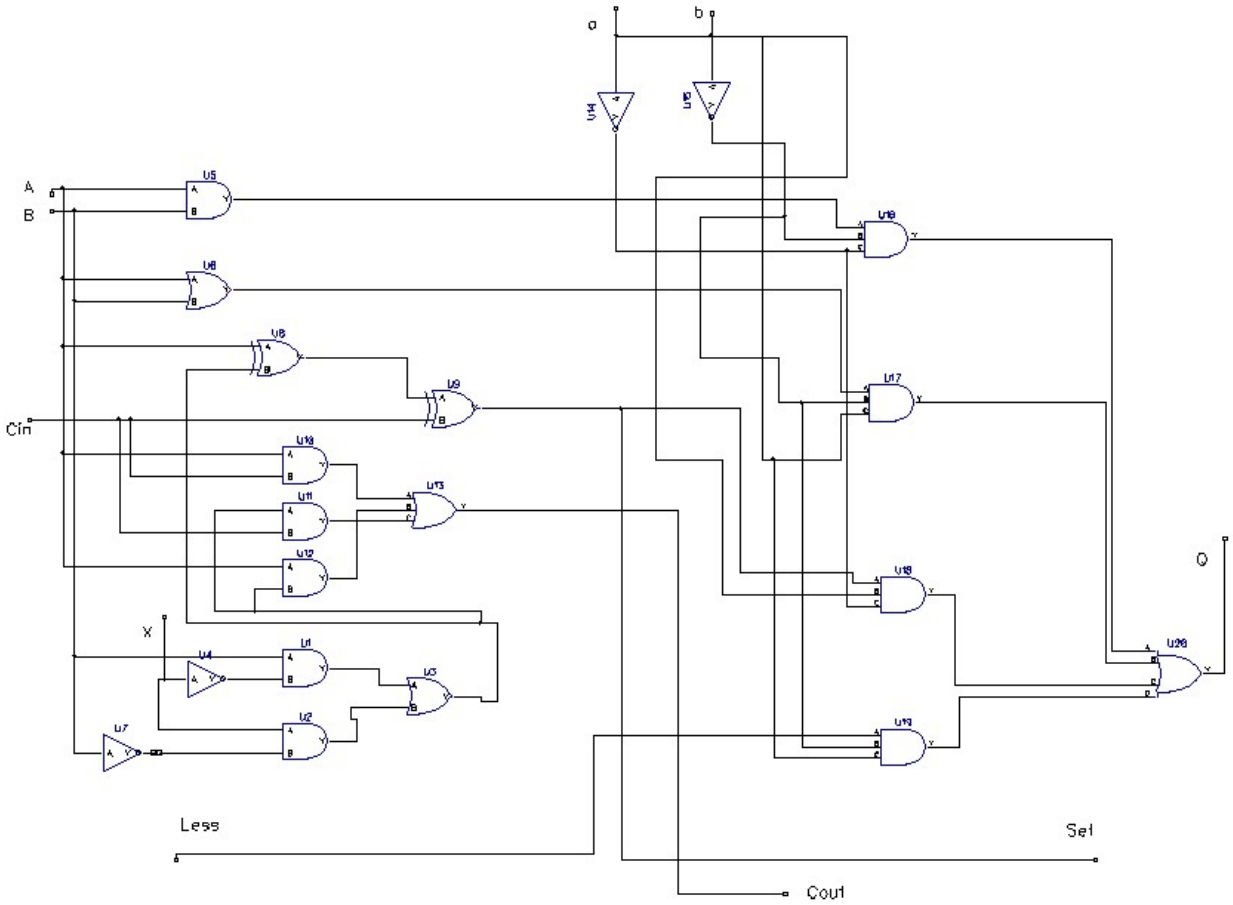
Florebencia Fils-Aime November 10, 2018

