic Multiplication
      REG2[3]: Arithmetic Division
      REG2[4]: Logical AND
      REG2[5]: Logical OR
      REG2[6]: Logical NAND
      REG2[7]: Logical NOR

4) **REG3 (Address: 0x3)** ALU Config1

   2. **Each bit value represents enable certain ALU function in case of value = "1" and disable the function in case of value = "0"**

      REG3[0]: Logical XOR
      REG3[1]: Logical XNOR
      REG3[2]: CMP (A = B)
      REG3[3]: CMP (A > B)
      REG3[4]: CMP (A < B)
      REG3[5]: Shift (A >> 1)
      REG3[6]: Shift (A << 1)
      REG3[7]: No Operation

5) **REG4 (Address: 0x4)**

UART Config

REG4[0]: Parity Enable
REG3[1]: Parity Type
REG3[2:7]: Not Used

6) **REG5 (Address: 0x5)**

Div Ratio

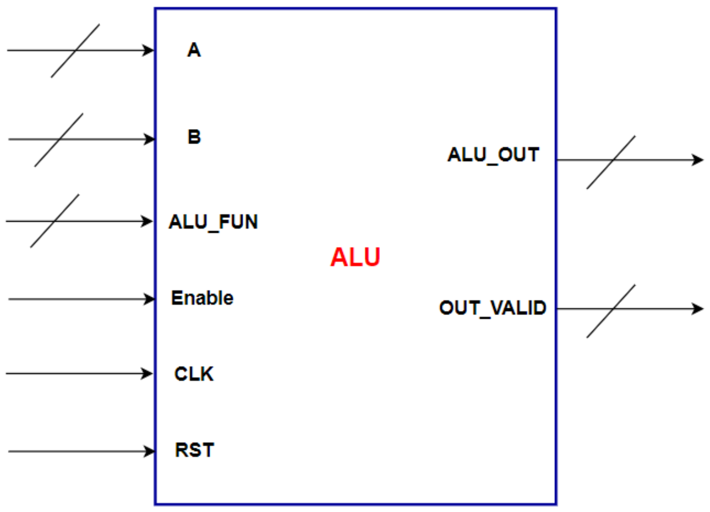
REG5[0:3]: Division ratio
REG3[4:7]: Not Used

<u>Modifications: -</u>

3. **Refer to Assignment 4.2, you need to add the following registers as outputs on the Register File interface: -**
   1) **Register at address 0x0 on port REG0**
   2) **Register at address 0x1 on port REG1**
   3) **Register at address 0x2 on port REG2**
   4) **Register at address 0x3 on port REG3**
   5) **Register at address 0x4 on port REG4**
   6) **Register at address 0x5 on port REG5**

2) ALU:

- **Block Interface: -**



- **Signal Description: -**

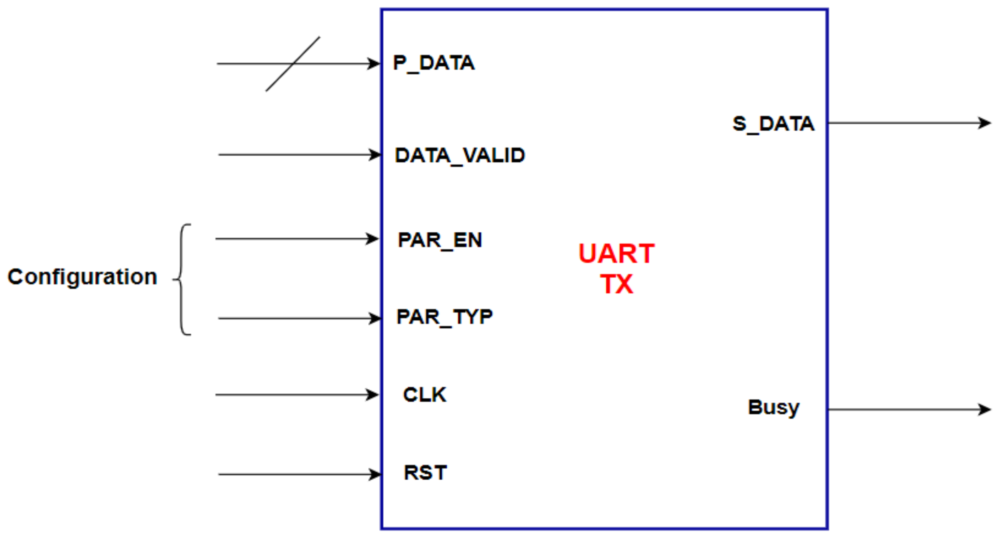
Port	Direction	Width	Description	Connected to
CLK	IN	1	Clock Signal	TOP Input Port
RST	IN	1	Active Low Async Reset	TOP Input Port
A	IN	Parameterized (default : 8 bits)	Operand A	RegFile (REG0)
B	IN	Parameterized (default : 8 bits)	Operand B	RegFile (REG1)
ALU_FUN	IN	Parameterized (default : 4 bits)	ALU Function	FSM Controller (ALU_FUN)
Enable	IN	1	ALU Enable	FSM Controller (ALU_Enable)
ALU_OUT	OUT	Parameterized (default : 8 bits)	ALU Result	UART (P_DATA)
OUT_VALID	OUT	1	Result Valid	UART (DATA_VALID)

