4-bit ALSU with 40 Operations in Modular Design

Ahmed Hisham Abdelfattah
ECE Department

Table Of Contents:

1.Introduction:

- **This Document Presents the Design and Implementation of 4-bit ALSU with 40 Operations in Modular Design in Verilog HDL**

- **This Project is made in modular Design as I made the Architecture of Arithmetic & Arithmetical Shift Unit and the Architecture of Logical and Logic Shift unit and All the Blocks in the system.**

- **And this Project includes also like 4-bit Divider and its Architecture, 4-bit Multiplier, Incrementer, Decrementer taking in consideration the Negative Sign in Operations Like Subtractor and Decrementer by making an output (Negative Sign Flag).**

- **This Project also Contains Logic Operations like AND, OR, XOR, XNOR and a lot of other logical operations.**

- **This project is mainly designed in around 3000 lines of Code using Verilog HDL as I made a design and Test Bench for each module individually and for the whole system (you Can See them from my Repo.)**

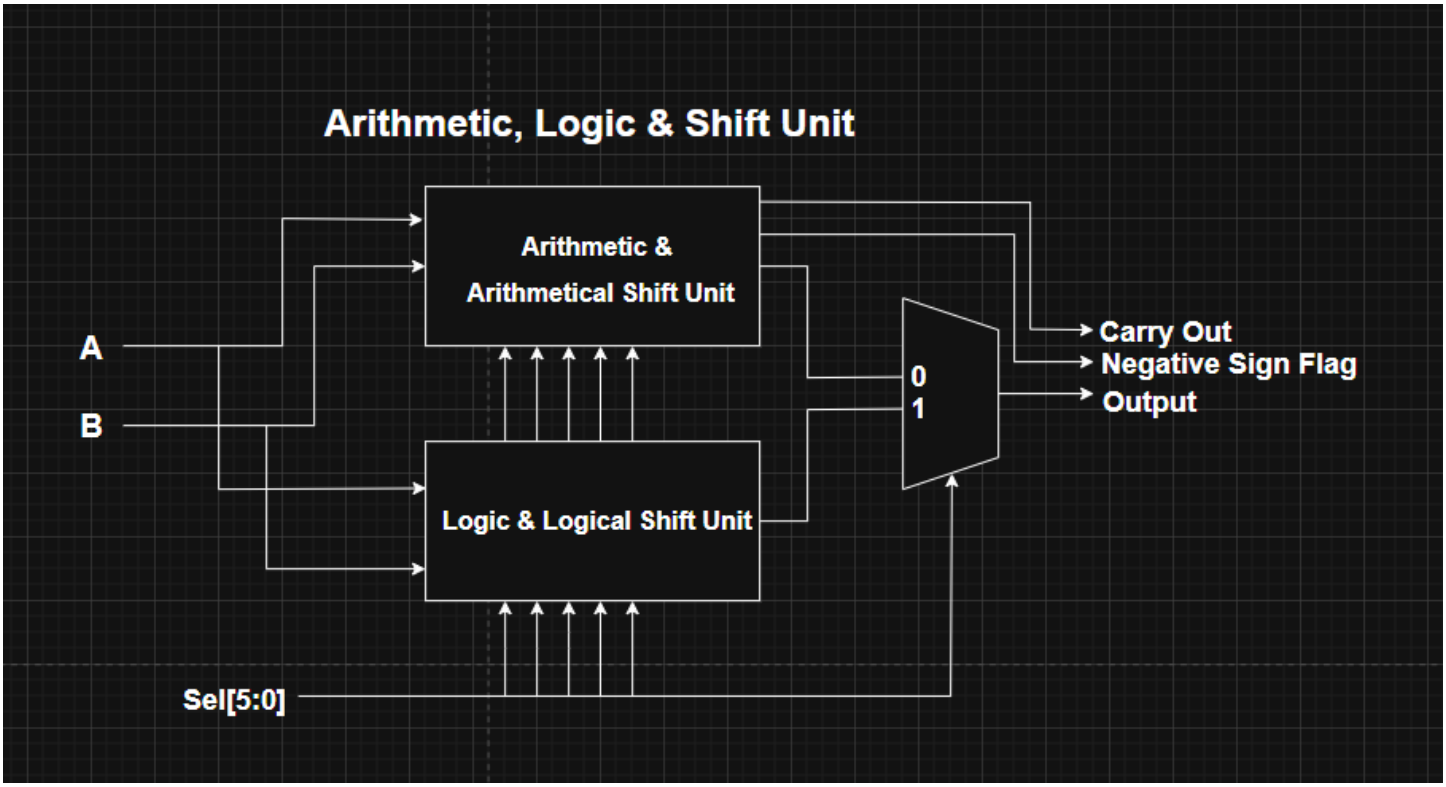
- **Repo: Repo ALSU 40-Operations**

- **LinkedIn Account: LinkedIn Ahmed Hisham Abdelfattah**

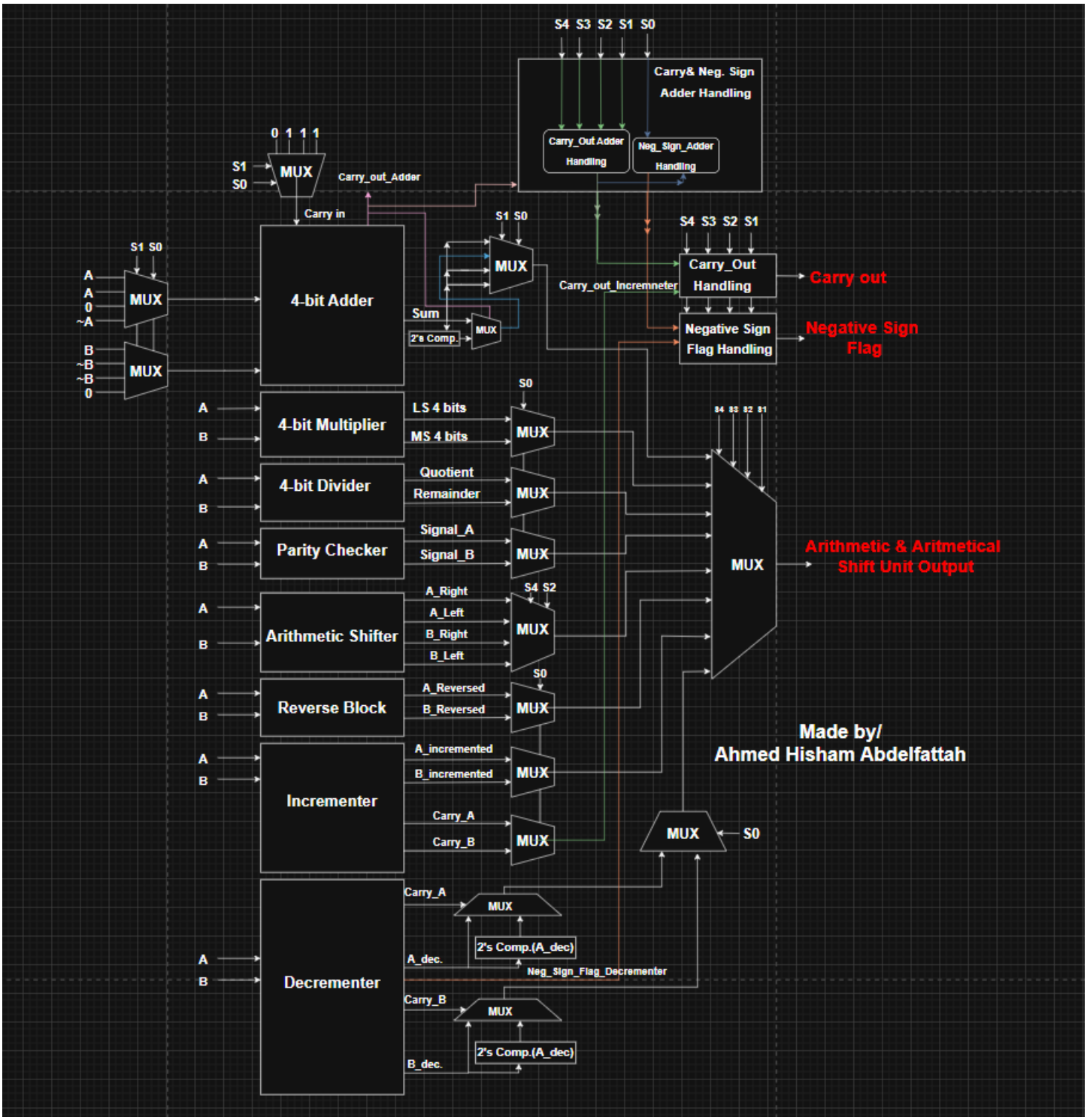
- **For any Questions, feel free to reach out at: ahmed.hesham2005.bu@gmail.com**

2.Design Architecture:

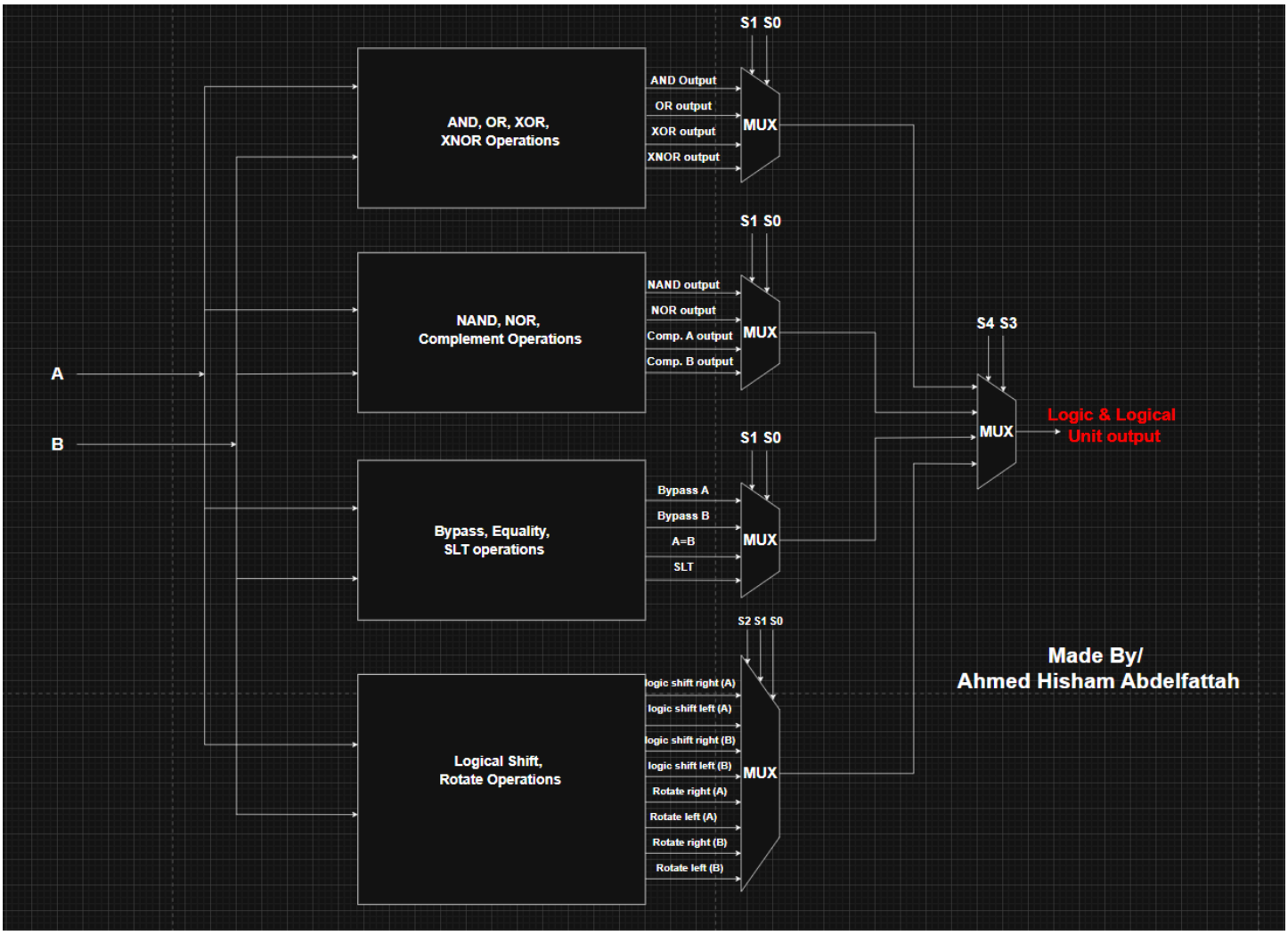
2.1 ALSU Design Overview:



2.2 Arithmetic& Arithmetical Shift Unit:



2.3 Logic& Logical Shift Unit:



3.ALSU Operations – Selectors Table:

3.1 Logic& Logical Shift Unit:

S5	S4	S3	S2	S1	S0	Function
1	0	0	0	0	0	AND
1	0	0	0	0	1	OR
1	0	0	0	1	0	XOR
1	0	0	0	1	1	XNOR
1	0	1	1	0	0	NAND
1	0	1	1	0	1	NOR
1	0	1	1	1	0	Complement (A)
1	0	1	1	1	1	Complement (B)
1	1	0	0	0	0	Bypass (A)
1	1	0	0	0	1	Bypass (B)
1	1	0	0	1	0	Equality (A = B)
1	1	0	0	1	1	SLT (A < B)
1	1	1	0	0	0	Logic Shift Right (A)
1	1	1	0	0	1	Logic Shift Left (A)
1	1	1	0	1	0	Logic Shift Right (B)
1	1	1	0	1	1	Logic Shift Left (B)
1	1	1	1	0	0	Rotate Right (A)
1	1	1	1	0	1	Rotate Left (A)
1	1	1	1	1	0	Rotate Right (B)
1	1	1	1	1	1	Rotate Left (B)

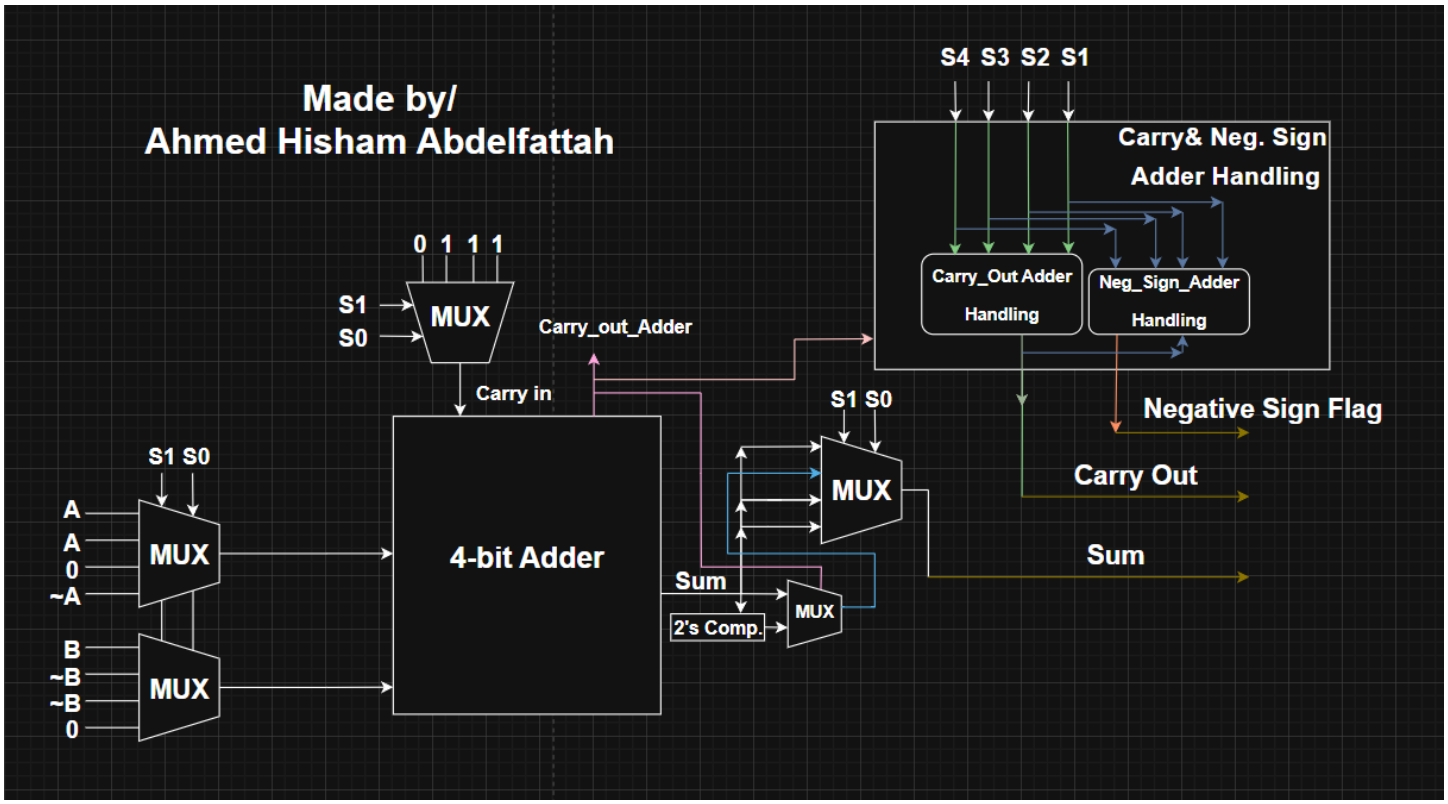
3.2 Arithmetic& Arithmetical Shift Unit:

S5	S4	S3	S2	S1	S0	**Function**
0	0	0	0	0	0	A + B
0	0	0	0	0	1	A - B
0	0	0	0	1	0	2's Comp. (B)
0	0	0	0	1	1	2's Comp. (A)
0	0	0	1	0	0	A × B (least significant 4 bits)
0	0	0	1	0	1	A × B (most significant 4 bits)
0	0	1	1	0	0	A / B (Quotient)
0	0	1	1	0	1	A / B (Remainder)
0	1	0	1	0	0	Parity Checker (A)
0	1	0	1	0	1	Parity Checker (B)
0	0	1	0	1	0	Arithmetic Shift Right (A)
0	0	1	1	1	1	Arithmetic Shift Left (A)
0	1	1	0	1	1	Arithmetic Shift Right (B)
0	1	1	1	1	1	Arithmetic Shift Left (B)
0	1	0	0	0	0	Reverse (A)
0	1	0	0	0	1	Reverse (B)
0	1	1	1	0	0	Increment (A)
0	1	1	1	0	1	Increment (B)
0	1	1	0	0	0	Decrement (A)
0	1	1	0	0	1	Decrement (B)

4. Detailed Architecture Description for Arithmetic& Arithmetical Shift Unit:

4.1 Adder Top Module:

- **Architecture:**

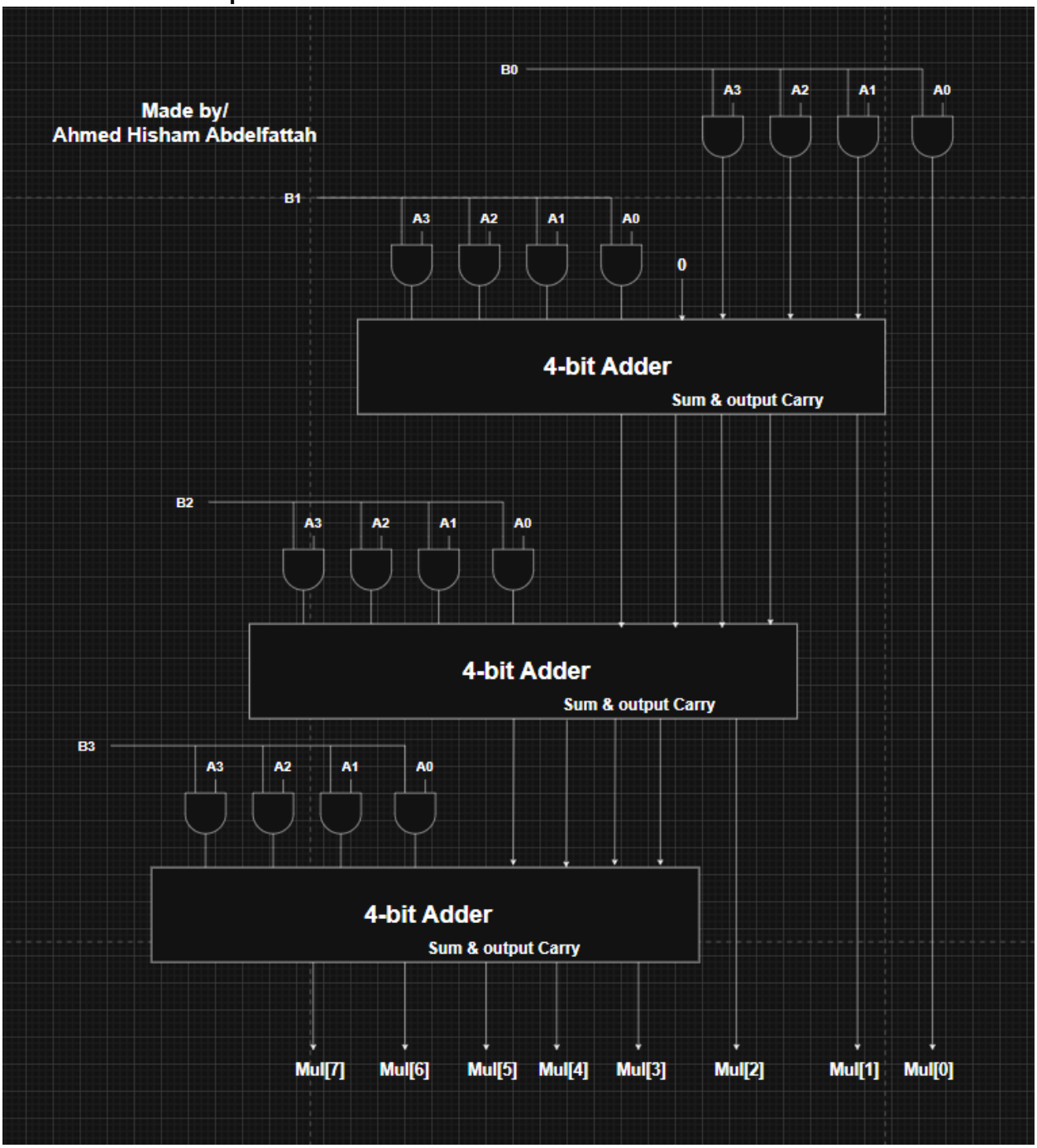


- This Module works according to the Selectors
  If Sel = 00 ➔ (A+B), Sel = 01 ➔ (A-B), Sel = 10 ➔ (2's Comp.(B)) , If Sel = 10 ➔ (2's Comp.(A))

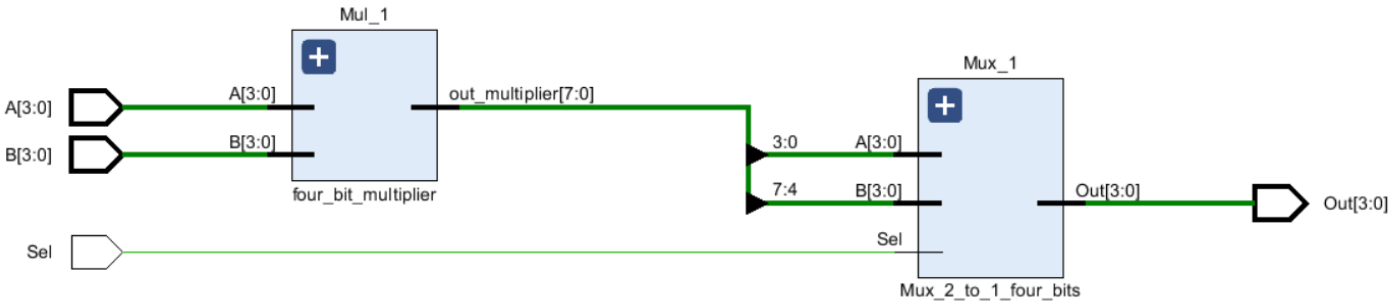
- Test Bench Simulation:

4.2 Multiplier Top Module:

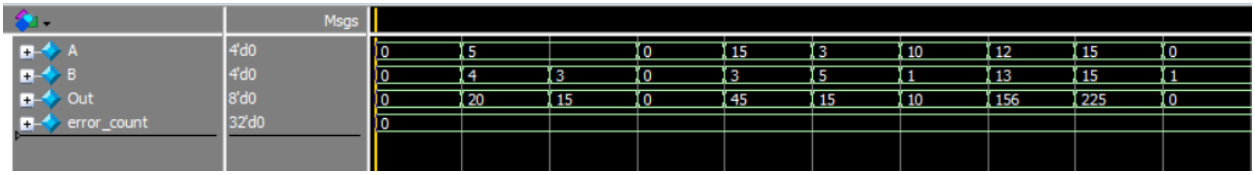
4.2.1 4-bit Multiplier Block:



4.2.2 Multiplier Top Module Implementation:

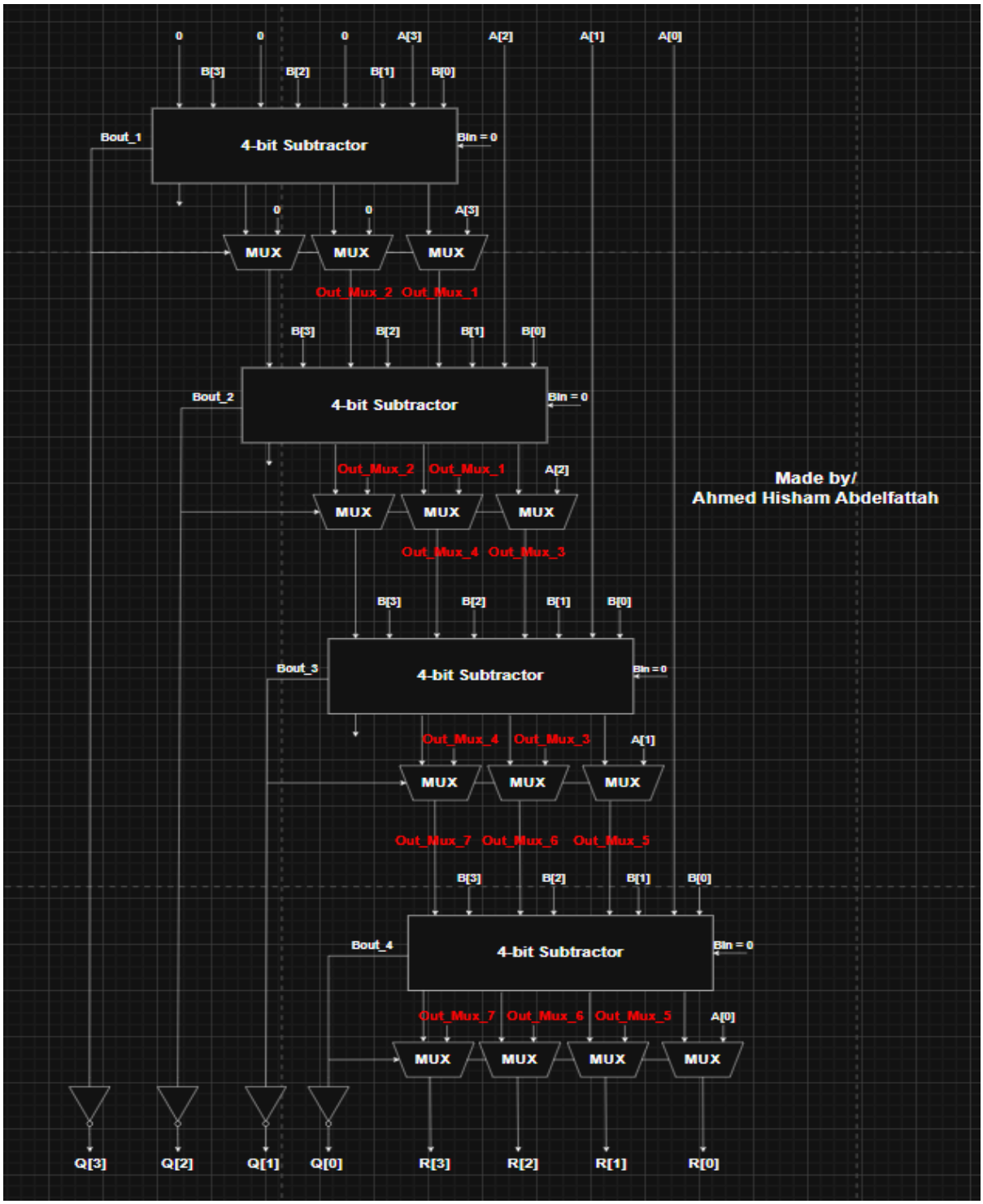


- This Module works according to the Selector:
  If Sel = 0 ➔ Out = Least Significant 4 bits and If Sel = 1 ➔ Out = Most Significant 4 bits.
- Test Bench Simulation:

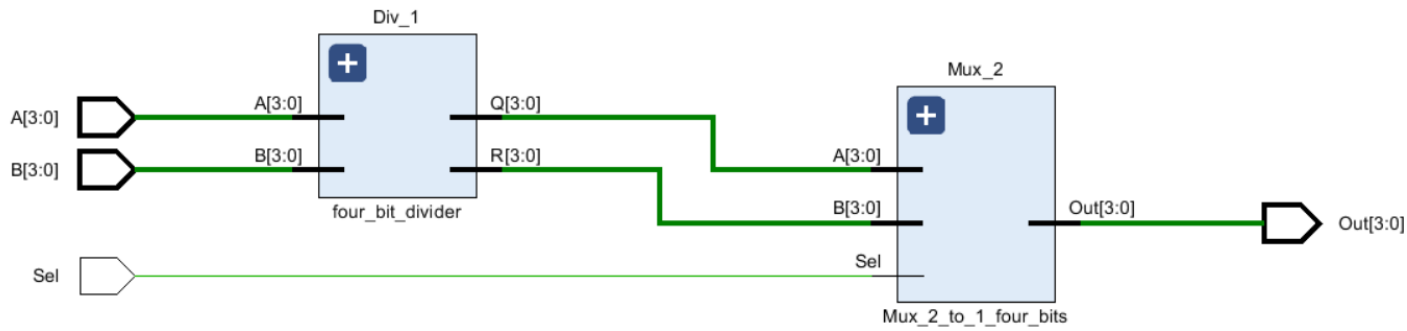


4.3 Divider Top Module:

4.3.1 4-bit Divider Architecture:

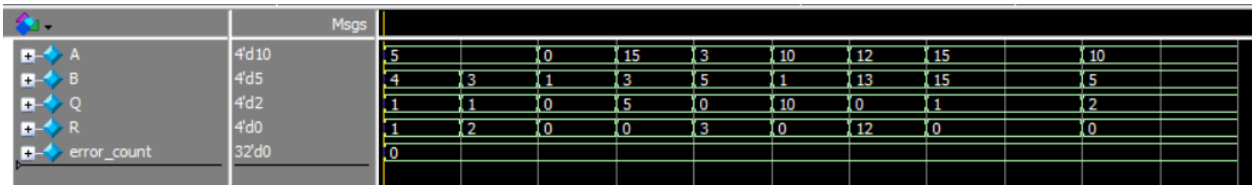


4.3.2 Divider Top Module Implementation:



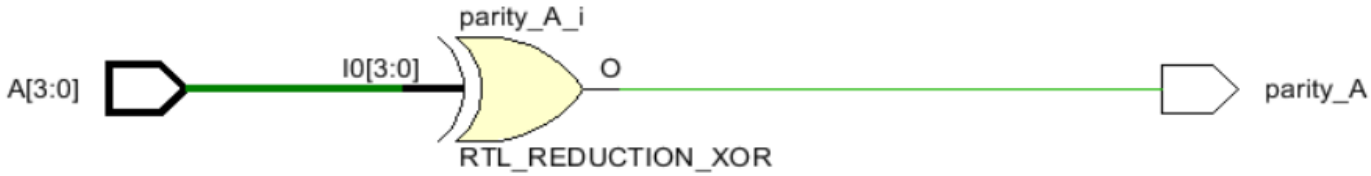
- When Sel = 0, the output will be the Quotient and when Sel = 1, the output will be the Remainder.

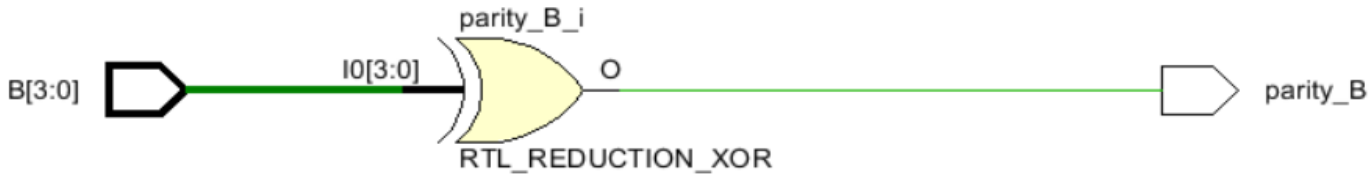
4.3.3 Test Bench Simulation:



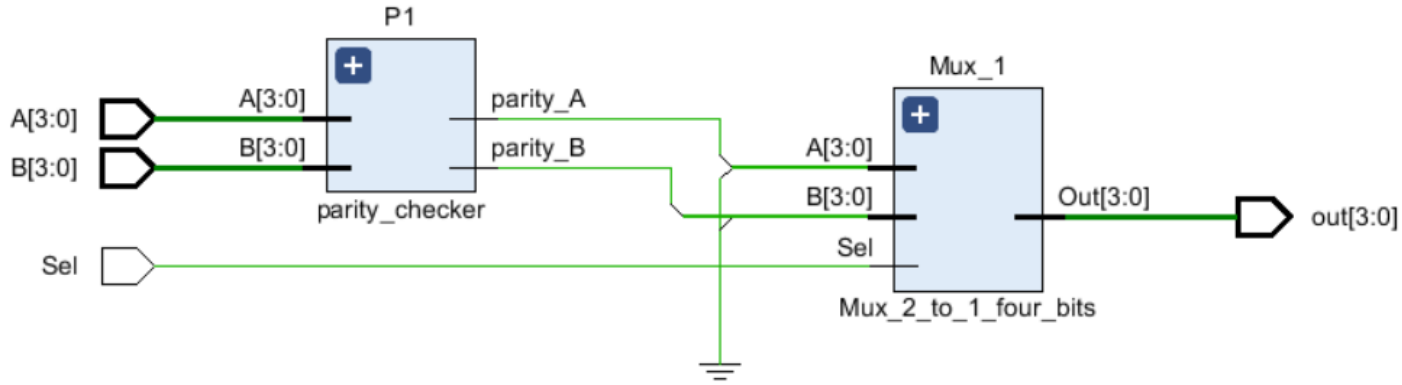
4.4 Parity Checker Top Module:

4.4.1 4-bit Parity Checker:



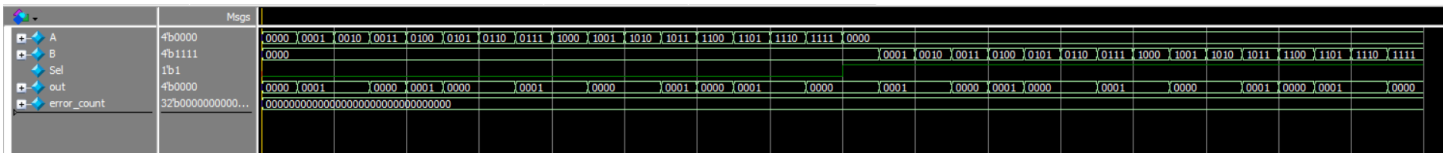


4.4.2 Parity Checker Top Module Implementation:



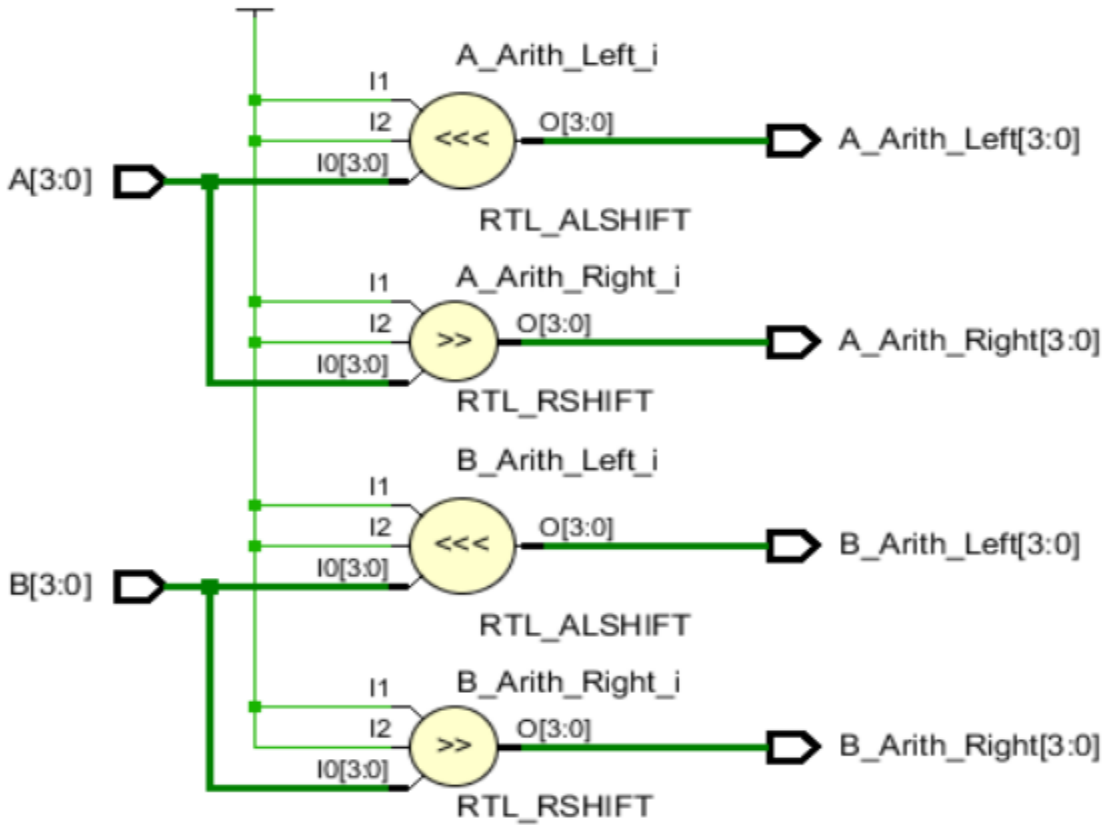
- This Module when Sel = 0, the output will equal Parity_Checker_A
  And when Sel = 1, the output will equal Parity_checker_B