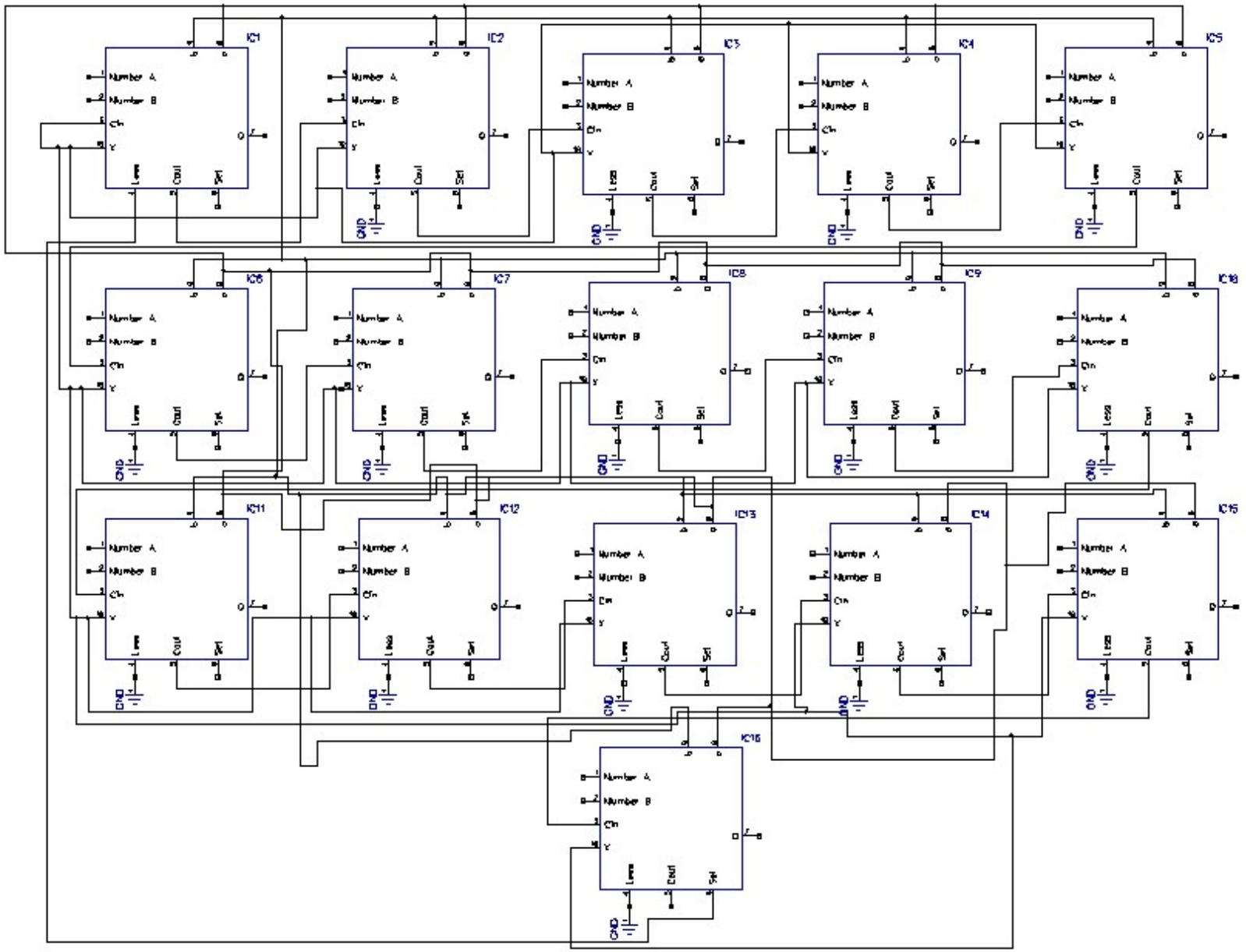
CSE 341: Computer Organization University at Buffalo

