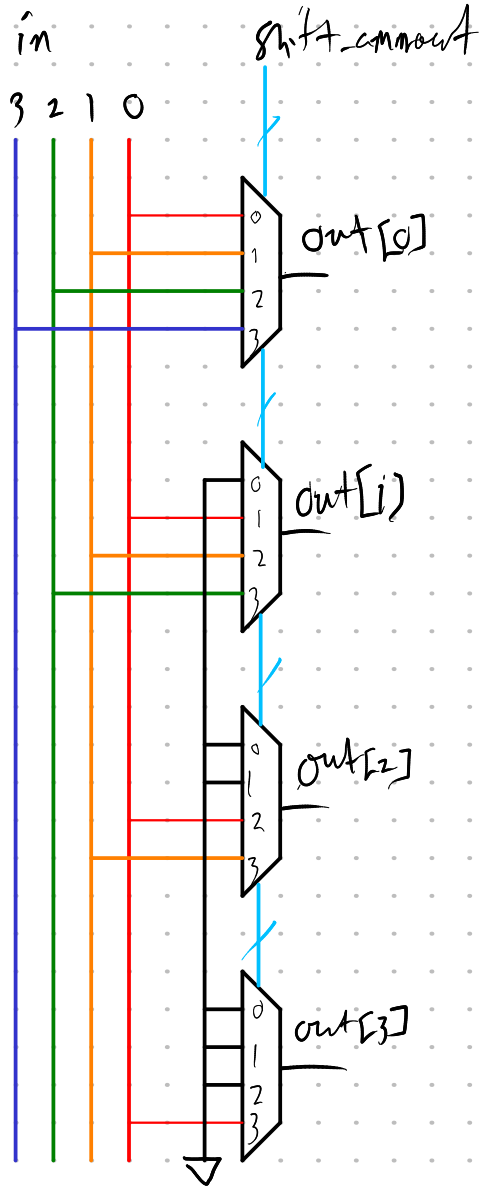
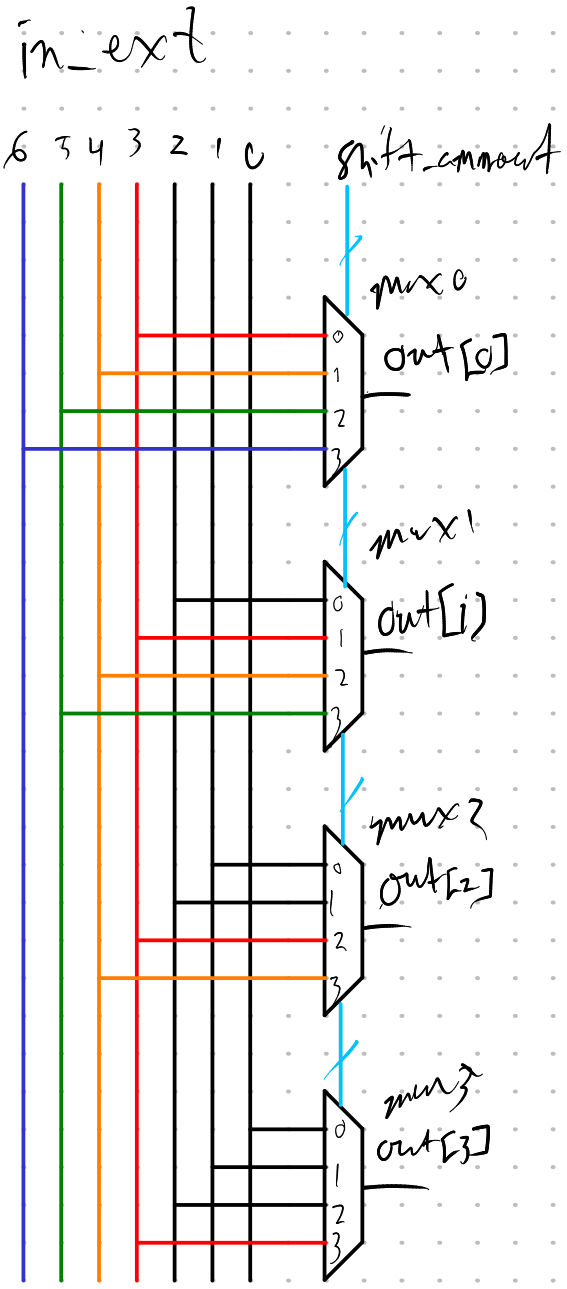
Logical Left Shift

in[n-1:0], shift_ammount[clog2(n)-1:0], out[n-1:0]



Schem uses n=4 for my sanity

IDEA: Concat in with bus of 0s.
Better for generate statements.