4.4.3 Test Bench Simulation:

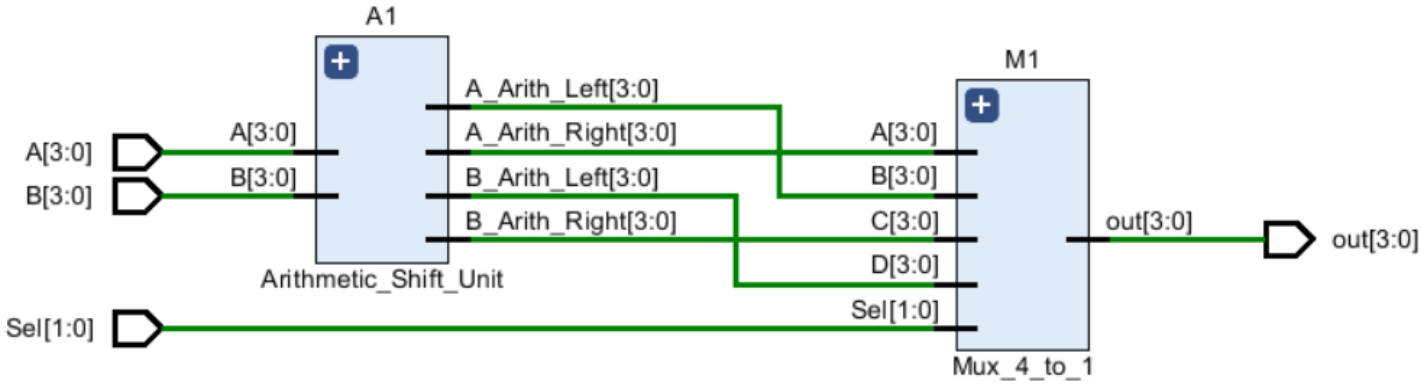


4.5 Arithmetic Shifter Top Module:

4.5.1 4-bit Arithmetic Shifter:



4.5.2 Arithmetic Shifter Top Module Implementation:

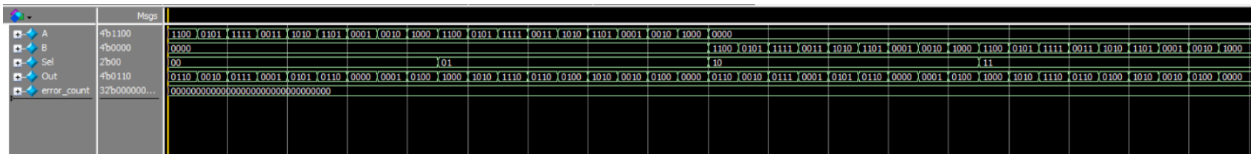


- This Module when Sel = 00, the output will equal Arithmetic Shift Right(A)
  , when Sel = 01, the output will equal Arithmetic Shift Left(A)
  , when Sel = 10, the output will equal Arithmetic Shift Right(B)
  And when Sel = 11, the output will equal Arithmetic Shift Left(A).
- **Notes about Arithmetic Shift:**

```
// ----------------------- Note -----------------------
// in Case of Signed Numbers: For Arithmetic Shift Right:
// Arithmetic right shift (>>>) replicates the sign bit
// (MSB = 1 for negative)
// (MSB = 0 for positive)

reg signed [3:0] A = 4'b0101;
// A = 4 + 1 = +5
result = A >>> 1;
A = 0101  →  >>> 1  →  0010

reg signed [3:0] A = 4'b1100;
// A = -8 + 4 = -4
result = A >>> 1;
A       = 1100   // -4
A >>> 1 = 1110   // -2

// ------------- what if ( >>> 2 )

reg signed [3:0] A = 4'b0101;
// A = 4 + 1 = +5
result = A >>> 2;
// Step 1 >>> 1 = 0010  (+2)
// Step 2 >>> 2 = 0001  (+1)
A >>> 2 = 4'b0001 // which is +1 in 4-bit signed

reg signed [3:0] A = 4'b1100;
// A = -8 + 4 = -4
result = A >>> 2;
// Step 1 >>> 1 = 1110  (-2)
// Step 2 >>> 2 = 1111  (-1)
A >>> 2 = 4'b1111 // which is -1 in 4-bit signed

// ----------------------- Note -----------------------
// in Case of Unsigned Numbers: For Arithmetic Shift Right:
// Arithmetic right shift (>>>) works as Logical Shift Right

reg [3:0] A = 4'b1100;

A       = 1100   // 12
A >>> 1 = 0110   // 6

A       = 1100   // 12
A >>> 2 = 0011   // 3

reg [3:0] A = 4'b0101;

A       = 0101  // 5
A >>> 1 = 0010  // 2

A       = 0101  // 5
A >>> 2 = 0001  // 1
```

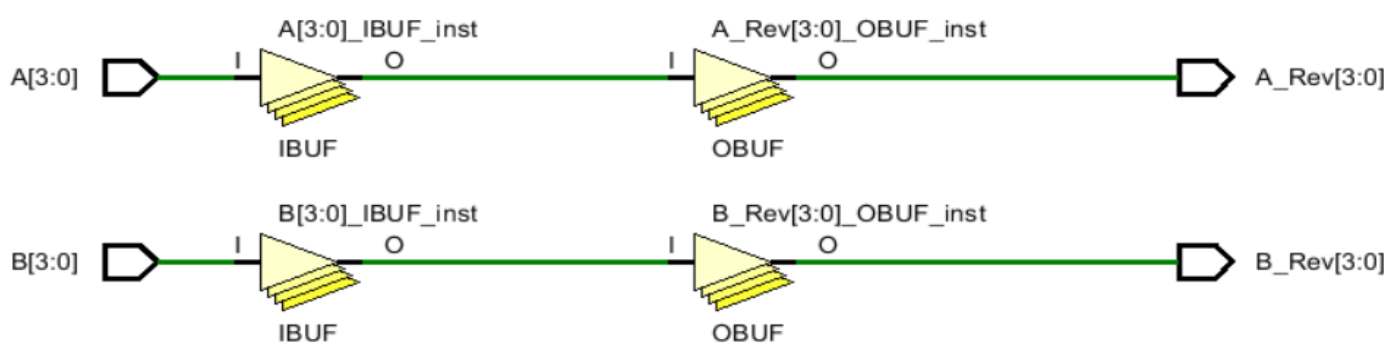
```
// ---------------------- Note ----------------------
// in Case of Unsigned & Signed Numbers: For Arithmetic Shift Left:
// Arithmetic Left Shift (<<<) behaves the same as Logical Left Shift (<<).
// This is true for both signed and unsigned types.

reg [3:0] A = 4'b0101 ;

A       = 0101  (5)
A <<< 1 = 1010  (10)

A       = 0101  (5)
A <<< 2 = 0100  (4)


reg signed [3:0] A = 4'b0101;

A        = 0101  (+5)
A <<< 1  = 0101 <<< 1 = 1010 = (-6) in 4-bit signed

A        = 0101  (+5)
A <<< 2  = 0101 <<< 2 = 0100 = (+4) in 4-bit signed
```

4.5.3 Test Bench Simulation:

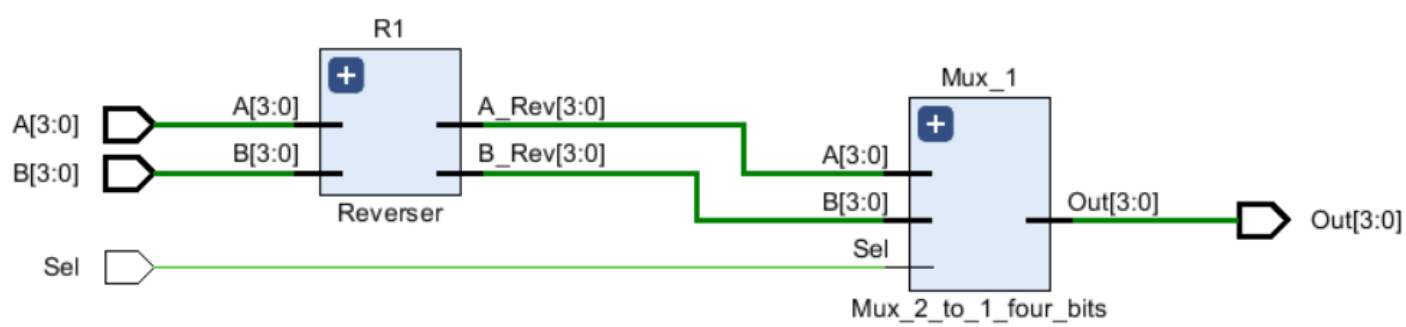


4.6 Reverser Top Module:

4.6.1 4-bit Reverser:

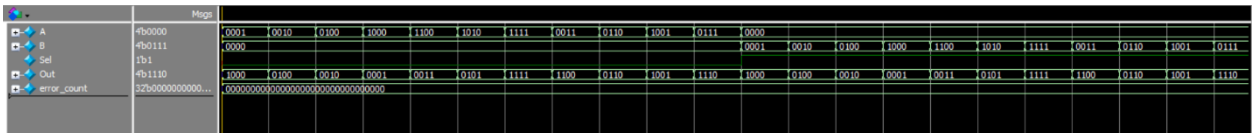


4.6.2 Reverser Top Module:



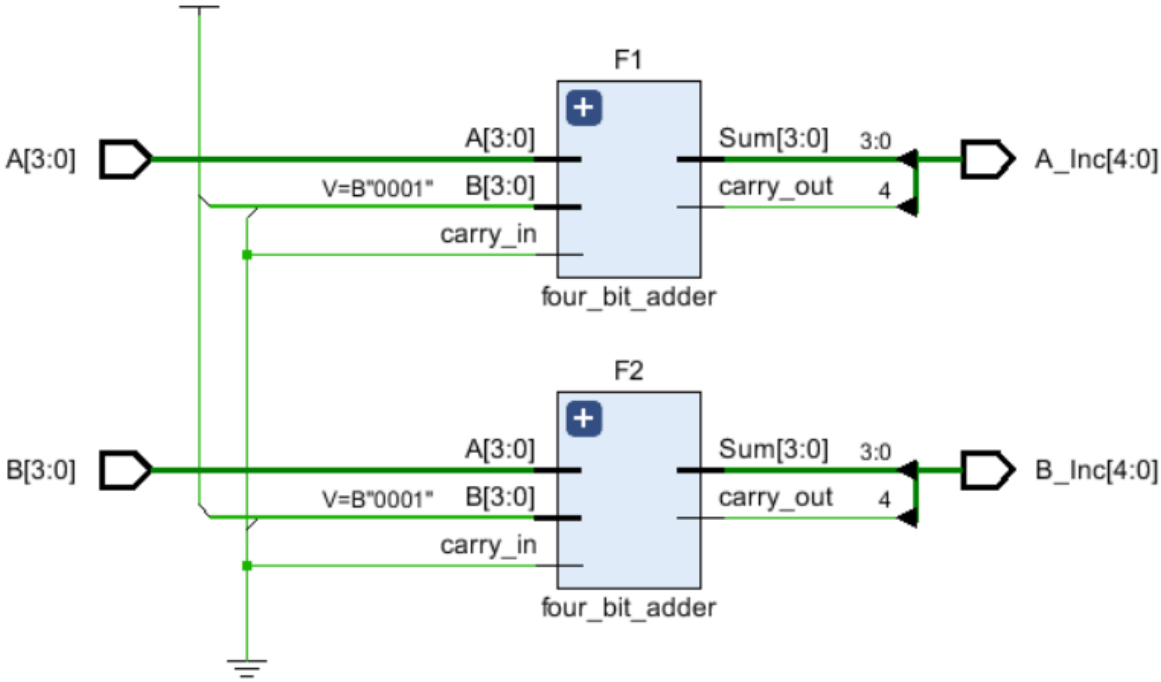
- This Module when Sel = 0, the output will equal Reverse_A
  And when Sel = 1, the output will equal Reverse_B.