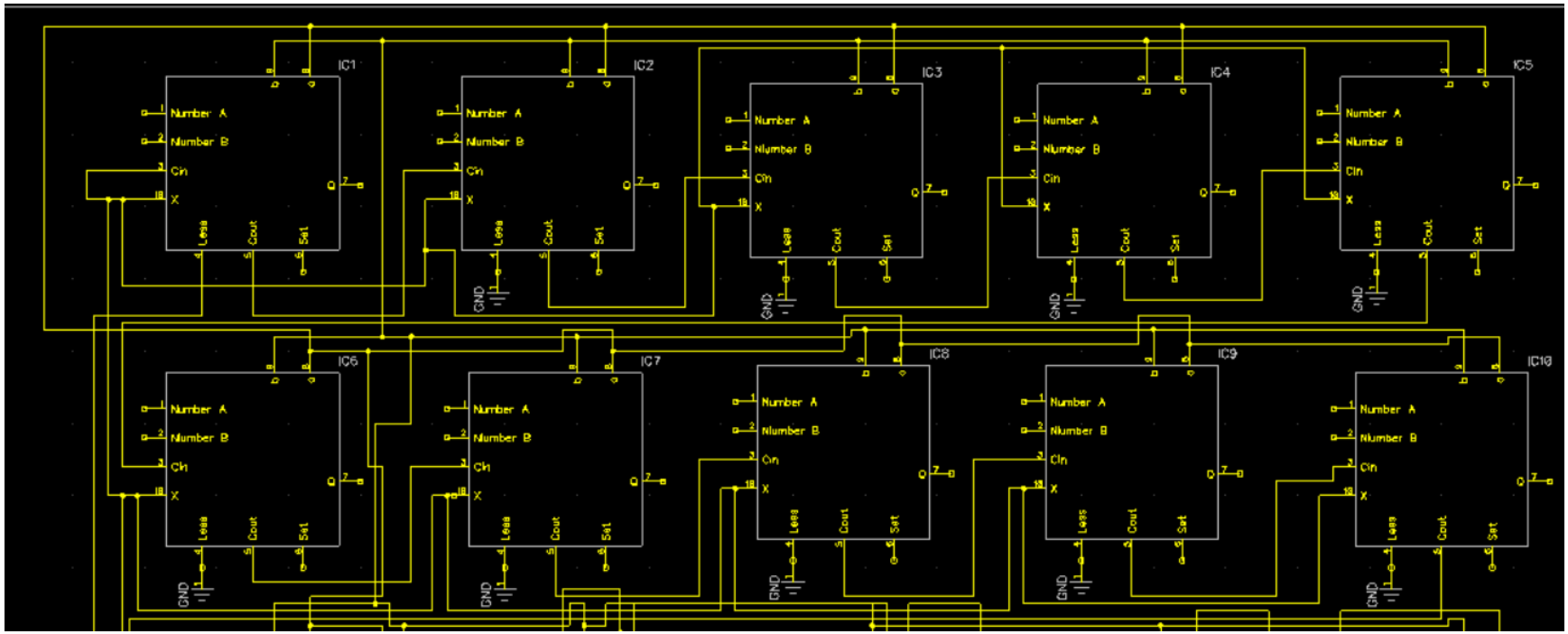
Project 2: Building an ALU

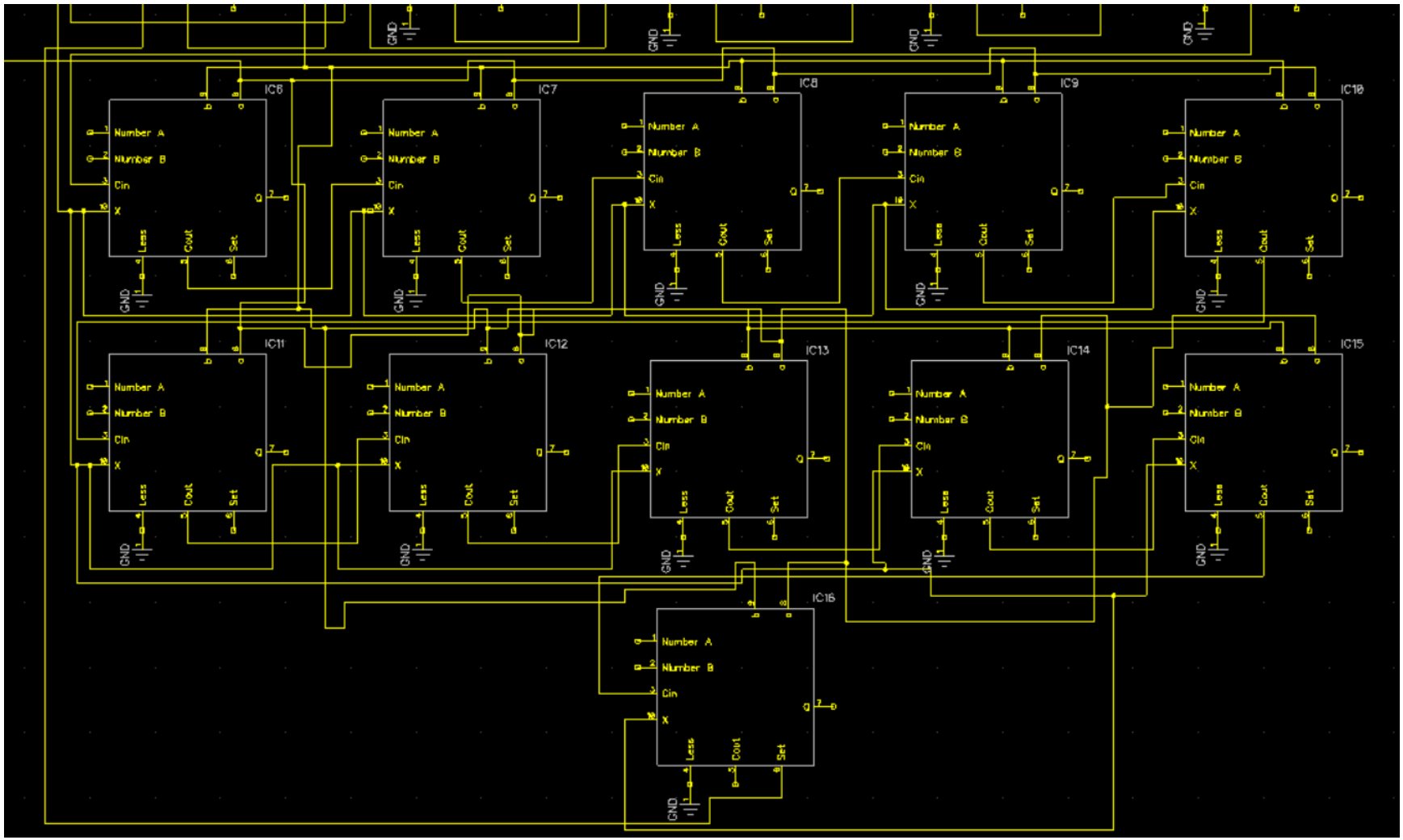
Gate Level of the one bit ALU



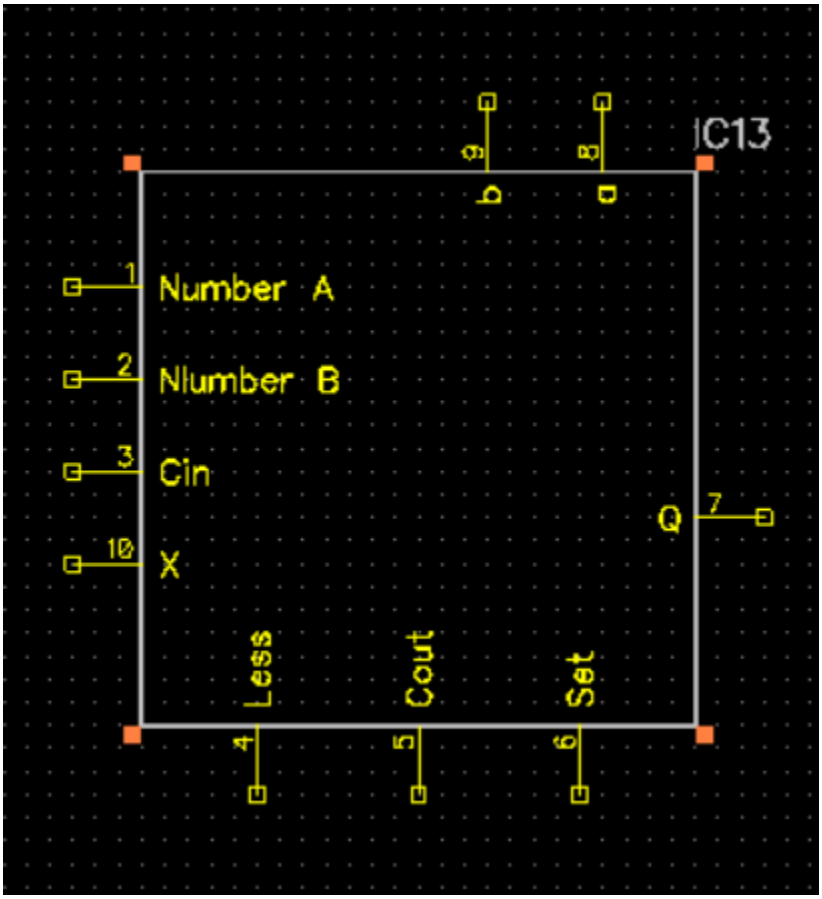
Q is the output of the 4 to 1 multiplexer. A and B are the numbers being used for arithmetic. The circuit above is the 1 bit ALU down to the gate level. The open wires a, b, and x for the opcode and their order of most significant bit is xab. For example if the opcode 110 was used then x = 1, a = 1, and b = 0. When number a is greater than or equal to number b, set less than returns a 0. If number b is greater than number a then SLT returns a 1.


IC1
IC2
IC3
IC4
IC5
IC6
IC7
IC8
IC9
IC10
IC11
IC12
IC13
IC14
IC15
IC16
Number A
Number B
Cin
Y
Less
Cout
Sel
GND


IC1
IC2
IC3
IC4
IC5
IC6
IC7
IC8
IC9
IC10
Number A
Number B
Cin
X
Q
Less
Cout
Set
GND