Modifications: -

4. Refer to Assignment 3, you need to add the following modifications: -
    1. Replace all the flags (Arith_flag, Logic_flag, CMP_flag, Shift_flag) by OUT_VALID signal
    2. All the outputs (ALU_OUT, OUT_VALID) are registered
    3. Add Enable signal,
        - when Activated (Enable = 1’b1)
            - ALU_OUT = result of the operation
            - VALID_OUT = 1’b1
        - when deactivated
            - ALU_OUT = 0
            - VALID_OUT = 0

3) UART: -

- **Block Interface: -**



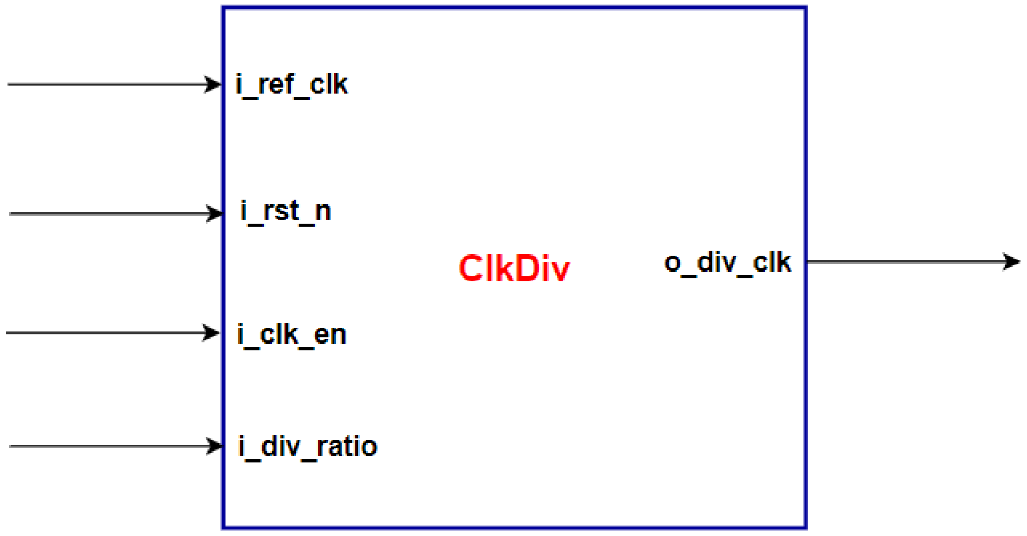
- **Signal Description: -**

Port	Direction	Width	Description	Connected to
CLK	IN	1	Clock Signal	TOP Input Port
RST	IN	1	Active Low Async Reset	TOP Input Port
PAR_EN	IN	1	Parity Enable	RegFile (UART_Config[0])
PAR_TYP	IN	1	Parity Type	RegFile (UART_Config[1])
P_DATA	IN	Parameterized (default : 8 bits)	Parallel IN Data	ALU (ALU_OUT)
DATA_VALID	IN	1	IN Data Valid	ALU (OUT_VALID)
S_DATA	OUT	1	frame serial bits	TOP Output Port
Busy	OUT	1	Uart status signal	1) TOP Output Port 2) FSM Controller (UART_Status)

No Modifications is needed

4) Clock Divider: -

- **Block Interface: -**



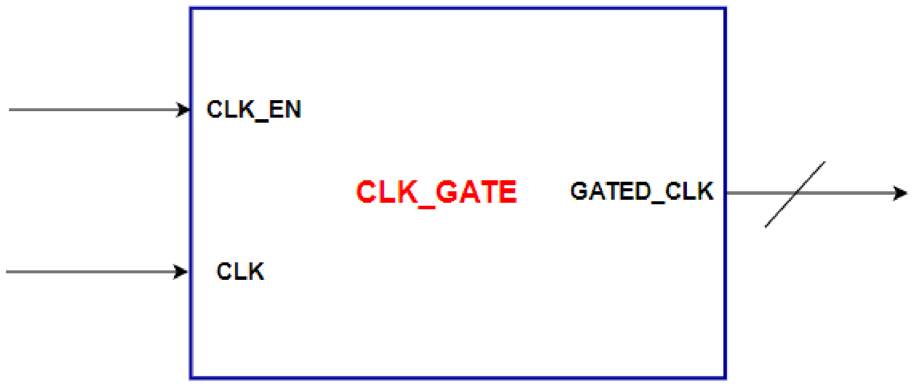
- **Signal Description: -**

Port	Direction	Width	Description	Connected to
I_ref_clk	IN	1	Clock Signal	TOP Input Port
I_rst_n	IN	1	Active Low Async Reset	TOP Input Port
I_clk_en	IN	1	Clock divider enable	TOP Input Port
I_div_ratio	IN	Parameterized (default : 4 bits)	Division ratio	RegFile (Div_Ratio)
O_div_clk	out	1	Divided clock	UART (CLK)