4.6.3 Test Bench Simulation:

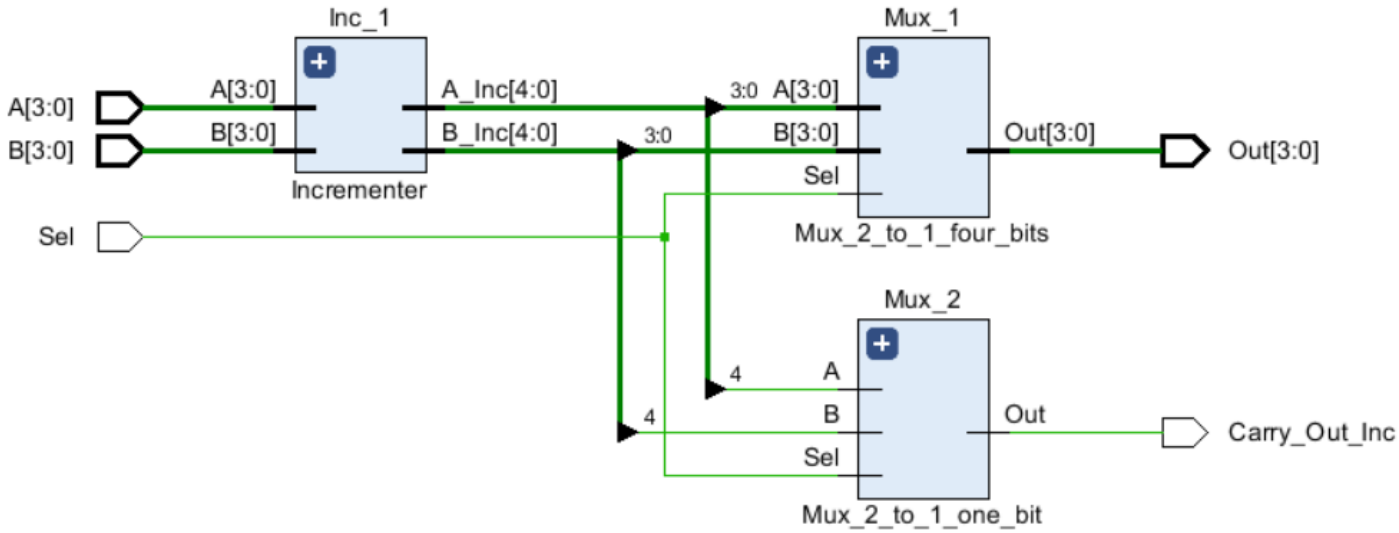


4.7 Incrementer Top Module:

4.7.1 4-bit Incrementer:

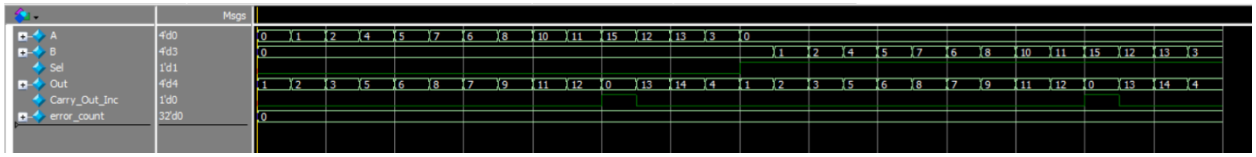


4.7.2 Incrementer Top Module:



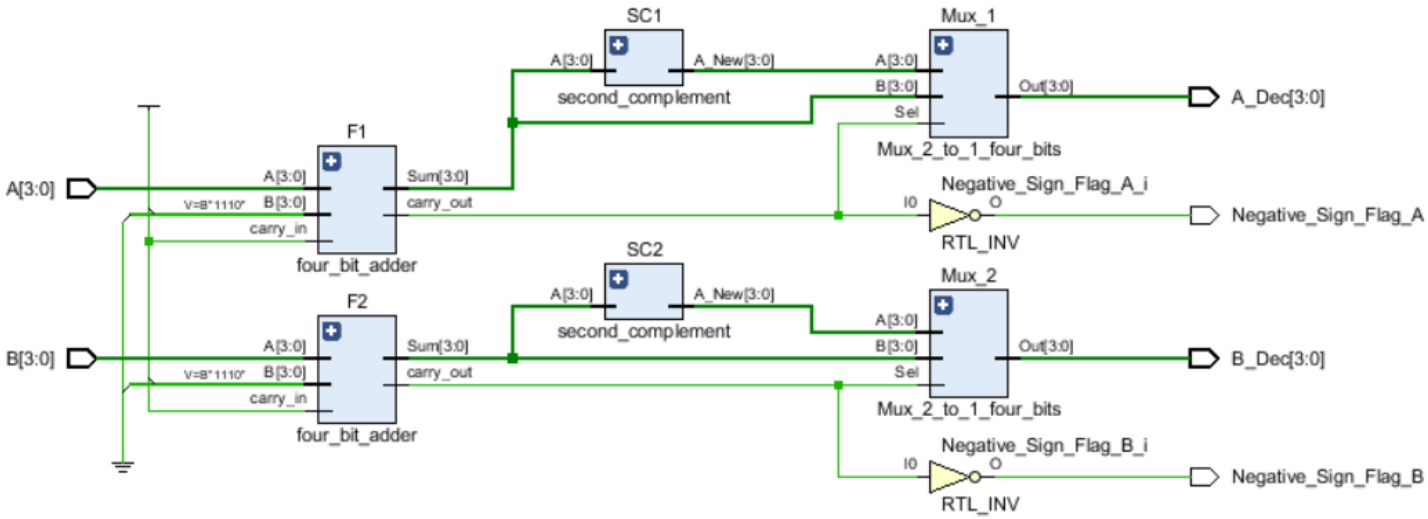
- This Module when Sel = 0, the output will equal A_Incremented and Carry_Out_A
  And when Sel = 1, the output will equal B_Incremented and Carry_Out_B.

4.7.3 Test Bench Simulation:

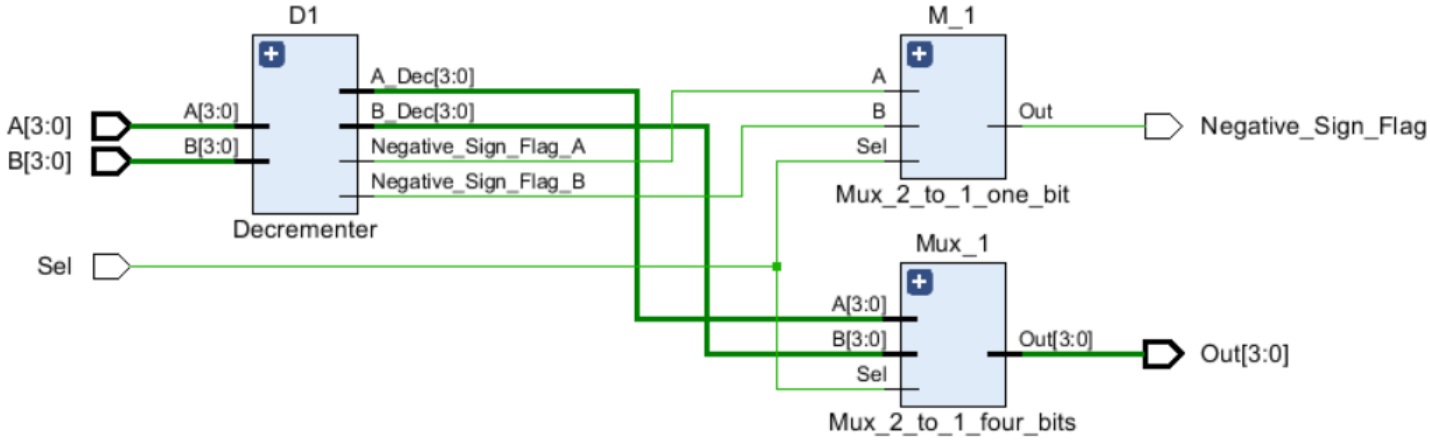


4.8 Decrementer Top Module:

4.8.1 4-bit Decrementer:

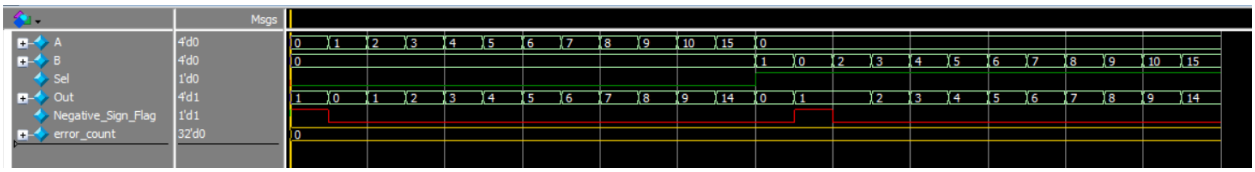


4.8.2 Decrementer Top Module:



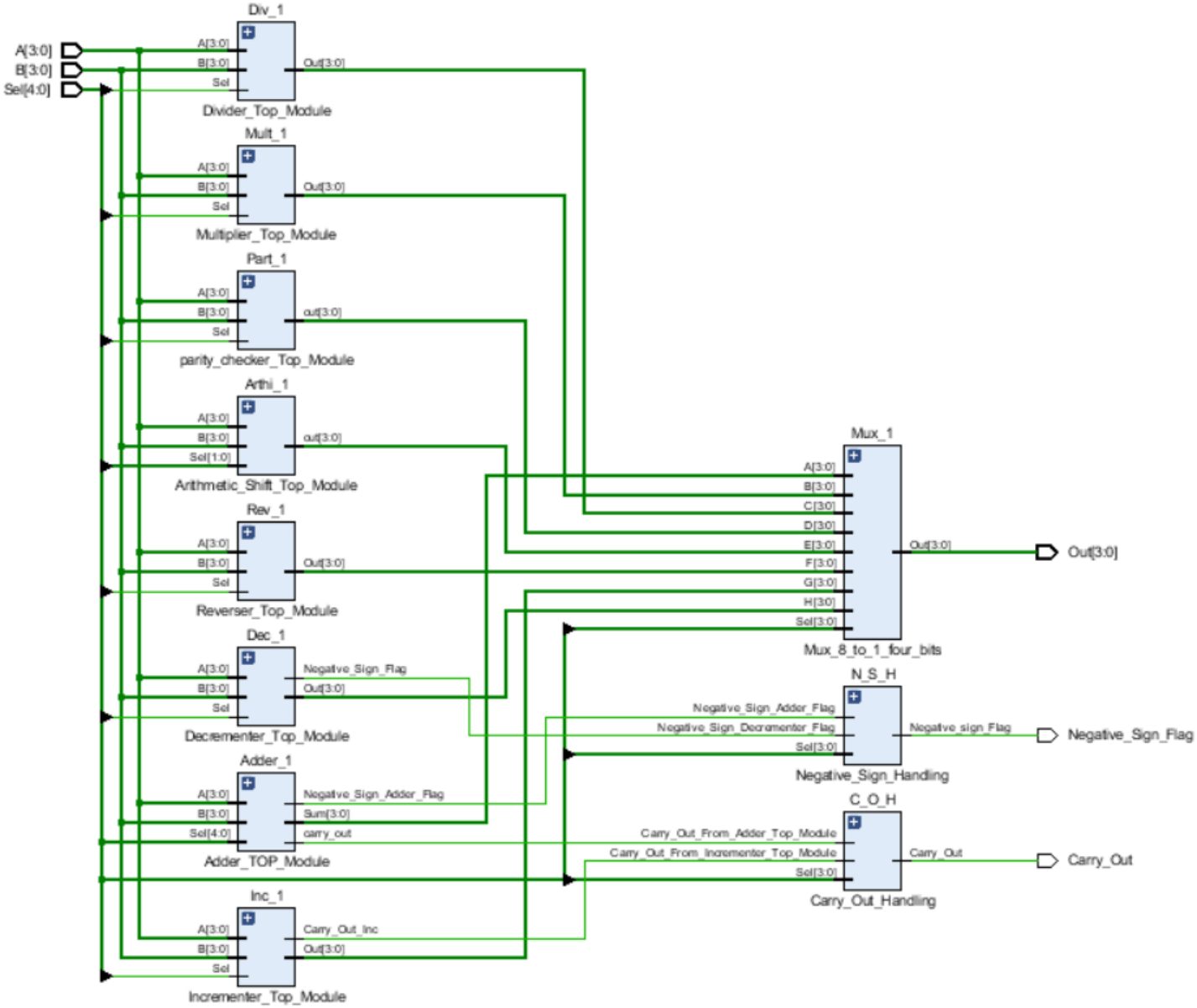
- This Module when Sel = 0, the output will A_Decremented and Negative_Sign_Flag_For_A
  And when Sel = 1, the output will equal B_Decremented and Negative_Sign_Flag_For_B.