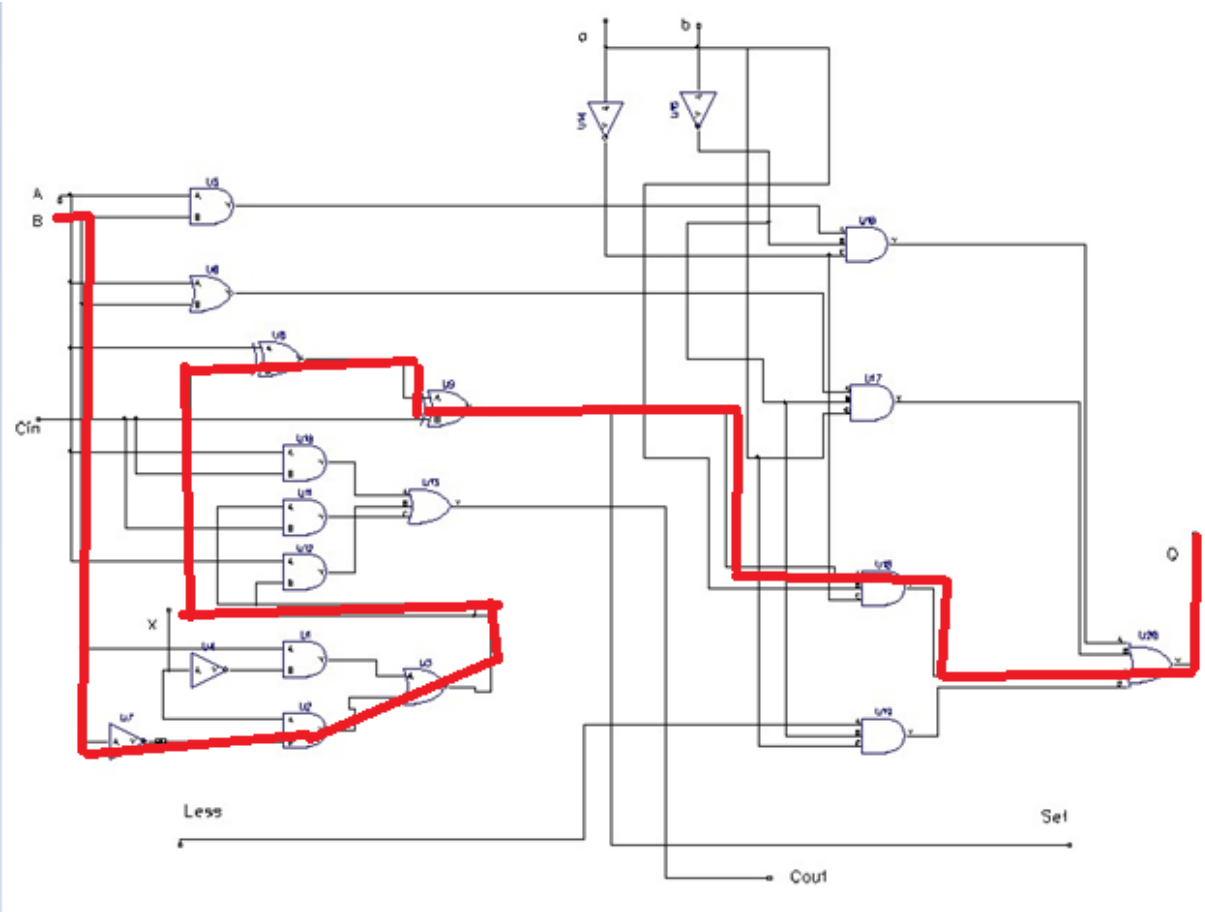

IC6
IC7
IC8
IC9
IC10
IC11
IC12
IC13
IC14
IC15
IC16
Number A
Number B
Cin
X
Q
Less
Cout
Set
GND

Sixteen bit ALU



Each IC looks like this up close. The connected 16 bit ALU circuit is at the page 2. A clearer version of the schematic is at page 3 and 4. The first bit is at the top left corner. You read this from left to right and then you start on the left in the next row. All of the xs are connected. All of the as are connected. All of the bs are connected to each other. The set of the 15th bit which is at the bottom is connected to the first bit's less. This returns a set less than of one or zero since the 15th bit is the most significant. It works because set is connected to the sum part of the full adder and less is actually connected to the 4 to 1 multiplexer. All the other less ports are connected to ground because they will always pass in a zero. The Cin of the first bit is connected to the X or the most significant bit of opcode was to decide if it would add 1 or 0 depending if it was an ADD or SUB instruction.

Critical Path



The critical path is the longest path taken by the circuit and in this case, the longest path is the SUB instruction. Since I used a 2 to 1 multiplexer, the SUB instruction is longer than the ADD instruction because of the not gate performing one's complement.

In the 1 bit ALU, the delay is 7 ns because there are seven gates along this path. In the 16 bit ALU, theoretically it should be 112 ns.

The gate cost of the ALU is how many gates are used in a circuit, so for the gate cost of 1 bit ALU is 20. For the 16 bit ALU, the gate cost is 320.

