ALU Project

CS 4341.001 Fall 2020

Cohort: Trohoc

Cohort Members: Emory Blair(eab170030), Khalid Dakak (knd170000), Anindita Palit(axp172130), Ryan Radloff(rjr180002), Rutvij Shah(rds190000)

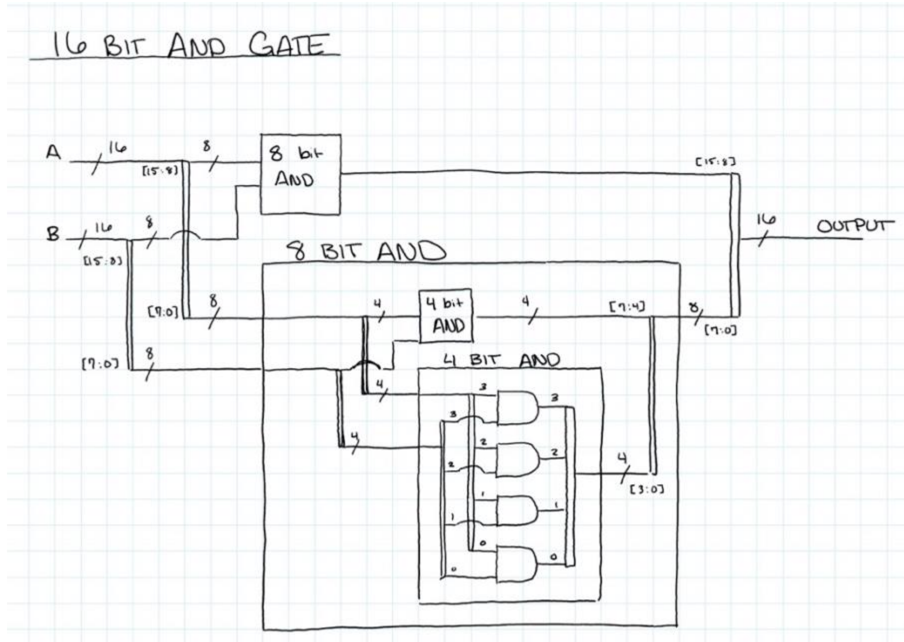
GitHub: https://github.com/0xrutvij/16bit-ALU-iverilog

List of Parts:

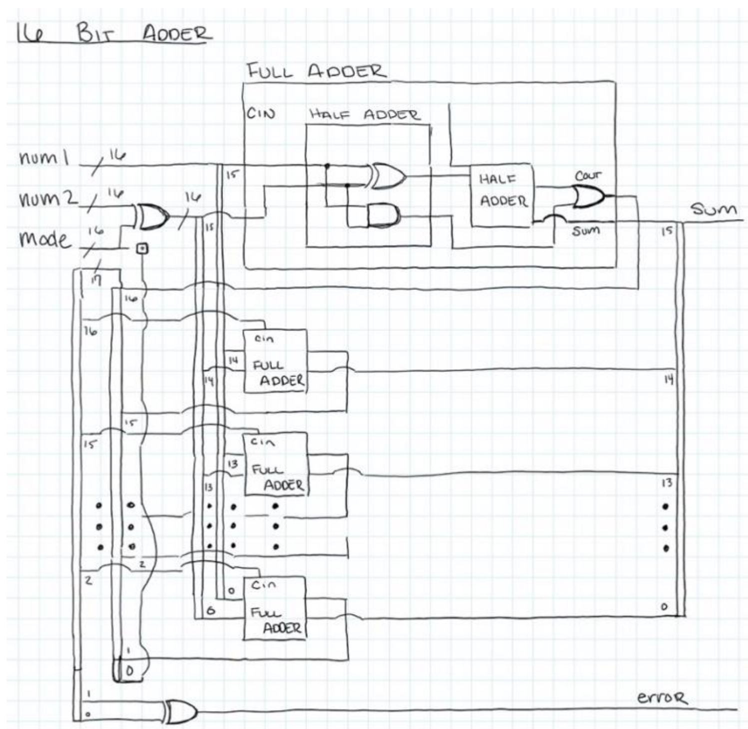
- **Inputs:**
  - Clock-1 bit.**[clk]** Feeds into the clock position on all registers in the system
  - Input A-16 bits.**[a_in]** Is the first binary term used for calculations in the ALU system. Unsigned 16-bit integer (max value 65535).
  - Input B-16 bits.**[b_in]** Is the second binary term used for calculations in the ALU system. Unsigned 16-bit integer (max value 65535).
  - Opcode-4 bits. **[opcode]** Binary code to indicate the operation used for calculations.
- **Outputs:**
  - Error output-1 bit. **[error]** Is true when an error state has occurred.
  - A-16 bits. **[a]** Output of the first binary term, for use in display of the system.
  - B-16 bits. **[b]** Output of the second binary term, for use in display of the system.
  - Final Output-16 bits.**[final_output]** The results of the calculations done by the ALU. Unsigned 16-bit integer (max value 65535).
- **Gates:**
  - AND gate-16 bits.**[and_test]** First channel is the value of A, second channel is the value of B. Output is the results of A AND B.
  - OR gate-16 bits.**[or_test]** First channel is the value of A, second channel is the value of B. Output is the results of A OR B.
  - XOR gate-16 bits.**[xor_test]** First channel is the value of A, second channel is the value of B. Output is the results of A XOR B.
  - NAND gate-16 bits.**[nand_test]** First channel is the value of A, second channel is the value of B. Output is the results of A NAND B.
  - NOR gate-16 bits.**[nor_test]** First channel is the value of A; second channel is the value of B. Output is the results or A NOR B.
  - XNOR gate-16 bits.**[xnor_test]** First channel is the value of A; second channel is the value of B. Output is the results of A XNOR B.
  - NOT gate-16 bits. **[not_t1]** Output is the results of A NOT.
  - NOR gate-1 bit. 16 channels, each channel is a bit from the value of B, to test for the possibility out dividing by zero. The output feeds into the error multiplexor.
  - OR gate-1 bit. 16 channels, each channel is a bit from the cout of the multiplier component to check for overflow in multiplication. The output feeds into the error multiplexor
  - AND gate-1 bit. 4 channels, each channel is a bit from the opcode input to test for the opcode “1111”. Upon receiving this opcode, the output of this and gate will send a signal to reset the system.

