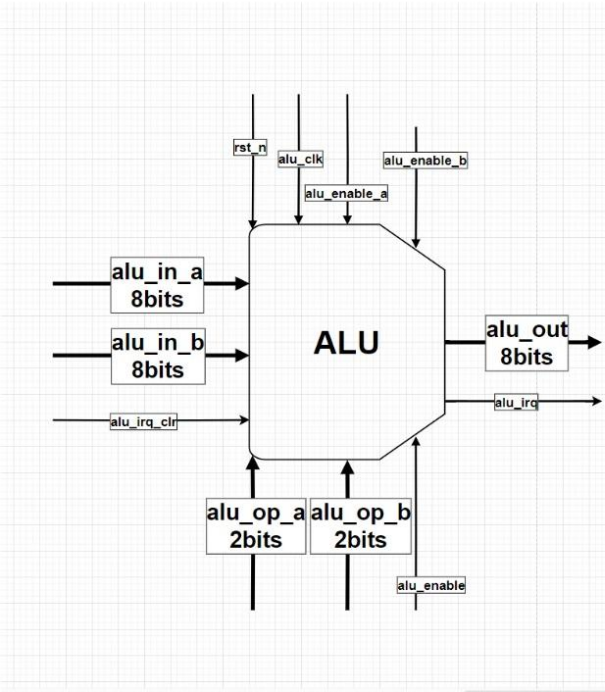
Digital IC Verification Track - New Capital Branch

Assignment 1: Verification Requirements Extraction

Submitted to: Eng. Mahmoud Saeed Elbosily

Group 2

Ahmed Ashraf Mohamed El-Hady
Youssef Nasser Abd El-Aal Abd El-Fattah
Tasneem Abo El-Esaad Abd El-Razik



Operation a == alu_enable_a && alu_enable				
alu_op_a	alu_out	alu_in_a	alu_in_b	alu_irq == 1'b1
2'b00	AND	ALL	!=8'h0	alu_out == 8'hFF
2'b01	NAND	!=8'hFF	!=8'h03	alu_out == 8'h00
2'b10	OR	ALL	ALL	alu_out == 8'hF8
2'b11	XOR	ALL	ALL	alu_out == 8'h83

Operation b == alu_enable_b && alu_enable				
alu_op_b	alu_out	alu_in_a	alu_in_b	alu_irq == 1'b1
2'b00	XNOR	ALL	ALL	alu_out == 8'hF1
2'b01	AND	ALL	!=8'h03	alu_out == 8'hF4
2'b10	NOR	!=8'hF5	ALL	alu_out == 8'hF5
2'b11	OR	ALL	ALL	alu_out == 8'hFF

Verification Requirements Extraction

Note: all inputs should be driven before the rising edge of the clock, also all inputs and outputs are synchronized to the positive edge clock.

- **alu_in_a**
  - Driven constrained randomly.
  - Cover that the forbidden values aren't generated.
  - Cover that all input pins toggle from 1 to 0 and vice versa.
- **alu_in_b**
  - Driven constrained randomly.
  - Cover that the forbidden values are not generated.
  - Cover that all input pins toggle from 1 to 0 and vice versa.
- **rst_n**
  - Drive it with 0 at the start, then it can be 0 or 1 at different times during normal operation independent of the clock, most of the time it is high.
  - Cover that reset is de-asserted and asserted.
  - Check that outputs are driven to all 0's immediately when it is low.
  - Check that outputs are as expected when it is high.
- **alu_clk**
  - Driven positive edge trigger clock with a period of 31.25 ns, (1000/32MHz). *(Note: timescale 1ns/1ps)*

- **alu_enable**
  - Drive it with 1 or 0, most of the time it is high.
  - Cover that ***alu_enable***, ***alu_enable_a*** and ***alu_enable_b*** toggle from 1 to 0 and vice versa
  - Check that all outputs are maintained when ***alu_enable*** is low.
- **alu_enable_a**
  - Drive it with 1 or 0, most of the time ***alu_enable_a*** dosen't equal ***alu_enable_b***.
  - Cover that ***alu_enable_a*** toggles with all variations of ***alu_enable***.
  - Check that when ***alu_enable_a*** is high, ***alu_out*** is generated according to operation a.
- **alu_enable_b**
  - Drive it with 1 or 0, most of the time ***alu_enable_a*** dosen't equal ***alu_enable_b***.
  - Cover that ***alu_enable_b*** toggles with all variations of ***alu_enable***.
  - Check that when ***alu_enable_b*** is high, ***alu_out*** is generated according to operation b
- **alu_op_a**
  - Driven with all possible combinations randomly.
  - Cover all possible combinations while ***alu_enable*** and ***alu_enable_a*** are high, and cover all other possible cases, listed in design requirements.
  - Check that when the operation is NAND while ***alu_enable*** and ***alu_enable_a*** are asserted to high, ***alu_out*** will never be 8'h00.
- **alu_op_b**
  - Driven with all possible combinations randomly.
  - Cover all possible combinations while ***alu_enable*** and ***alu_enable_b*** are high, and cover all other possible cases, listed in design requirements.

- **alu_irq_clr**
  - Drive it with 1 while ***alu_irq*** is de-asserted.
  - Cover ***alu_irq_clr*** that is asserted and ***alu_irq*** is de-asserted.
  - Check if ***alu_irq*** is still 0 or not.
  - Drive it with 1 after ***alu_irq*** is asserted.
  - Cover that ***alu_irq_clr*** is asserted and ***alu_irq*** is also asserted.
  - Check that ***alu_irq*** is cleared, equals 0.
  - Drive it with 0 regardless of the value of ***alu_irq***, 1 or 0.
  - Cover that ***alu_irq_clr*** is low when ***alu_irq*** is asserted and de-asserted at different times.
  - Check that ***alu_irq*** is maintained.
- **alu_out**
  - Cover that all ***alu_out*** combinations that asserts ***alu_irq*** are generated.
  - Cover that ***alu_out*** pins toggles.
  - Check that ***alu_out*** is asserted after one clock cycle from applying the inputs.
  - Check that ***alu_out*** is maintained when ***alu_enable_a*** and ***alu_enable_b*** are asserted to low at the same time.

- **alu_irq**
  - Check that ***alu_irq*** is asserted to high at the same cycle as ***alu_out***.
  - Check that ***alu_irq*** isn't asserted unless ***alu_out*** is one of the cases listed in the design requirements.
  - Drive ***alu_irq_clr*** to high while there are two successive events that assert ***alu_irq***.
  - Cover that ***alu_irq_clr*** is asserted while there is two successive events that assert ***alu_irq***.
  - Check that events have priority over ***alu_irq_clr***
    - Check that ***alu_irq*** is low when ***alu_irq_clr*** high, taking into consideration that there isn't two successive events.
    - Check that ***alu_irq*** is still asserted when there is two successive events while ***alu_irq_clr*** is high.
  - Check that ***alu_irq*** is de-asserted after one clock cycle from the assertion of ***alu_irq_clr***.