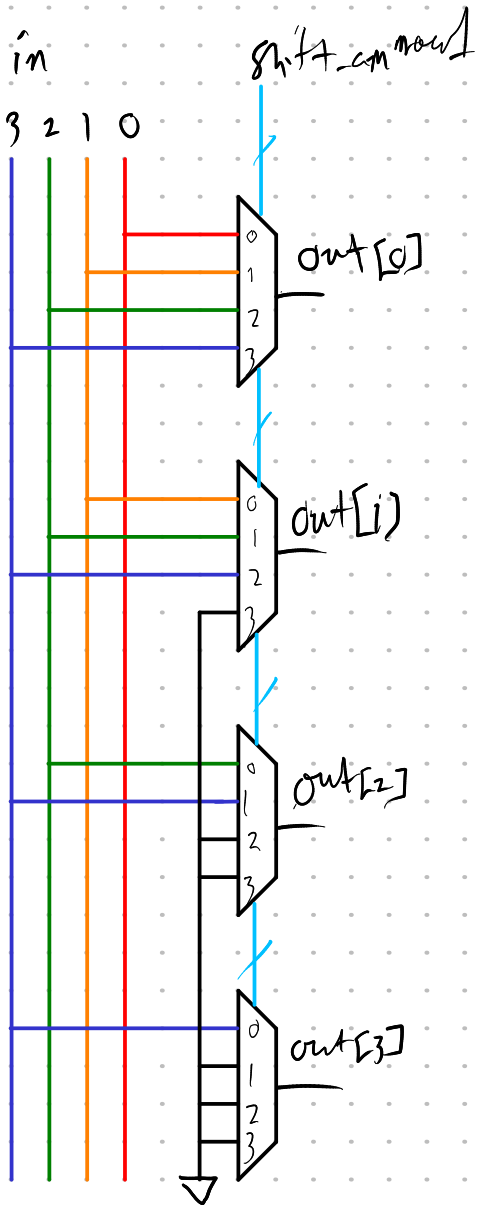


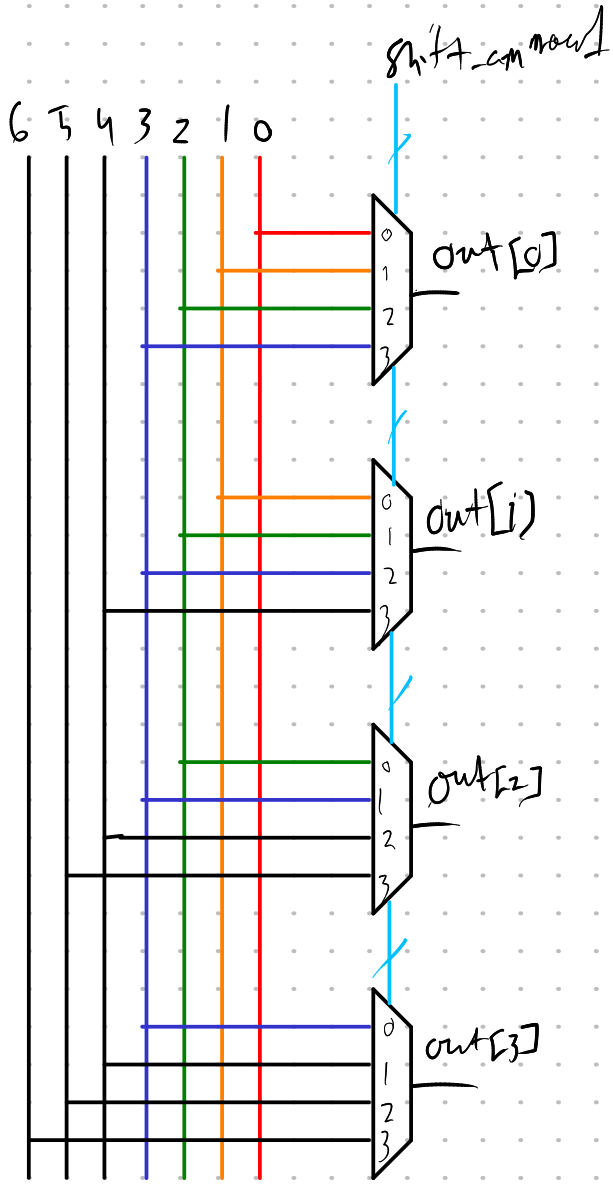
Logical Right Shift

in[n-1:0], shift_ammount[clog2(n)-1:0], out[n:0]