- **Combinational Logic Components:**
    - MUX16 **[mux_output_select]** Each channel has an output of the operation. This multiplexer chooses the correct output based on the opcode as select.
    - MUX16 **[error_state]** Each channel has an error state tied to an operation. This multiplexer outputs true to an error state based on the opcode as select.
    - Full-Adder**[adder_test]**, 16-bit, Full adder.
    - Multiplier**[mult_test]**, 16-bit Multiplier. (Implemented behaviorally)
    - Modulus**[mod_test]**, 16-bit Modulus. (Implemented behaviorally)
    - Divider **[div_test]**, 16-bit Divider. (Implemented behaviorally)
    - Shift Left**[sll_test]**, 16-bit bit shifter to the left.
    - Shift Right**[srl_test]**, 16-bit shifter to the right.
- **Internal Interfaces**
    - A- 16-bit wire
    - B- 16-bit wire
    - Final_output-16-bit wire. Carries operation result to output
    - Prev_output-16-bit wire. For use in the no-op state to cycle operation result back into the output multiplexor .
    - Mode-16 bit wire. For use with the full adder so that the subtraction operation can occur.
    - Out - 16-bit wire
    - Error_mat-16-bit wire
    - Errorline-16-bit wire
    - Select-4-bit wire. For use for operation selection within the systems multiplexors.
    - Reset-1-bit wire. For use to reset the system when the reset operation is called.
    - Never_reset-1-bit wire defaulted to 0.
- **Sequential Logic Components:**
    - A_Reg. 16-bit D register that contains the value of A. The first binary term.
    - B_Reg. 16-bit D register that contains the value of B. The second binary term.
    - Selector. 4-bit D register that contains the value of the opcode used for operation selection.
    - Pers_op. 16-bit D register that contains the value of the previous operation.
- **Modules:**
    - Test Bench- Main module. Used for testing of the components used in this system. (not shown)

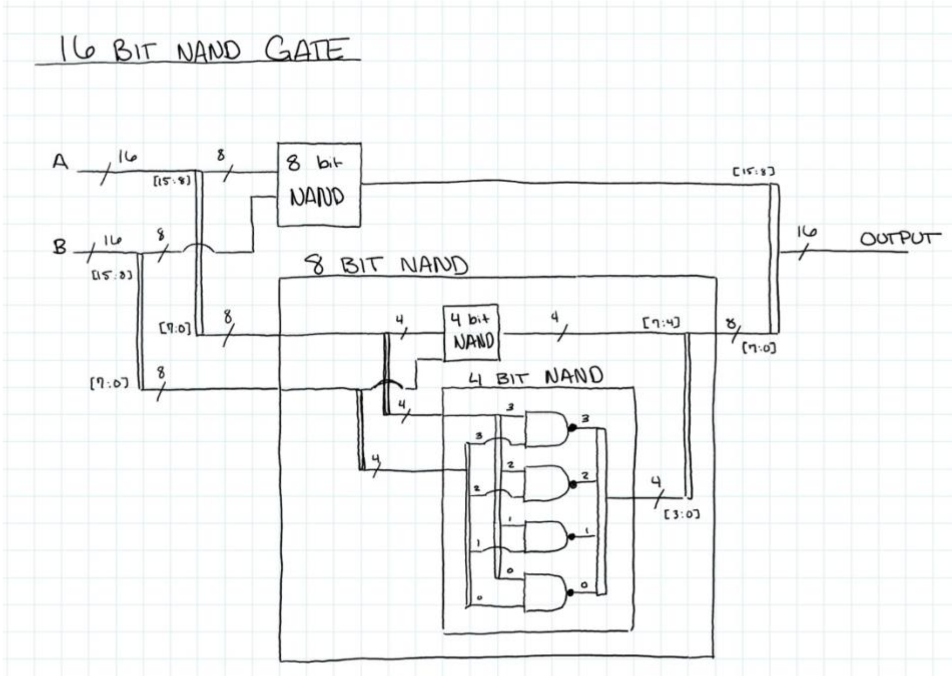
Circuit Diagrams:



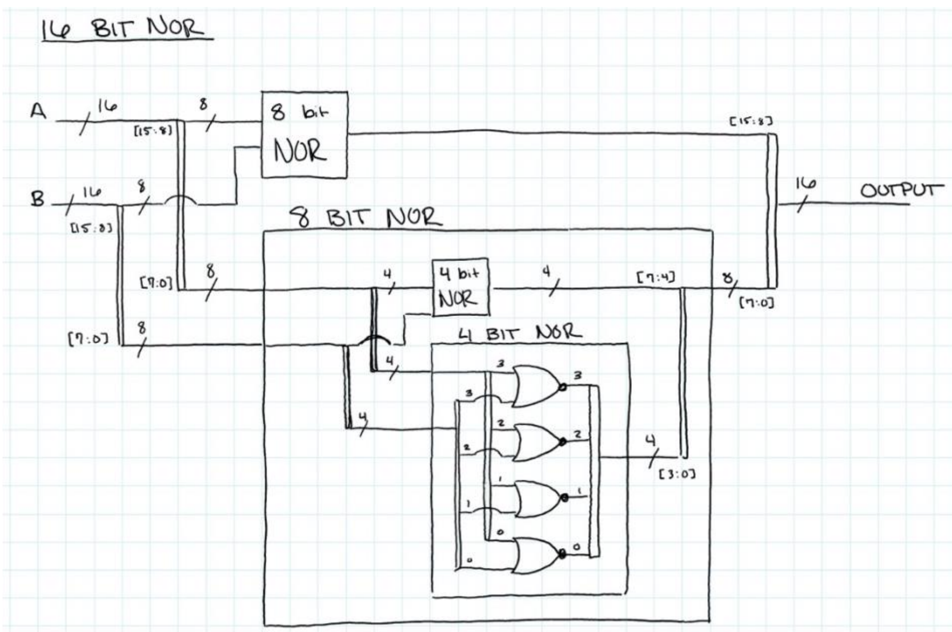
16 Bit AND Gate



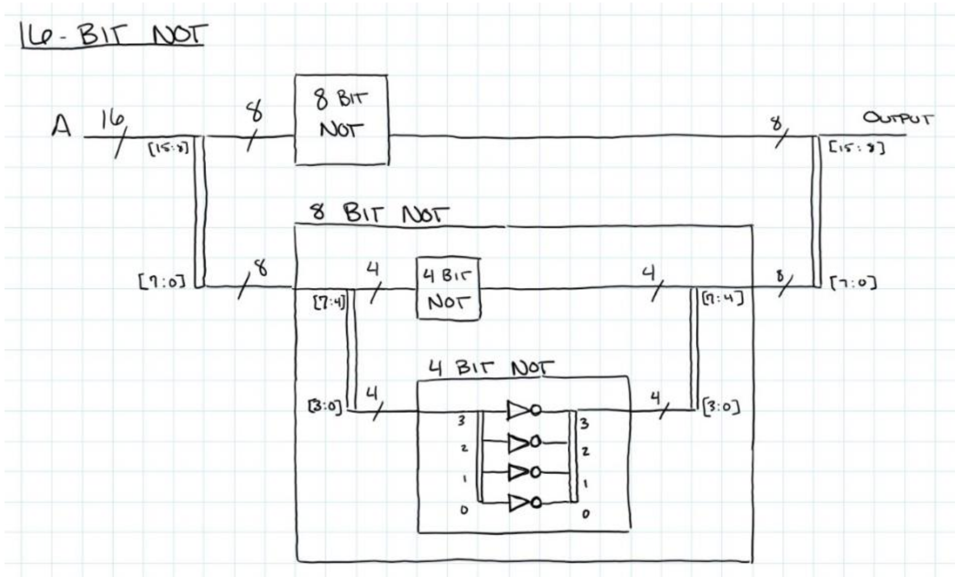
16 Bit Full Adder



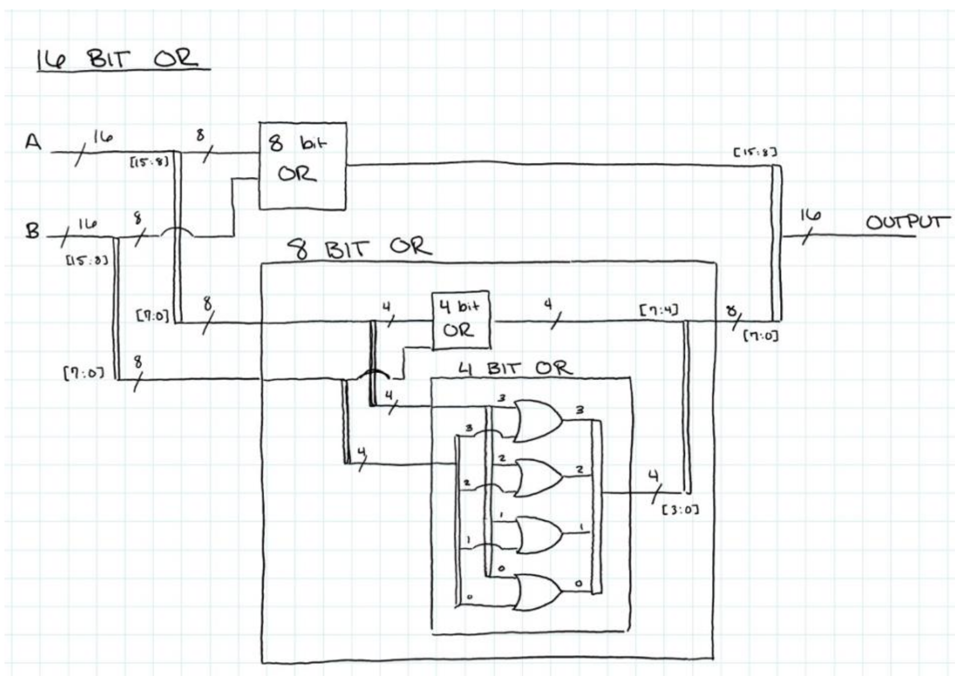
16 Bit NAND Gate



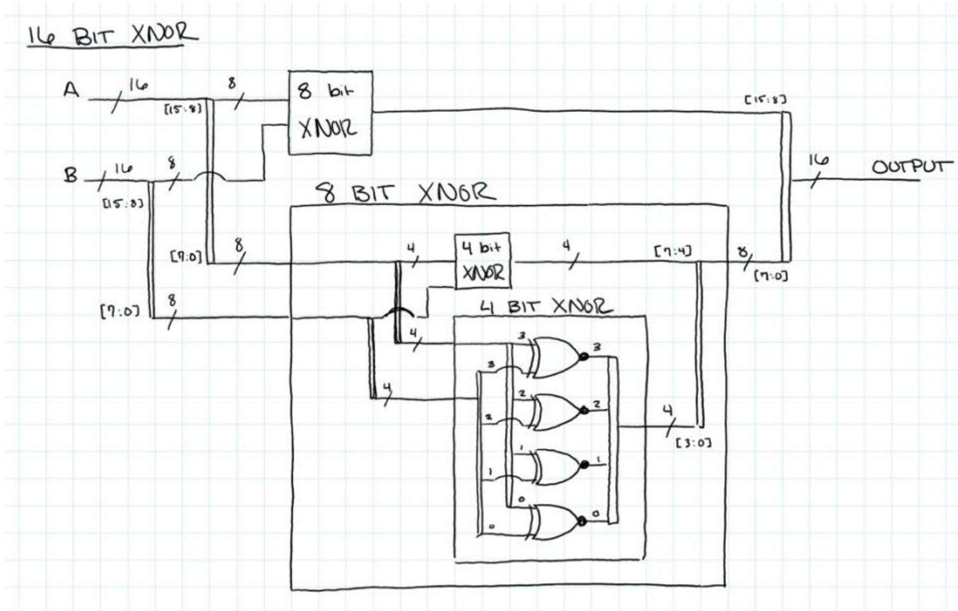
16 Bit NOR Gate