4.8.3 Test Bench Simulation:



4.9 Arithmetic& Arithmetical Shift Unit Top Module:

- **Schematic:**

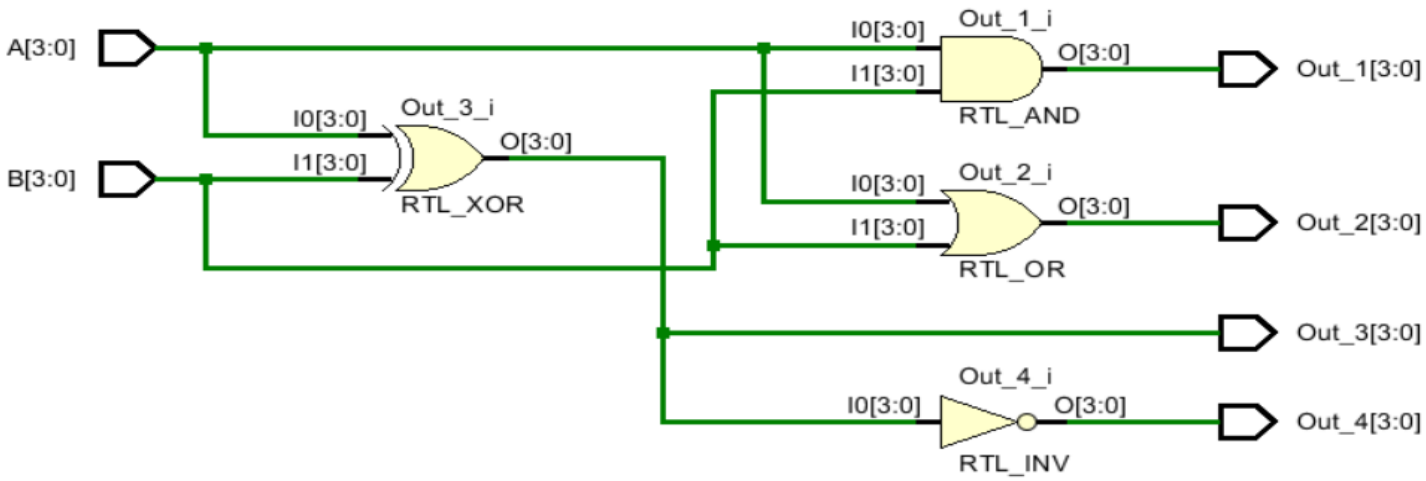


- This Module Works according to the Selectors table which is in the beginning of the report as you have five selectors here ➔ you will enter Sel[4:0] and Sel[5] isn't found here because it is constant for all Arithmetic& Arithmetical Shift operations, so we use it to select between Arithmetic& Arithmetical Shift Unit and Logic& Logical Shift unit because Sel[5] for the first is 0 and for the second is 1 .

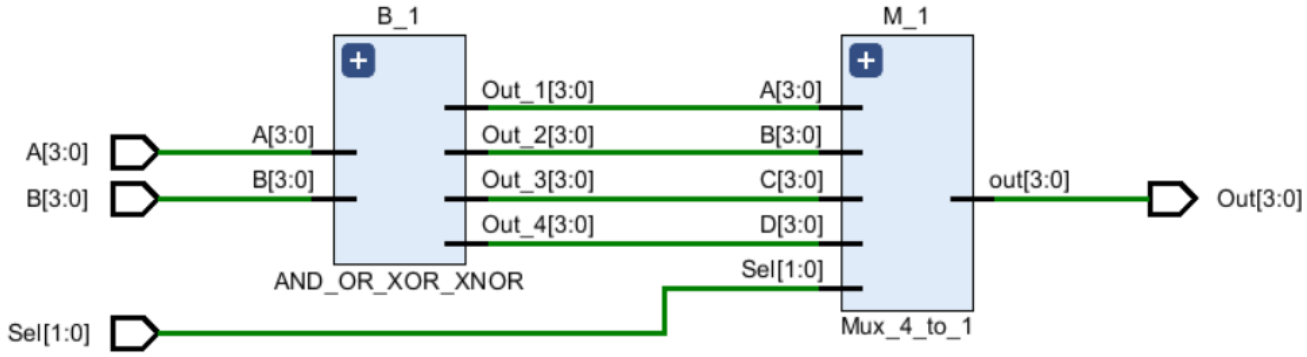
5.Detailed Design Architecture for Logic& Logical Shift Unit:

5.1 AND_OR_XOR_XNOR Operations Top Module:

5.1.1 AND_OR_XOR_XNOR Operations Block:

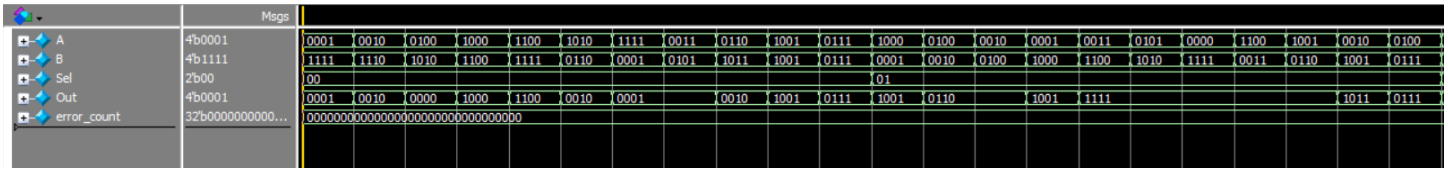


5.1.2 AND_OR_XOR_XNOR Operations Block:

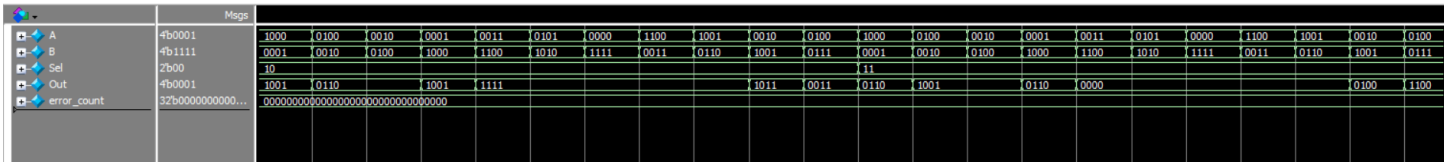


- This Module When Sel = 00 ➔ Output = A & B, and When Sel = 01 ➔Output = A | B And When Sel =10 ➔ Output = A ^ B, and when Sel =11 ➔ Output = ~(A^B)

- **Test Bench Simulation:**
  - When Sel = 00 and 01

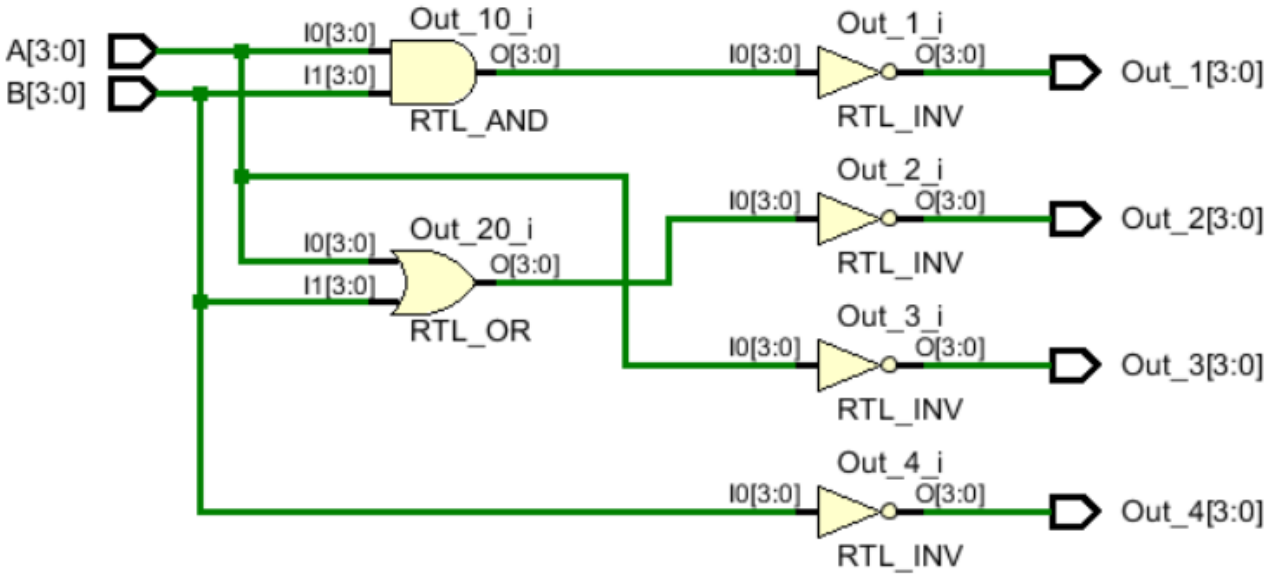


  - When Sel = 10 and 11

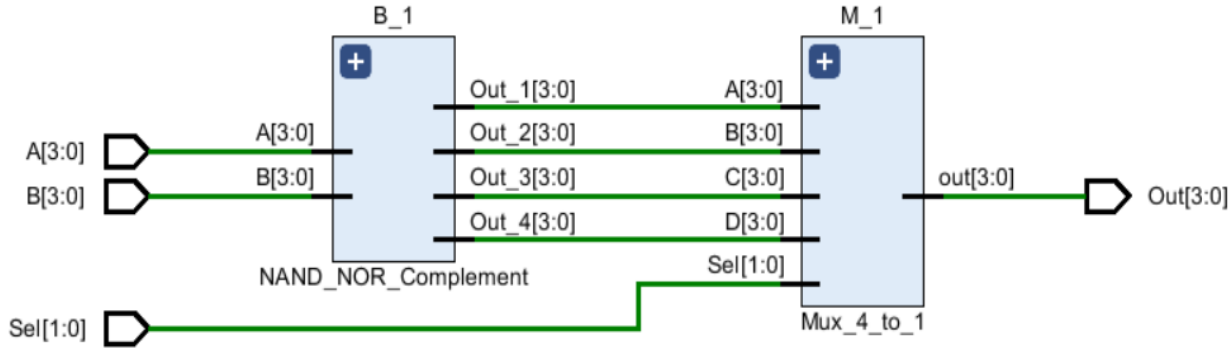


5.2 NAND_NOR_Complement Operations Top Module:

5.2.1 NAND_NOR_Complement Operations Block:

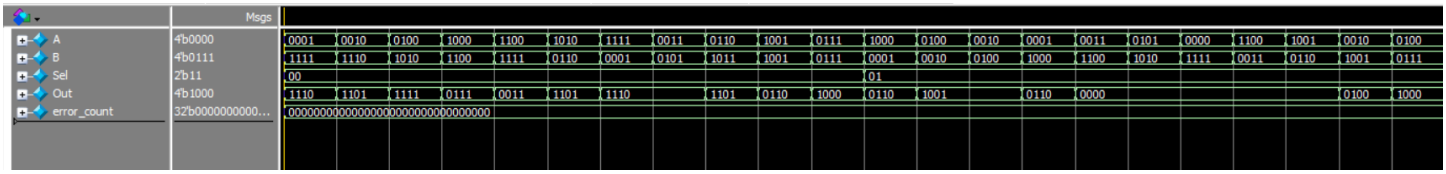


5.2.2 NAND_NOR_Complement Operations Top Module Block:

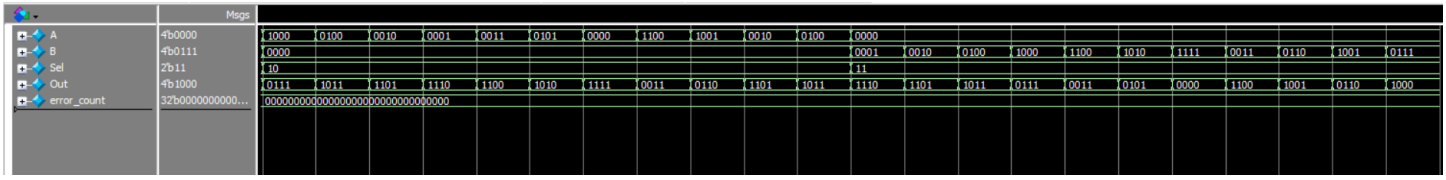


- This Module When Sel = 00 ➔ Output = ~(A & B), and When Sel = 01 ➔Output = ~(A | B) And When Sel =10 ➔ Output = ~(A), and when Sel =11 ➔ Output = ~(B)