Instruction	Opcode
AND	000
OR	001
ADD	010
SUB	110
SLT	111

Zero Delay Simulation

```
timberlake {~/CSE341 Project 2} > cat sixteenbitaddsub.txt
                  0 a=    x, b=    x, s=    z, cout=z, op=xxx, choice=zzzzzzzzzzzzzzzz, q=    z
                  0 a=    x, b=    x, s=    x, cout=x, op=xxx, choice=xxxxxxxxxxxxxxxx, q=    X
                 20 a=    2, b=    3, s=    5, cout=0, op=010, choice=0000000000000011, q=    5
                 40 a=   10, b=    7, s=   17, cout=0, op=010, choice=0000000000000111, q=   17
                 60 a=   25, b=  657, s=  682, cout=0, op=010, choice=0000001010010001, q=  682
                 80 a= 1234, b=25768, s=27002, cout=0, op=010, choice=0110010010101000, q=27002
                100 a=    2, b=    3, s=65535, cout=0, op=110, choice=1111111111111100, q=65535
                120 a=  657, b=   25, s=  632, cout=1, op=110, choice=1111111111100110, q=  632
                140 a=25768, b= 1234, s=24534, cout=1, op=110, choice=1111101100101101, q=24534
                160 a=60000, b=50000, s=10000, cout=1, op=110, choice=0011110010101111, q=10000
                180 a=60000, b=50000, s=10000, cout=1, op=110, choice=0011110010101111, q=10000
```

Q is the output of the multiplexer and the s is the sum of the full adder. Choice can be ignored because it's the output of my 2 to 1 multiplexer. As you can see the 4 to 1 multiplexer is showing the right value and ADD and SUB instructions work.

Here is the values for AND and OR:

```
timberlake {~/CSE341 Project 2} > iverilog -t vvp alu_zero.v
timberlake {~/CSE341 Project 2} > vvp a.out > "test.txt"
timberlake {~/CSE341 Project 2} > cat test.txt
                  0 a=xxxxxxxxxxxxxxxx, b=xxxxxxxxxxxxxxxx, s=    z, cout=z, op=xxx, choice=zzzzzzzzzzzzzzzz, q=zzzzzzzzzzzzzzzz
                  0 a=xxxxxxxxxxxxxxxx, b=xxxxxxxxxxxxxxxx, s=    x, cout=x, op=xxx, choice=xxxxxxxxxxxxxxxx, q=xxxxxxxxxxxxxxxz
                 20 a=0000001111110010, b=0000001111110011, s= 2021, cout=0, op=000, choice=0000001111110011, q=0000001111110010
                 40 a=0011000101111010, b=1100110100110111, s=65201, cout=0, op=000, choice=1100110100110111, q=0000000100110010
                 60 a=1101110001011001, b=0000011110101001, s=58370, cout=0, op=000, choice=0000011110101001, q=0000010000001001
                 80 a=0000111011000010, b=0010100011101000, s=14250, cout=0, op=000, choice=0010100011101000, q=0000100011000000
                100 a=0000001111110010, b=0000001111110011, s= 2021, cout=0, op=001, choice=0000001111110011, q=0000001111110011
                120 a=0011000101111010, b=1100110100110111, s=65201, cout=0, op=001, choice=1100110100110111, q=1111110101111111
                140 a=1101110001011001, b=0000011110101001, s=58370, cout=0, op=001, choice=0000011110101001, q=1101111111111001
                160 a=0000111011000010, b=0010100011101000, s=14250, cout=0, op=001, choice=0010100011101000, q=0010111011101010
                180 a=0000111011000010, b=0010100011101000, s=14250, cout=0, op=001, choice=0010100011101000, q=0010111011101010
```