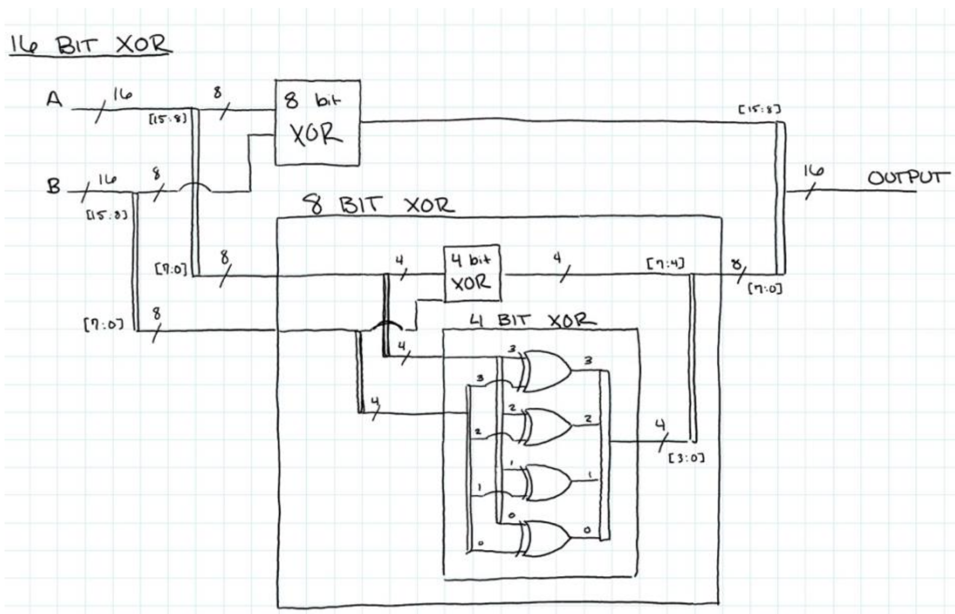
16 Bit NOT Gate



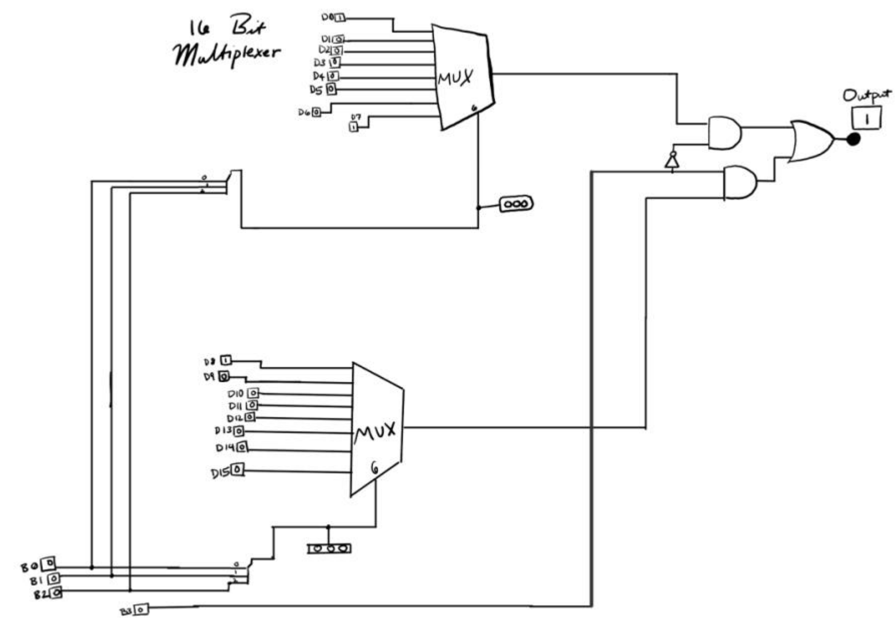
16 Bit OR Gate



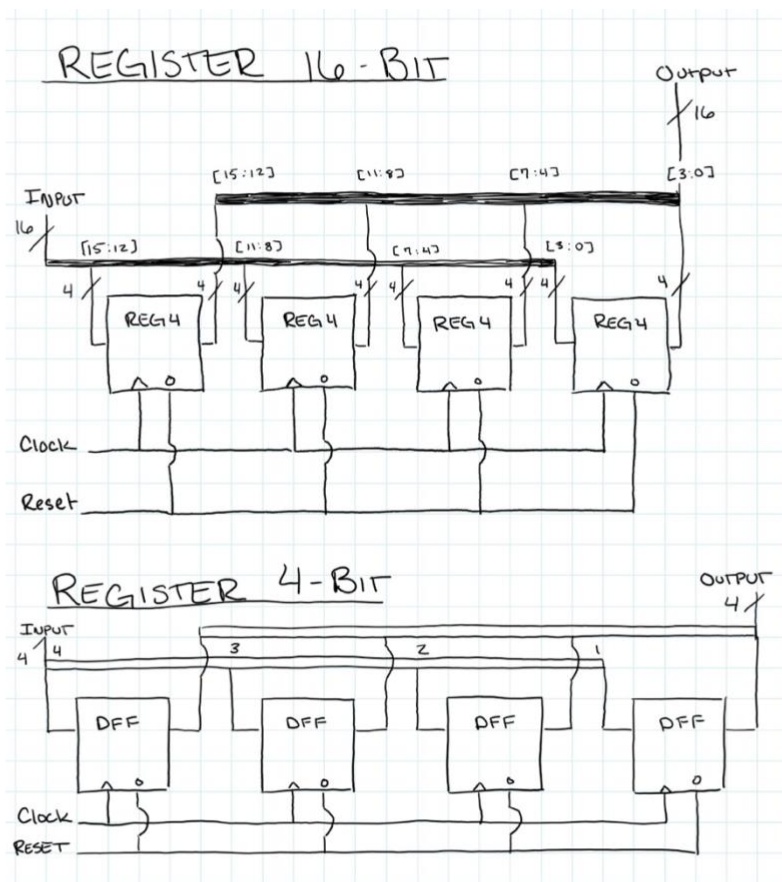
16 Bit XNOR Gate



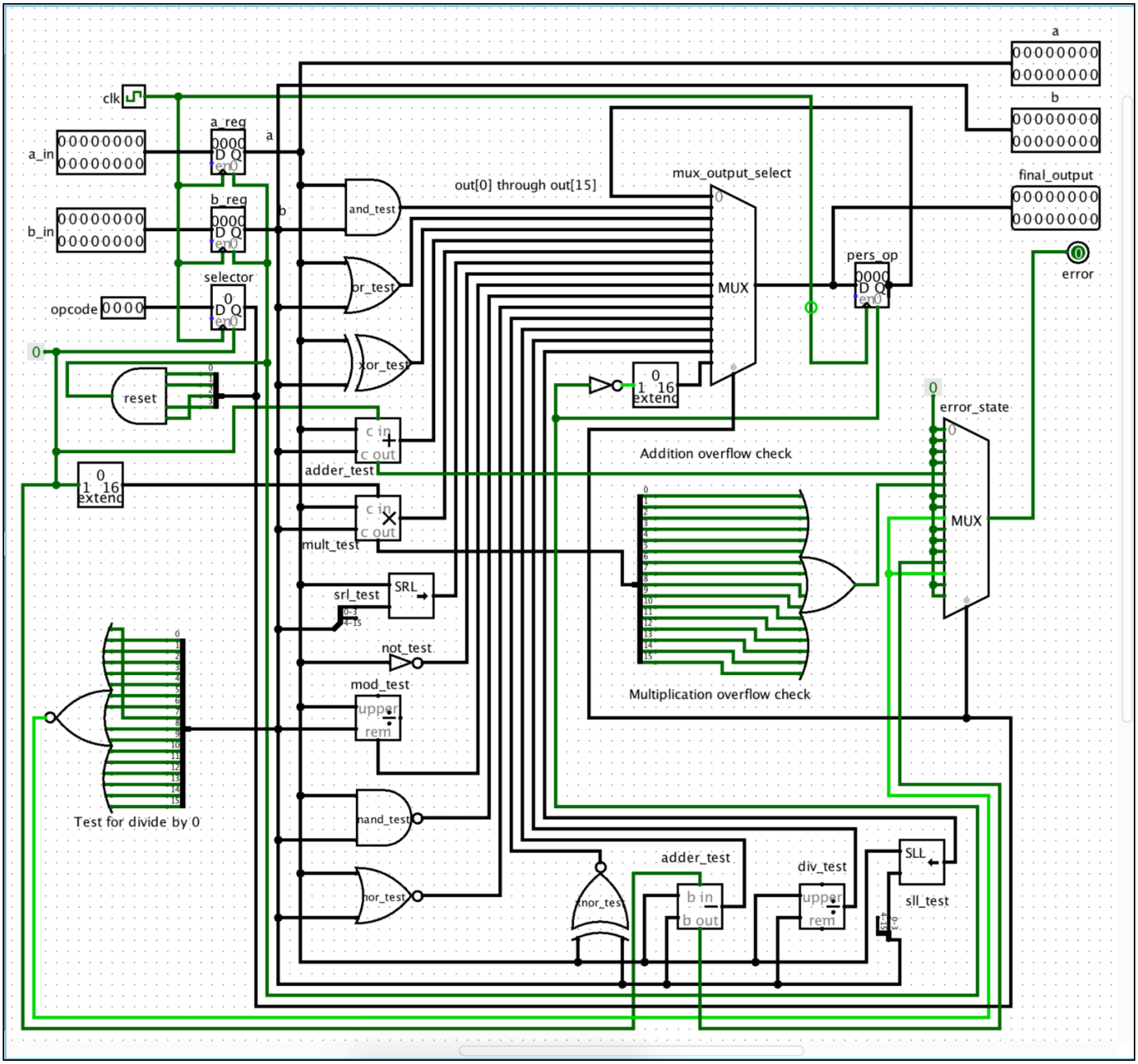
16 Bit XOR Gate



16 Bit Multiplexer