Schem using n=4 for my sanity

Idea: Concatinate with 0's



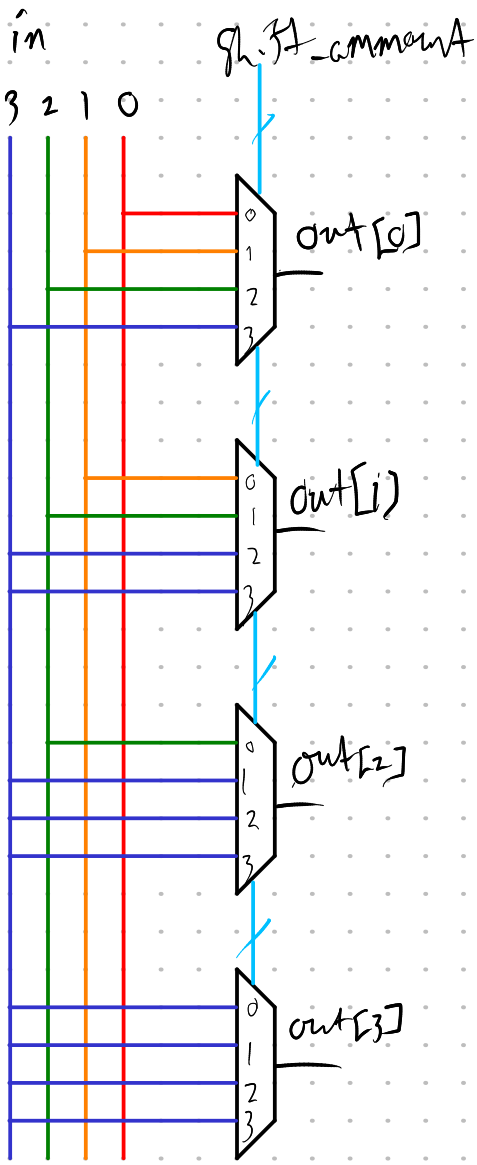


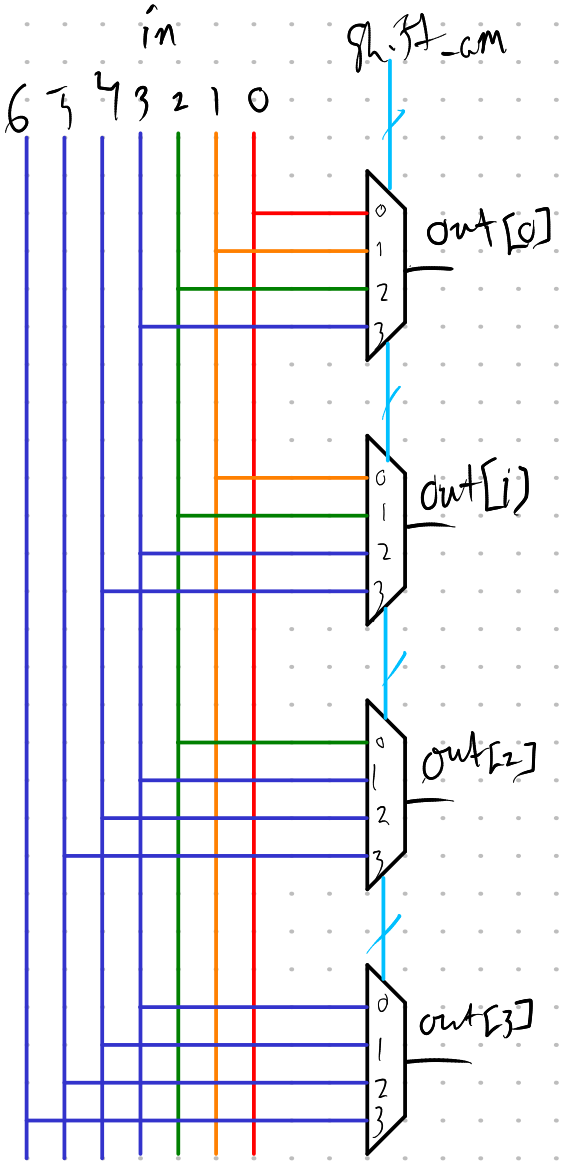
Arithmatic Shift Right

in[n-1:0], shift_amount[$\log_2[n]$-1:0], out[n-1:0]