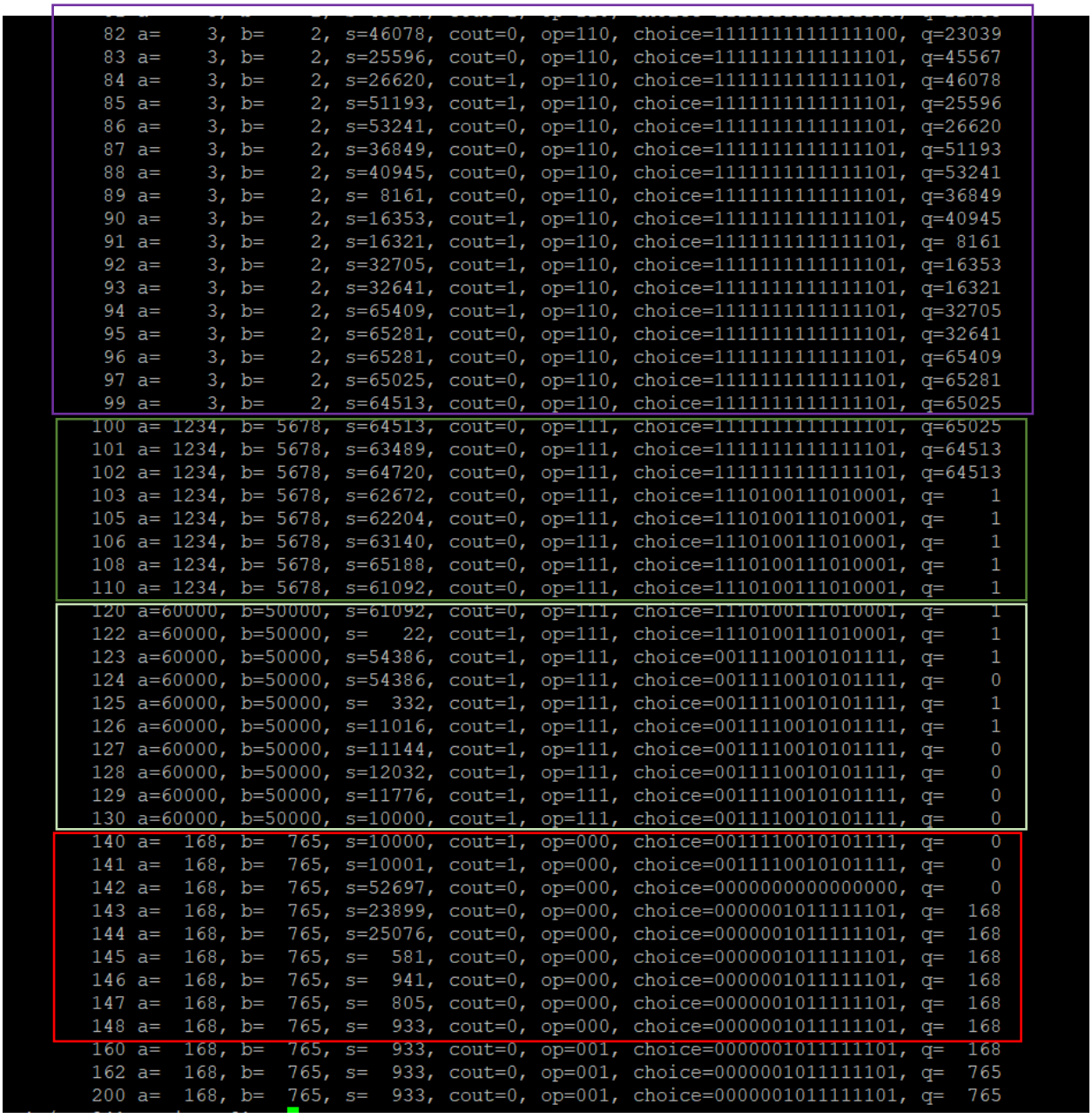
Here is the values for SLT:

```
timberlake {~/CSE341 Project 2} > iverilog -t vvp alu_zero.v
timberlake {~/CSE341 Project 2} > vvp a.out > "sixteenbitslt.txt"
timberlake {~/CSE341 Project 2} > cat sixteenbitslt.txt
                  0 a=    x, b=    x, s=    z, cout=z, op=xxx, choice=zzzzzzzzzzzzzzzz, q=zzzzzzzzzzzzzzzz
                  0 a=    x, b=    x, s=    x, cout=x, op=xxx, choice=xxxxxxxxxxxxxxxx, q=xxxxxxxxxxxxxxxz
                 20 a=    2, b=    3, s=65535, cout=0, op=111, choice=1111111111111100, q=0000000000000001
                 40 a=  654, b=  111, s=  543, cout=1, op=111, choice=1111111110010000, q=0000000000000000
                 60 a= 1234, b= 5678, s=61092, cout=0, op=111, choice=1110100111010001, q=0000000000000001
                 80 a=60000, b=50000, s=10000, cout=1, op=111, choice=0011110010101111, q=0000000000000000
                100 a=60000, b=50000, s=10000, cout=1, op=111, choice=0011110010101111, q=0000000000000000
```

When number a is greater than or equal to number b, set less than returns a 0. If number b is greater than number a then SLT returns a 1.