16 Bit and 4 Bit Register

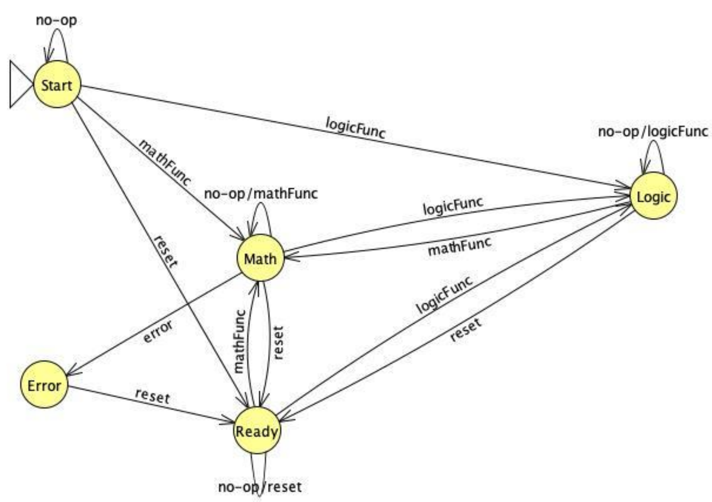


Complete ALU Diagram

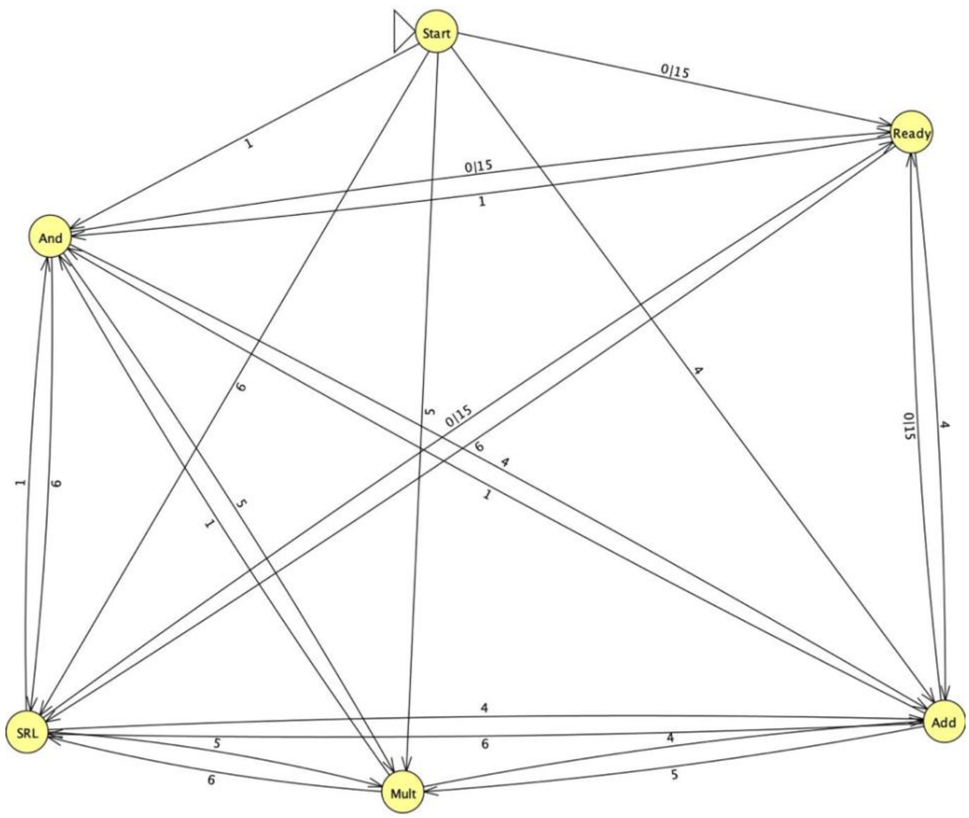
- The addition and subtraction components (adders) are separate on Logisim and thus do not exactly match the Verilog code. Our code uses an adder which has a mode bit to switch between addition and subtraction. The mode bit is the highest bit in the opcode. The opcode for addition is 0100 and subtraction is 1100, thus based on the MSB the output of the adder is modulated.