- Test Bench Simulation:
  - When Sel = 00 and 01

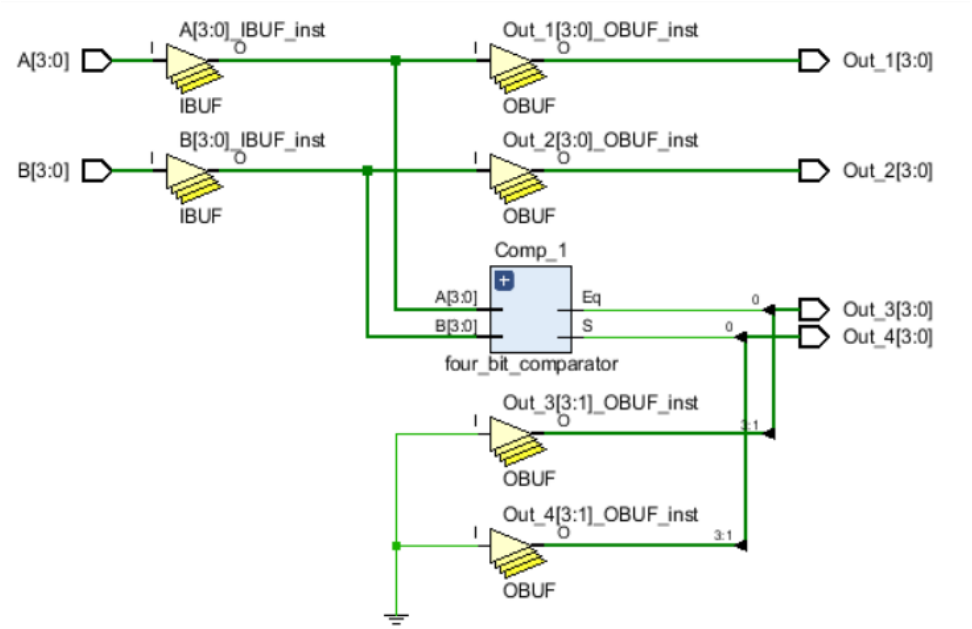


  - When Sel = 10 and 11

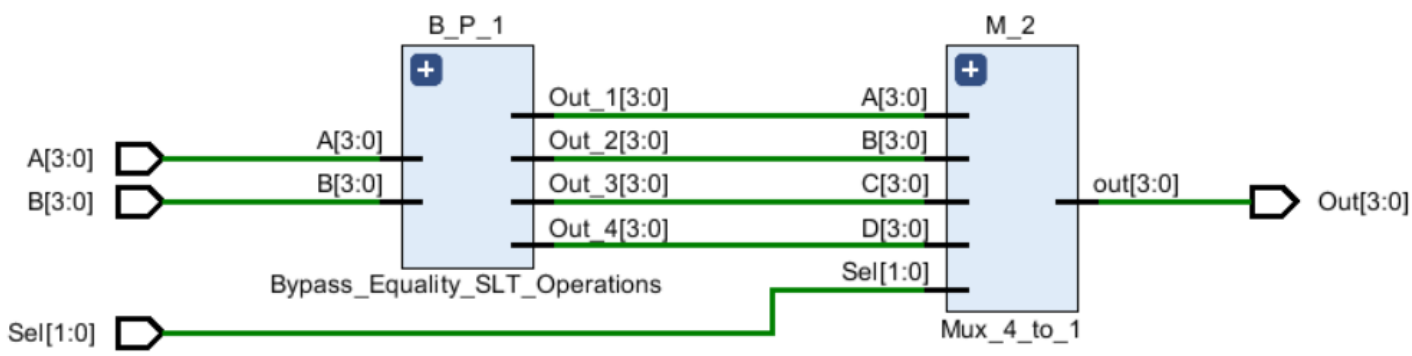


5.3 Bypass, Equality and SLT OperationsTop Module:

4.3.1 Bypass, Equality and SLT Operations Block:

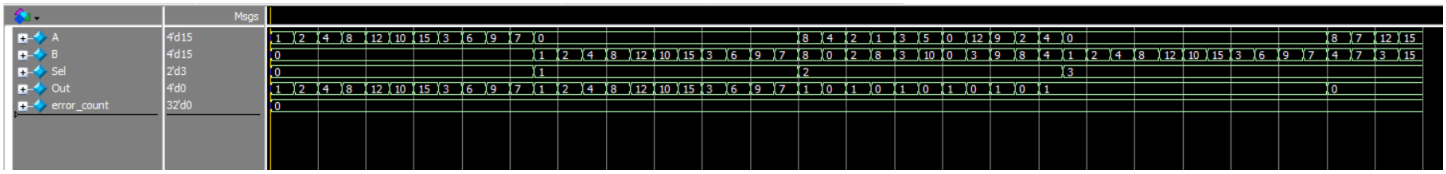


4.3.2 Bypass, Equality and SLT Operations Top Module Block:



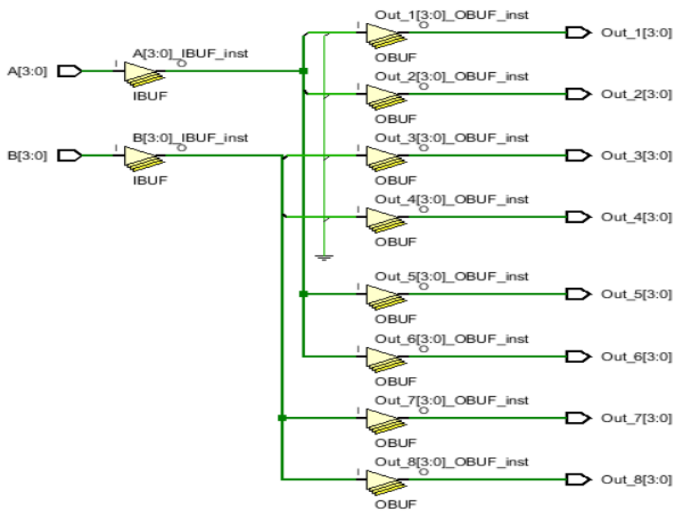
- This Module When Sel = 00, Output ➔ A
  , When Sel = 01, Output ➔ B
  , When Sel = 10, Output ➔ 1 if (A==B) and 0 otherwise.
  And when Sel = 11, Output ➔ 1 if (A<B) and 0 otherwise.

- Test Bench Simulation:

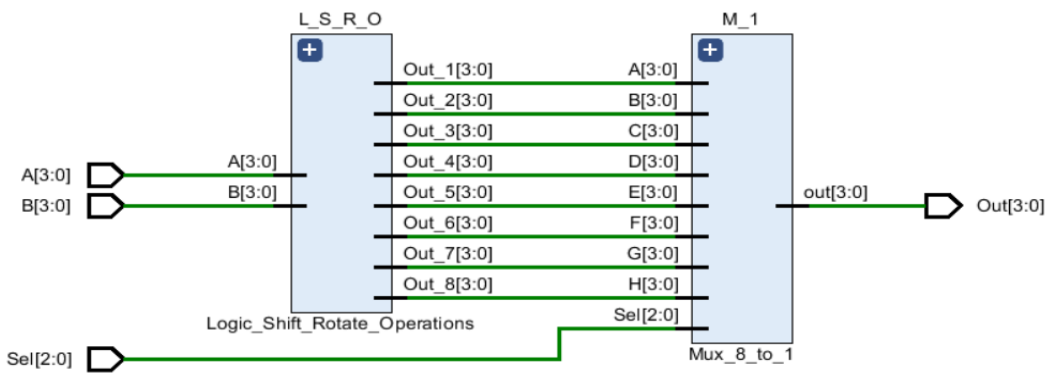


5.4 Logical Shift & Rotate Operations Top Module:

5.4.1 Logical Shift & Rotate Operations Block:



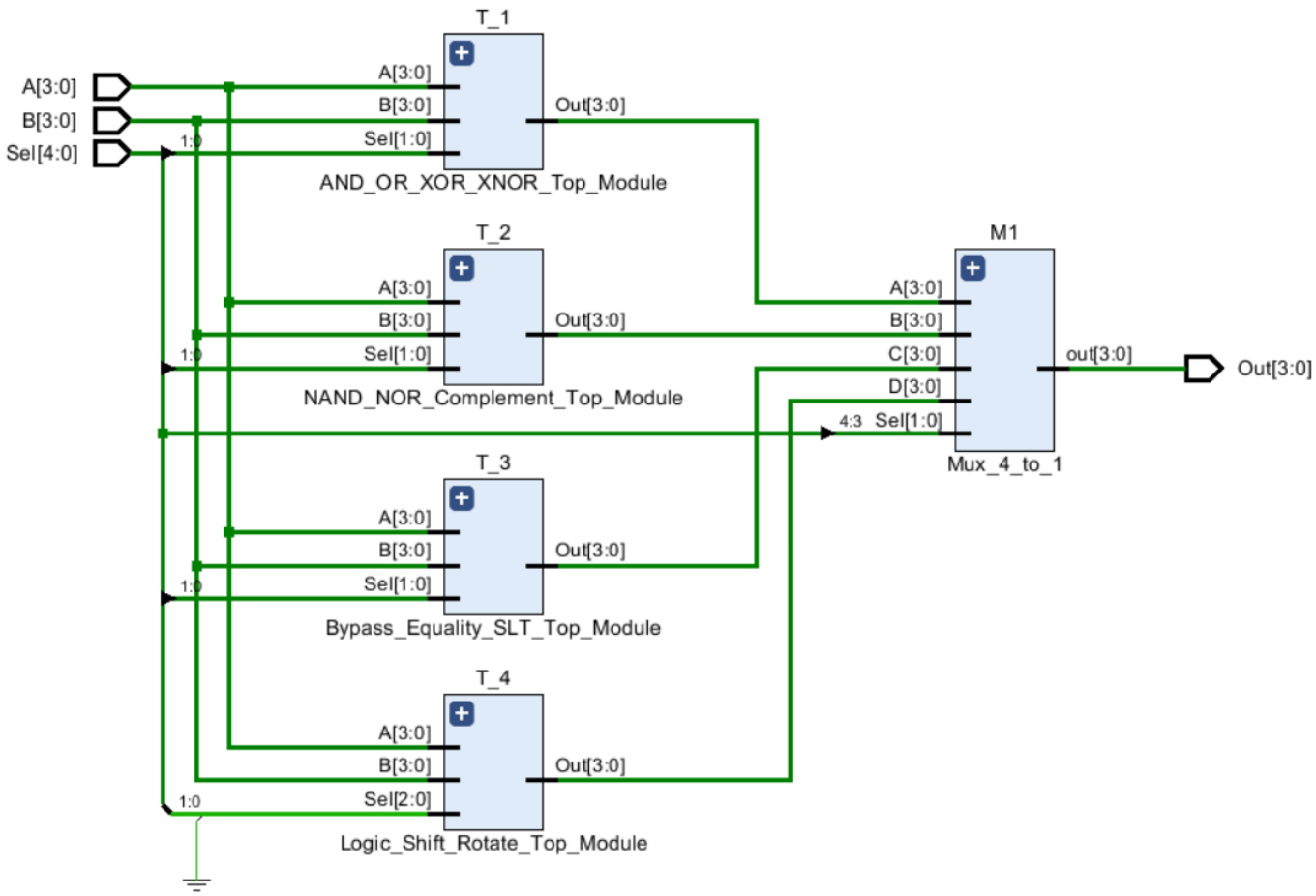
5.4.2 Logical Shift & Rotate Operations Top Module:



- This Module When Sel = 000 ➔ logic shift right(A), when Sel = 001 ➔logic shift left(A), when Sel = 010 ➔logic shift right(B), when Sel= 011 ➔logic shift left(A), when Sel = 100 ➔Rotate Right(A), when Sel =101 ➔Rotate Left(A), when Sel =110 ➔Rotate Right(B), when Sel = 111➔Rotate Left(B).
- Test Bench Simulation:

5.5 Logic & Logical Shift Unit Top Module:

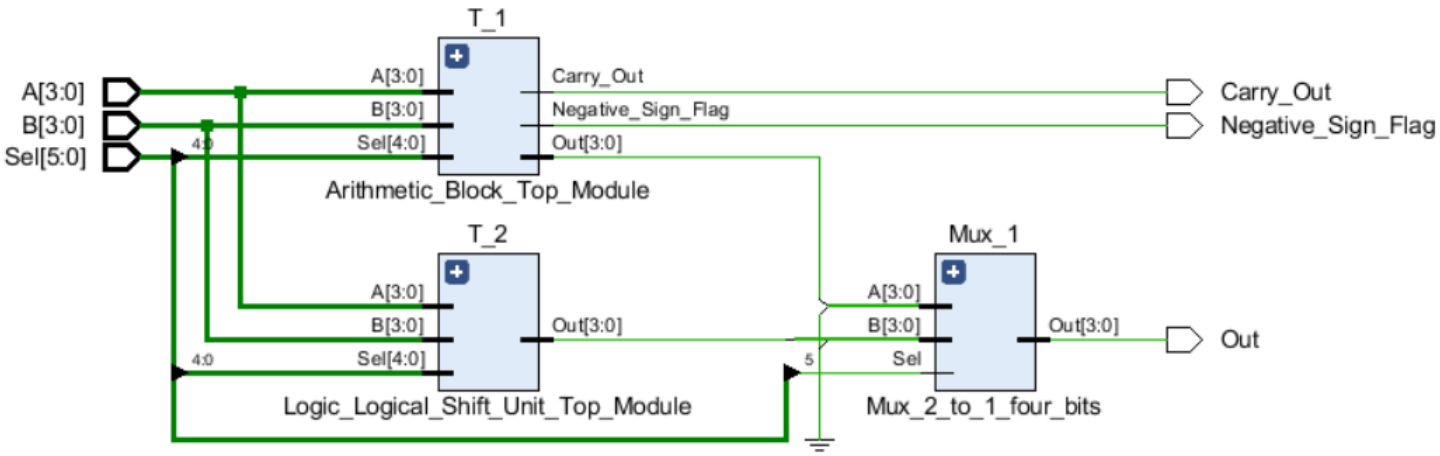
- Schematic:



- This Module Works according to the Selectors table which is in the beginning of the report as you have five selectors here ➔ you will enter Sel[4:0] and Sel[5] isn't found here because it is constant for all Logic& Logical Shift operations, so we use it to select between Arithmetic& Arithmetical Shift Unit and Logic& Logical Shift unit because Sel[5] for the first is 0 and for the second is 1 .

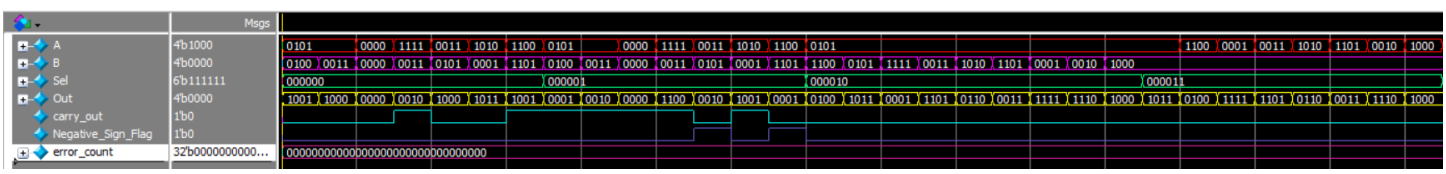
6. Simulation & Testing for the whole System:

- **Schematic:**

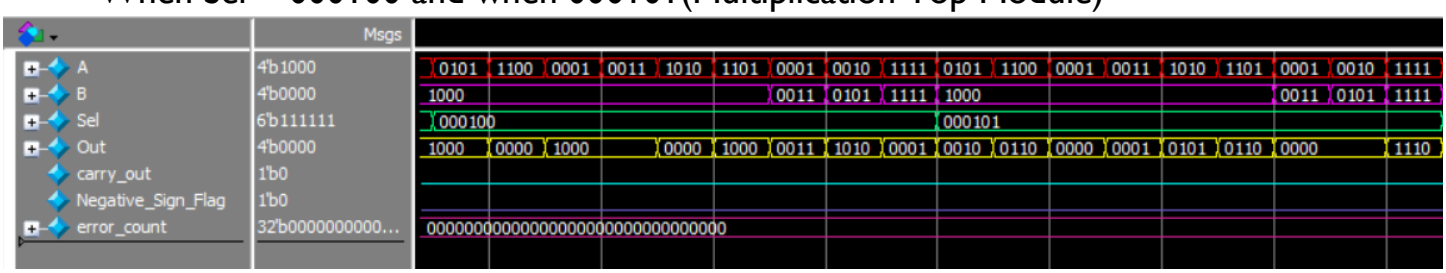


Waveform Snippets for the whole System:

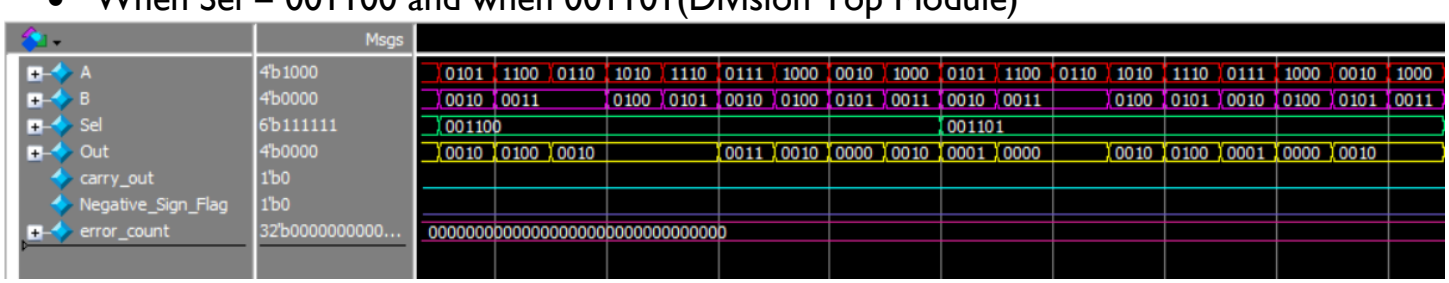
- When Sel = 000000 and when 000001 and when 000010 and when 000011(Adder Top Module)



- When Sel = 000100 and when 000101(Multiplication Top Module)



- When Sel = 001100 and when 001101(Division Top Module)

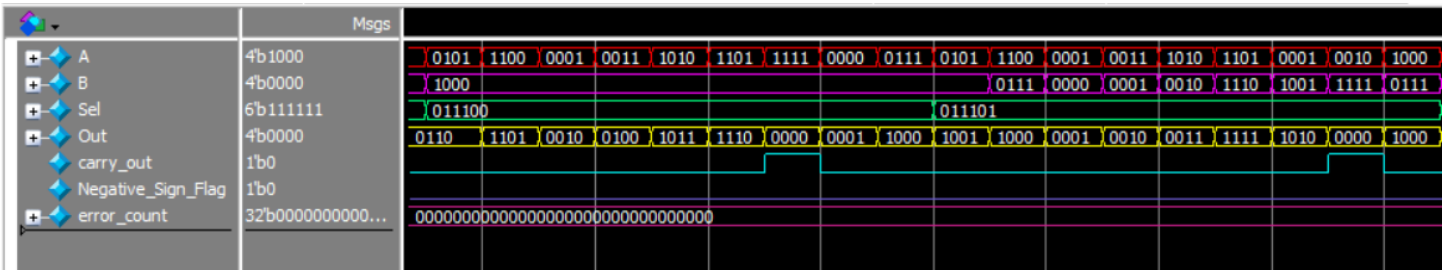


- When Sel = 010100 and when 010101(Parity Checker Top Module)

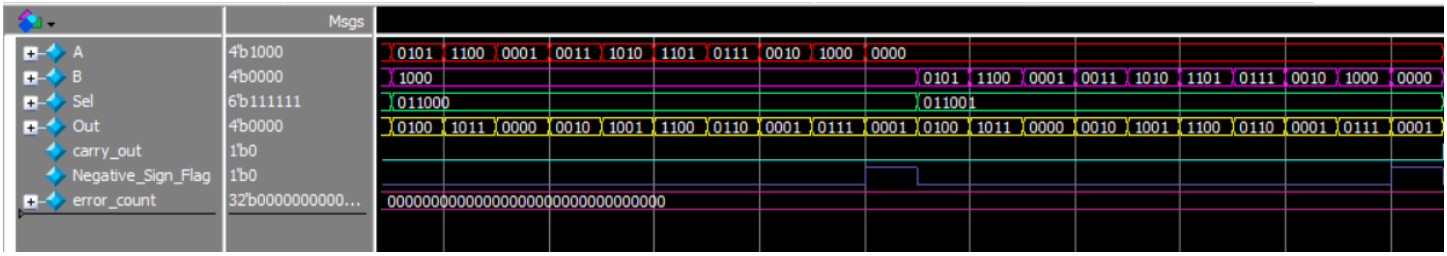
- When Sel = 001010 and when 001111 and when 011011 and when 011111(Arithmetic Shifter Top Module)

- When Sel = 010000 and when 010001(Reverser Top Module)

- When Sel = 011100 and when 011101(Incrementer Top Module)



- When Sel = 011000 and when 011001(Decrementer Top Module)

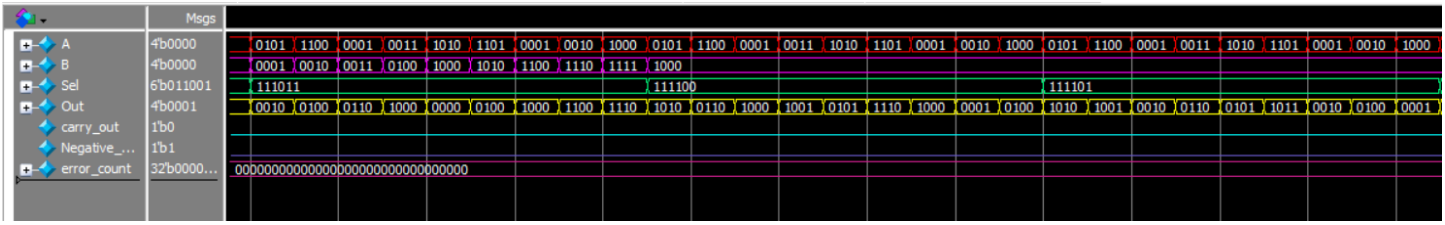


- When Sel = 100000 and when 100001 and when 100010 and when 100011(AND_OR_XOR_XNOR operations Block Top Module)

- When Sel = 101100 and when 101101 and when 101110 and when 101111(NAND_NOR_Complement operations Block Top Module)

- When Sel = 110000 and when 110001 and when 110010 and when 110011(Bypass_Equality_SLT operations Block Top Module)

- When Sel = 111000 and when 111001 and when 111010 and when 111011 and when 111100 and when 111101 and when 111110 and when 111111(Logic Shift and Rotate operations Block Top Module)


Msgs
A 4'b0000
B 4'b0000
Sel 6'b011001
Out 4'b0001
carry_out 1'b0
Negative_... 1'b1
error_count 32'b0000...


Msgs
A 4'b0000
B 4'b0000
Sel 6'b011001
Out 4'b0001
carry_out 1'b0
Negative_... 1'b1
error_count 32'b0000...