Unit Temporal Simulation

```
timberlake {~/CSE341 Project 2} > iverilog -t vvp alu_unit_temporal.v
timberlake {~/CSE341 Project 2} > vvp a.out > "sixteenbitunitdelay.txt"
timberlake {~/CSE341 Project 2} > cat sixteenbitunitdelay.txt
                   0 a=    x, b=    x, s=    z, cout=z, op=xxx, choice=zzzzzzzzzzzzzzzz, q=    z
                   0 a=    x, b=    x, s=    x, cout=x, op=xxx, choice=xxxxxxxxxxxxxxxx, q=    z
                   1 a=    x, b=    x, s=    x, cout=x, op=xxx, choice=xxxxxxxxxxxxxxxx, q=    X
                  20 a=    2, b=    3, s=    x, cout=x, op=010, choice=xxxxxxxxxxxxxxxx, q=    X
                  21 a=    2, b=    3, s=    x, cout=x, op=010, choice=xxxxxxxxxxxxxxxx, q=    x
                  22 a=    2, b=    3, s=    x, cout=x, op=010, choice=0000000000000xx, q=    x
                  23 a=    2, b=    3, s=    x, cout=x, op=010, choice=0000000000000011, q=    x
                  24 a=    2, b=    3, s=    x, cout=0, op=010, choice=0000000000000011, q=    x
                  25 a=    2, b=    3, s=    X, cout=0, op=010, choice=0000000000000011, q=    x
                  26 a=    2, b=    3, s=    5, cout=0, op=010, choice=0000000000000011, q=    x
                  27 a=    2, b=    3, s=    5, cout=0, op=010, choice=0000000000000011, q=    X
                  28 a=    2, b=    3, s=    5, cout=0, op=010, choice=0000000000000011, q=    5
                  40 a=  123, b= 5467, s=    5, cout=0, op=010, choice=0000000000000011, q=    5
                  42 a=  123, b= 5467, s=  124, cout=0, op=010, choice=0001010101011011, q=    5
                  43 a=  123, b= 5467, s=  126, cout=0, op=010, choice=0001010101011011, q=    5
                  44 a=  123, b= 5467, s= 5414, cout=0, op=010, choice=0001010101011011, q=  124
                  45 a=  123, b= 5467, s= 5526, cout=0, op=010, choice=0001010101011011, q=  126
                  46 a=  123, b= 5467, s= 5526, cout=0, op=010, choice=0001010101011011, q= 5414
                  47 a=  123, b= 5467, s= 5590, cout=0, op=010, choice=0001010101011011, q= 5526
                  49 a=  123, b= 5467, s= 5590, cout=0, op=010, choice=0001010101011011, q= 5590
                  60 a=    2, b=    3, s= 5590, cout=0, op=110, choice=0001010101011011, q= 5590
                  61 a=    2, b=    3, s= 5591, cout=0, op=110, choice=0001010101011011, q= 5590
                  62 a=    2, b=    3, s= 5550, cout=0, op=110, choice=1110101010100111, q= 5590
                  63 a=    2, b=    3, s= 5630, cout=0, op=110, choice=1111111111111100, q= 5591
                  64 a=    2, b=    3, s=59906, cout=0, op=110, choice=1111111111111100, q= 5550
                  65 a=    2, b=    3, s=65201, cout=0, op=110, choice=1111111111111100, q= 5630
                  66 a=    2, b=    3, s=65203, cout=0, op=110, choice=1111111111111100, q=59906
                  67 a=    2, b=    3, s=64867, cout=0, op=110, choice=1111111111111100, q=65201
                  68 a=    2, b=    3, s=64871, cout=0, op=110, choice=1111111111111100, q=65203
                  69 a=    2, b=    3, s=64199, cout=0, op=110, choice=1111111111111100, q=64867
                  70 a=    2, b=    3, s=64207, cout=0, op=110, choice=1111111111111100, q=64871
                  71 a=    2, b=    3, s=62863, cout=0, op=110, choice=1111111111111100, q=64199
                  72 a=    2, b=    3, s=62879, cout=0, op=110, choice=1111111111111100, q=64207
                  73 a=    2, b=    3, s=60191, cout=0, op=110, choice=1111111111111100, q=62863
                  74 a=    2, b=    3, s=60223, cout=0, op=110, choice=1111111111111100, q=62879
                  75 a=    2, b=    3, s=54847, cout=0, op=110, choice=1111111111111100, q=60191
                  76 a=    2, b=    3, s=54911, cout=0, op=110, choice=1111111111111100, q=60223
                  77 a=    2, b=    3, s=44159, cout=0, op=110, choice=1111111111111100, q=54847
                  78 a=    2, b=    3, s=44287, cout=0, op=110, choice=1111111111111100, q=54911
                  79 a=    2, b=    3, s=22783, cout=0, op=110, choice=1111111111111100, q=44159
                  80 a=    3, b=    2, s=23039, cout=1, op=110, choice=1111111111111100, q=44287
                  81 a=    3, b=    2, s=45567, cout=1, op=110, choice=1111111111111100, q=22783
```



Time Start and Finish of Each Instruction in nanoseconds					
	Instructions				
	ADD	***SUB***	***AND***	***OR***	***SLT***
start	20	60	140	160	100
end	28	79	148	162	110
start	40	80	X		120
end	49	99			130

X means not tested.

On each gate, I put a delay of one nanoseconds. There were eight tests done to analyze the temporal dependence of the delay. There were two add instructions, two sub instructions, one and instruction, one or instruction, and one slt instruction. The blue boxes contain ADD instructions, the purple boxes contain SUB instructions, the green boxes contain SLT instructions, the red box contains an AND instruction and the rest is an OR instruction.

An ADD instruction takes 8 to 9 ns to run. A SUB instruction takes 19 ns to run. An AND instruction takes 8 ns to run, and an OR instruction takes 2 ns. The SLT instruction takes 10 ns to run. They all take a different amount of time to run and they start at different times so that it is easier to see.

Analysis of the Average Delay

The average delay across all 1000 samples came out to be 7 ns which is the same value as the critical delay.