- SRL and SLL – The shifters are structurally coded and allow for a maximum shift of 15 bits to either side. This requires only the 4 LSB from B as the input and thus splitters are used to show the same.
- The error check of modulus and divide is done by taking the NOR of all the bits of the dividend B. Thus, only if B is 0 then NOR of its bits is 1 and there is a divide by zero error.
- The error check for multiplication involves the use of the carry out from the multiplication and taking an OR of all its bits, if even one of the bits is high, there has been an overflow and thus an error.
- Not function outputs the 'NOT' of the first input only. In this case, it outputs NOT of A.
- Reset is opcode 15 (1111). The opcode is used by taking an AND of all the bits, and only when the opcode is 15 and all the bits are high is the reset bit set to 1. When the reset bit is set to 1, it asynchronously resets all the registers in the system (output as well as input registers).
- The boxes on the top right marked 'a' and 'b' are outputs showing the current values of a and b in the system respectively.

State Machine Components



Top Level ALU State Machine



State Machine of ALU Components (Partial, since listing all components would lead to a complex diagram)

Detailed State Transitions

Start State Transitions							
##	Input			Current	Next	Output	
	A	B	Operation	State	State	A op B	Error
0	a	b	NO-OP	Start	Ready	x	x
1	a	b	AND	Start	And	a & b	0
2	a	b	OR	Start	Or	a | b	0
3	a	b	XOR	Start	Xor	a ^ b	0
4	a	b	ADD	Start	Add	a + b	0/1
5	a	b	MULT	Start	Mult	a * b	0/1
6	a	b	SRL	Start	Srl	a >> b	0
7	a	b	NOT	Start	Not	~a	0
8	a	b	MOD	Start	Mod	a % b	0/1
9	a	b	NAND	Start	Nand	~(a & b)	0
10	a	b	NOR	Start	Nor	~ (a | b)	0
11	a	b	XNOR	Start	Xnor	~(a ^ b)	0
12	a	b	SUBTRACT	Start	Subtract	a - b	0/1
13	a	b	DIVIDE	Start	Divide	a / b	0/1
14	a	b	SLL	Start	Sll	a << b	0
15	a	b	RESET	Start	Ready	0	0

State Transition Table (Detailed)

All State Transitions (Prev-Op = Any one of the 16)							
##	Input			Current State	Next State	Output	
	A	B	Operation			A op B	Error
0	a	b	NO-OP	Any State	Ready	Previous Output	Previous Error
1	a	b	AND	Any State	And	a & b	0
2	a	b	OR	Any State	Or	a | b	0
3	a	b	XOR	Any State	Xor	a ^ b	0
4	a	b	ADD	Any State	Add	a + b	0/1
5	a	b	MULT	Any State	Mult	a * b	0/1
6	a	b	SRL	Any State	Srl	a >> b	0
7	a	b	NOT	Any State	Not	~a	0
8	a	b	MOD	Any State	Mod	a % b	0/1
9	a	b	NAND	Any State	Nand	~(a & b)	0
10	a	b	NOR	Any State	Nor	~ (a | b)	0
11	a	b	XNOR	Any State	Xnor	~(a ^ b)	0
12	a	b	SUBTRACT	Any State	Subtract	a - b	0/1
13	a	b	DIVIDE	Any State	Divide	a / b	0/1
14	a	b	SLL	Any State	Sll	a << b	0
15	a	b	RESET	Any State	Ready	0	0

Table of All state transitions

Top Level State Diagram

ALU State Transitions								
##	Input				Current State	Next State	Output	
	A	B	PrevError	Operation			A op B	Error
0	x	x	x	no-op	Start	Start	x	x
1	x	x	x	reset	Start	Ready	0	0
2	a	b	x	mathFunc	Start	Math	a mathOP b	0/1
3	a	b	x	logicFunc	Start	Logic	a logOP b	0
4	x	x	x	no-op	Logic	Logic	prev. output	0
5	a	b	x	logicFunc	Logic	Logic	a logOP b	0
6	x	x	x	reset	Logic	Ready	0	0
7	a	b	x	mathFunc	Logic	Math	a mathOP b	0/1
8	x	x	0	no-op	Math	Math	prev. output	0
9	a	b	0	mathFunc	Math	Math	a mathOP b	0
10	x	x	1	error	Math	Error	x	1
11	x	x	x	reset	Math	Ready	0	0
12	a	b	0	logicFunc	Math	Logic	a logOP b	0
13	x	x	1	reset	Error	Ready	0	0
14	x	x	x	no-op	Ready	Ready	0	0
15	x	x	x	reset	Ready	Ready	0	0
16	a	b	x	mathFunc	Ready	Math	a mathOP b	0/1
17	a	b	x	LogicFunc	Ready	Logic	a logOP b	0