No Modifications is needed

5) Clock Gating: -

- **Block Interface: -**



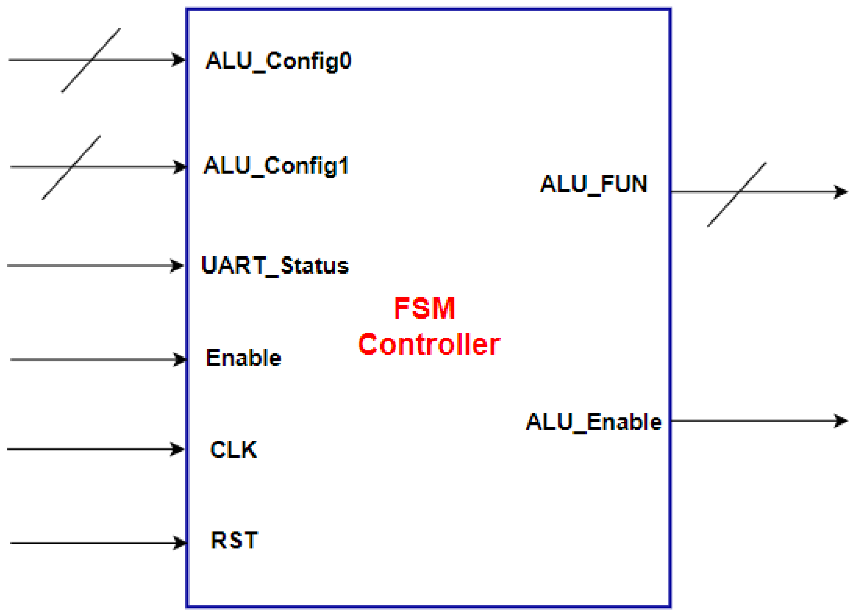
- **Signal Description: -**

Port	**Direction**	**Width**	**Description**	**Connected to**
CLK	IN	1	Clock Signal	TOP Input Port
CLK_EN	IN	1	Clock Enable	FSM Controller (Gate_Enable)
GATED_CLK	out	1	Gated Clock signal	ALU (CLK)

No Modifications is needed

6) FSM_Controller: -

- **Block Interface: -**



- **Signal Description: -**

Port	Direction	Width	Description	Connected to
CLK	IN	1	Clock Signal	TOP Input Port
RST	IN	1	Active Low Async Reset	TOP Input Port
UART_Status	IN	1	Uart status signal	UART (busy)
Enable	IN	1	Parity Type	TOP Input Port
ALU_Config0	IN	Parameterized (default : 8 bits)	ALU Configuration Register 0	ALU (ALU_OUT)
ALU_Config0	IN	Parameterized (default : 8 bits)	ALU Configuration Register 1	ALU (OUT_VALID)
ALU_FUN	OUT	Parameterized (default : 4 bits)	ALU Function signal	ALU (ALU_FUN)
ALU_Enable	OUT	1	ALU Enable signal	ALU (Enable)

System Specifications: -

- The system need to do some ALU functions based on the values stored in ALU_config0 and ALU_Config1 registers in Register File and send the result of the ALU operation serially through UART protocol
- Minimum of ALU operations equal 0
  - (register at 0x0 = 8'h0 && register at 0x1= 8'h0)
- Maximum of ALU operations equal 16
  - (register at 0x0 = 8'hFF && register at 0x1= 8'hFF)
- Reference clock is 20 MHz
- Div_ratio can be 3 or 6 or 8
- Clock Divider is always on (clock divider enable = 1)

Sequence of Operation (Must include in the testbench): -

- Initially 6 write operation is needed to write the configurations in registers from address 0x0 to 0x5
- Initially FSM_Controller is disabled (Enable = 0) until the write configurations registers is done
- After Write operations, the FSM_Contrller will be enabled to enable the ALU to perform certain operation and then send the result through the UART
- While UART is processing the result, the FSM_Controller will disable the ALU & disable ALU Clock until UART finishes and busy signal get deactivated.
- Once UART busy get deactivated, the FSM_Controller start to enable the ALU again to perform the next operation and so on until checking all the ALU_config0 and ALU_Config1 bits.