Schem using n=4

Idea: extend bus again





ALU

in: a, b, control

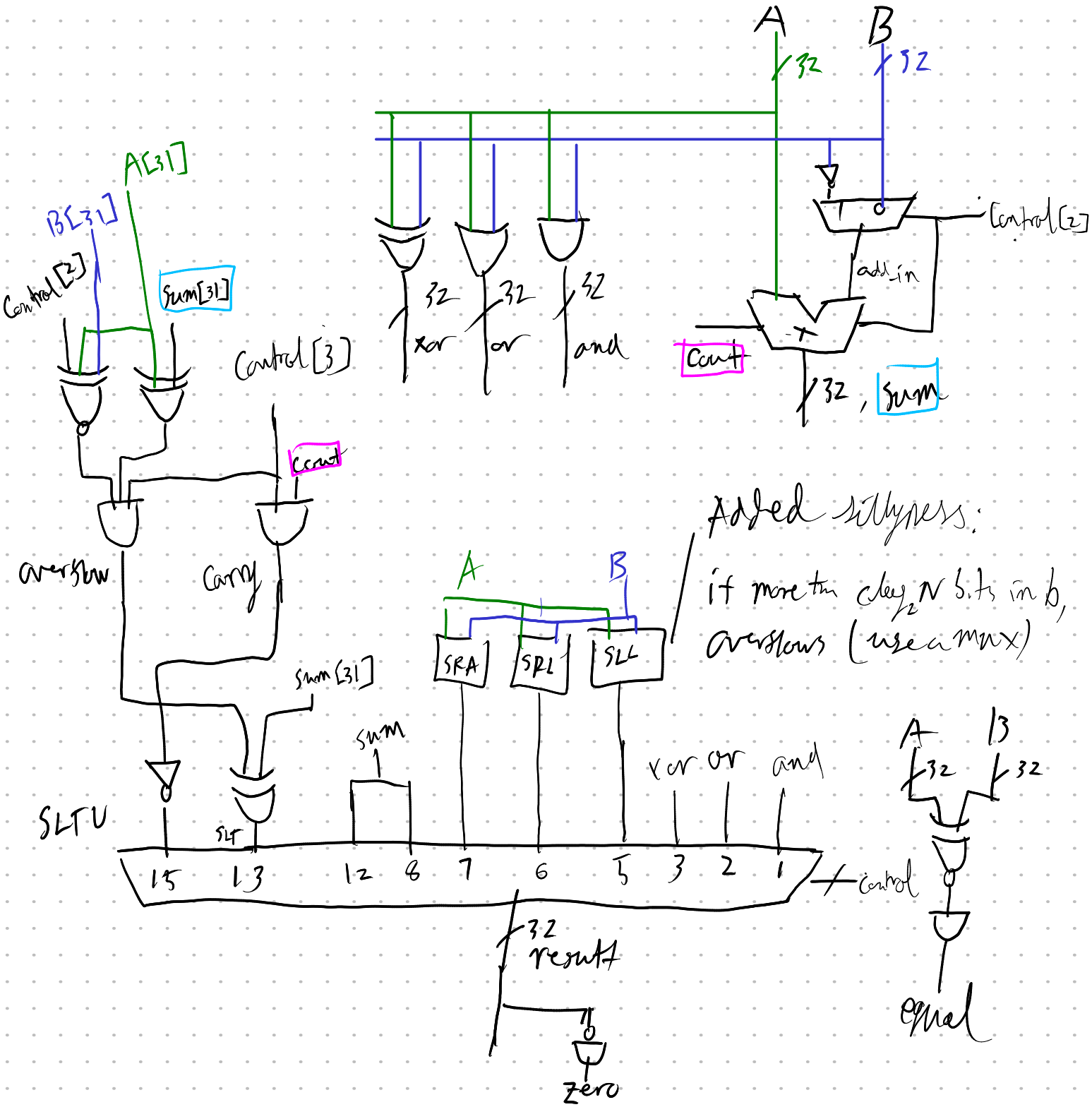
out: result, overflow, Zero, equal

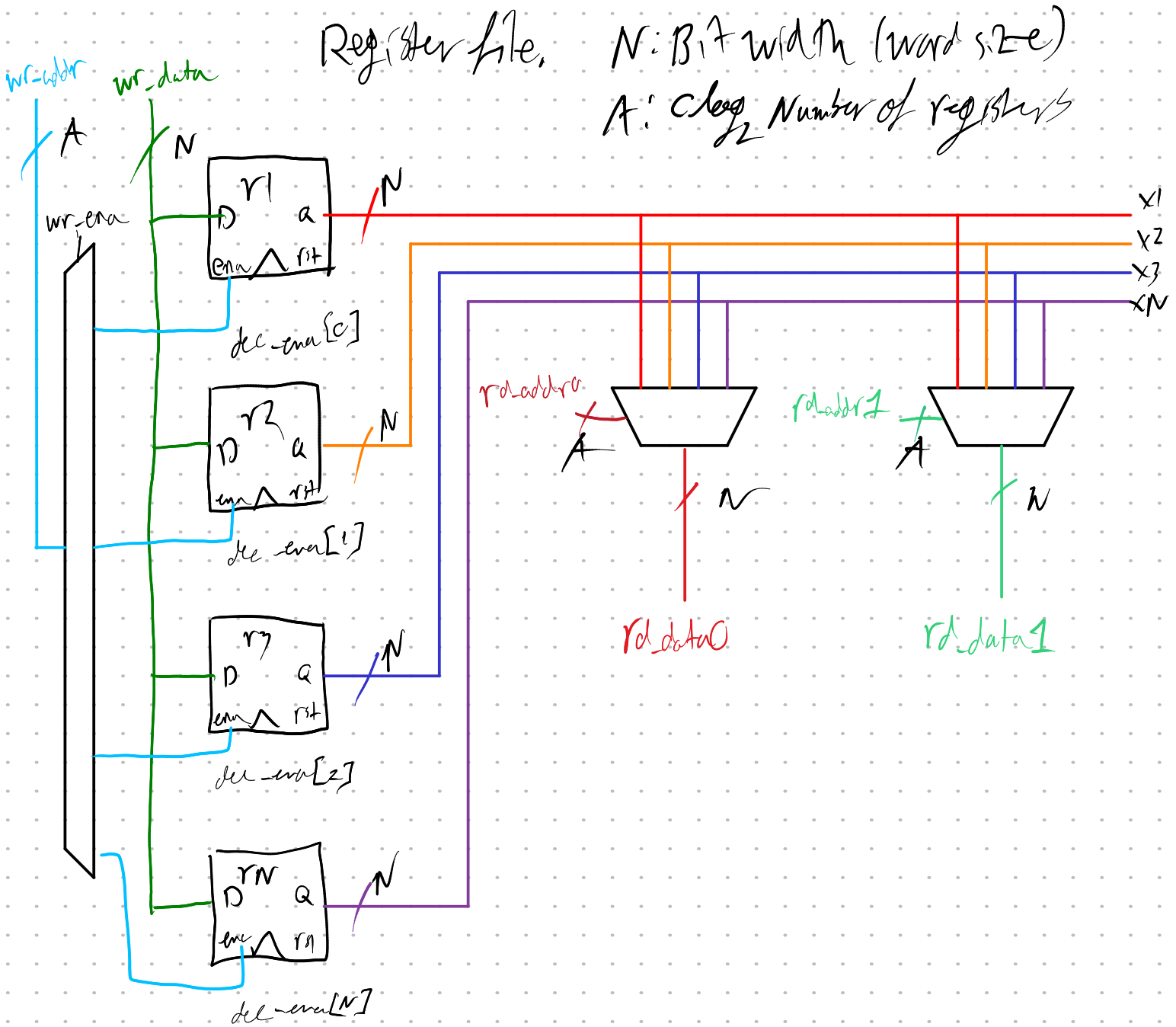
```
ALU_AND = 4'b0001,
ALU_OR  = 4'b0010,
ALU_XOR = 4'b0011,
ALU_SLL = 4'b0101,
ALU_SRL = 4'b0110,
ALU_SRA = 4'b0111,
ALU_ADD = 4'b1000,
ALU_SUB = 4'b1100,
ALU_SLT = 4'b1101,
ALU_SLTU = 4'b1111
```

bitwise

less than, signed

less than, unsigned




Register file. N: Bit width (word size)
A: clog2 Number of registers
wr_addr
wr_data
A
N
wr_ena
r1
D
Q
ena
rst
N
dec_ena[0]
r2
dec_ena[1]
r3
dec_ena[2]
rN
dec_ena[N]
x1
x2
x3
xN
rd_addr0
rd_addr1
rd_data0
rd_data1