Top level state diagram table

Opcode:

Command	Opcode	Description
No Op	0000	No action on this clock tick
AND	0001	Bitwise AND of 2 16-bit numbers
OR	0010	Bitwise OR of 2 16-bit numbers
XOR	0011	Bitwise XOR of 2 16-bit numbers
ADD	0100	Addition of 2 of 2 16-bit numbers
MULTIPLY	0101	Multiplication of 2 16-bit numbers
SHIFT R	0110	Shift right num1 by num2 places
NOT	0111	Bitwise NOT of 1 16-bit number
MOD	1000	Modulus of 2 16-bit numbers
NAND	1001	Bitwise NAND of 2 16-bit numbers
NOR	1010	Bitwise NOR of 2 16-bit numbers
XNOR	1011	Bitwise XNOR of 2 16-bit numbers
SUBTRACT	1100	Subtraction of 2 16-bit numbers
DIVIDE	1101	Division of 2 16-bit numbers
SHIFT L	1110	Shift left num1 by num2 places
RESET	1111	Reset all registers on this clock tick

Modes of Operation:

Mode	Command	Description
Start	System Start	The state when the system is booted on.
Math	Any Math Function	State for add, subtract, multiply, divide and modulus.
Logic	Any Logical Function	State for N/AND, N/OR, X(N)OR, NOT, SLL, and SRL.
Error	Error in a math operation	Caused due to mul, add or sub overflow or divide by 0.
Ready	Reset	Asynchronously reset all internal registers to 0.

iVerilog Output:

Compilation: Have all three Verilog source files in the same folder and compile and output using

> > iverilog trohoc.v
> > ./a.out

Example output below and included in the text file output.txt

```
→  code git:(main) ✗ iverilog trohoc.v
→  code git:(main) ✗ ./a.out
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|Input A|    A Binary    |Input B|    B Binary    |Operation|Op Binary|Opcode|Output|    O Binary    |Error| TM|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|      1|0000000000000001|      1|0000000000000001|   AND   |     0001|     1|     1|0000000000000001|    0|  7|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|      9|0000000000001001|      6|0000000000000110|   OR    |     0010|     2|    15|0000000000001111|    0| 17|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|     15|0000000000001111|      9|0000000000001001|   XOR   |     0011|     3|     6|0000000000000110|    0| 27|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|     15|0000000000001111|      9|0000000000001001|  NO-OP  |     0000|     0|     6|0000000000000110|    0| 37|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|    355|0000000101100011|      5|0000000000000101|   ADD   |     0100|     4|   360|0000000101101000|    0| 47|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|  50000|1100001101010000|  50000|1100001101010000|   ADD   |     0100|     4| 34464|1000011010100000|    1| 57|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|      0|0000000000000000|      0|0000000000000000|  RESET  |     1111|    15|     0|0000000000000000|    0| 67|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|     25|0000000000011001|     25|0000000000011001| MULTIPLY|     0101|     5|   625|0000001001110001|    0| 77|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|   2500|0000100111000100|   2500|0000100111000100| MULTIPLY|     0101|     5| 24080|0101111000010000|    1| 87|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|      0|0000000000000000|      0|0000000000000000|  RESET  |     1111|    15|     0|0000000000000000|    0| 97|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|      0|0000000000000000|      0|0000000000000000|  NO-OP  |     0000|     0|     0|0000000000000000|    0|107|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|Input A|    A Binary    |Input B|    B Binary    |Operation|Op Binary|Opcode|Output|    O Binary    |Error| TM|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|     32|0000000000100000|      4|0000000000000100| SHIFT R |     0110|     6|     2|0000000000000010|    0|117|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|      0|0000000000000000|      1|0000000000000001|   NOT   |     0111|     7| 65535|1111111111111111|    0|127|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|    209|0000000011010001|     50|0000000000110010|   MOD   |     1000|     8|     9|0000000000001001|    0|137|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|      1|0000000000000001|      1|0000000000000001|  NAND   |     1001|     9| 65534|1111111111111110|    0|147|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|      1|0000000000000001|      1|0000000000000001|   NOR   |     1010|    10| 65534|1111111111111110|    0|157|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|      1|0000000000000001|      1|0000000000000001|  XNOR   |     1011|    11| 65535|1111111111111111|    0|167|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|  30000|0111010100110000|  25000|0110000110101000| SUBTRACT|     1100|    12|  5000|0001001110001000|    0|177|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|   9801|0010011001001001|    121|0000000001111001|  DIVIDE |     1101|    13|    81|0000000001010001|    0|187|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
|      1|0000000000000001|     15|0000000000001111| SHIFT L |     1110|    14| 32768|1000000000000000|    0|197|
+-------+----------------+-------+----------------+---------+---------+------+------+----------------+-----+---+
```