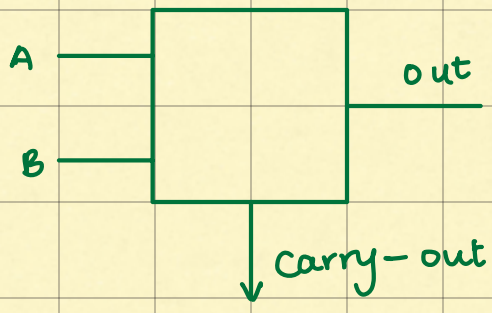
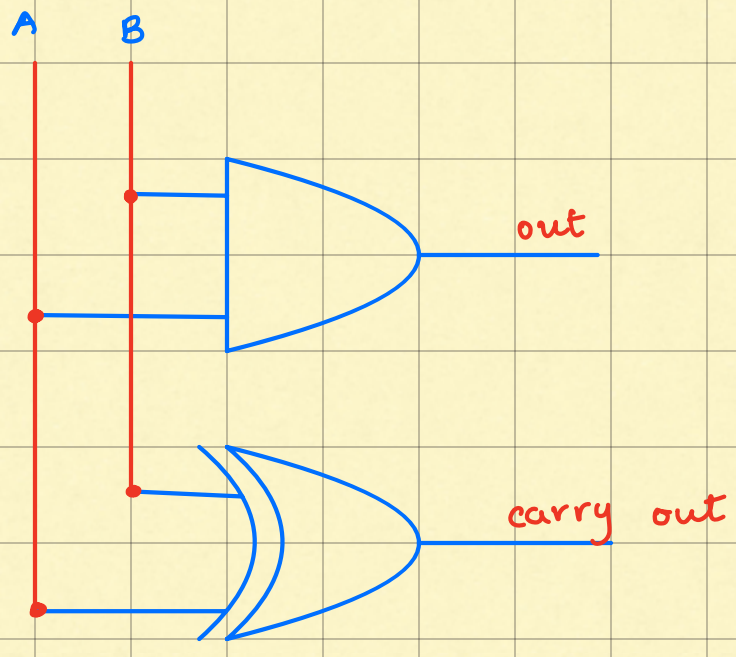
1 - bit half adder

A	B	(A+B)	A AND B	A XOR B
0	0	0 0	0	0
0	1	0 1	0	1
1	0	0 1	0	1
1	1	1 0	1	0

carry

out





1 bit full adder

A	B	Cin	A+B+Cin	
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

B[1] B[0]

K - map

AB \ Cin	0	1
00	0	0
01	0	1
11	1	1
10	0	1

B[1]

$AB + AC + BC$

K - map

AB \ Cin	0	1
00	0	1
01	1	0
11	0	1
10	1	0

B[0]

$\bar{A}BC + A\bar{B}C + \bar{A}\bar{B}C + ABC$

$= A \oplus B \oplus C$

Carry = $AB + BC + AC$

Sum = $A \oplus B \oplus C$

Gate	Transistor Cost
NOT	2
NAND, NOR	4
AND, OR	6
XOR, XNOR	12

, Finding best design

Gen: (n-1) mosfets for n bit input

n- AND = (n-1) NAND + NOT

n- OR = (n-1) OR .. ..

Design cost ⇒ $AB + BC + AC$

D. cost ⇒ 3 XOR

3 AND + 1 OR

⇒ $3 \cdot 6 + 2 \cdot 6 = 30T$

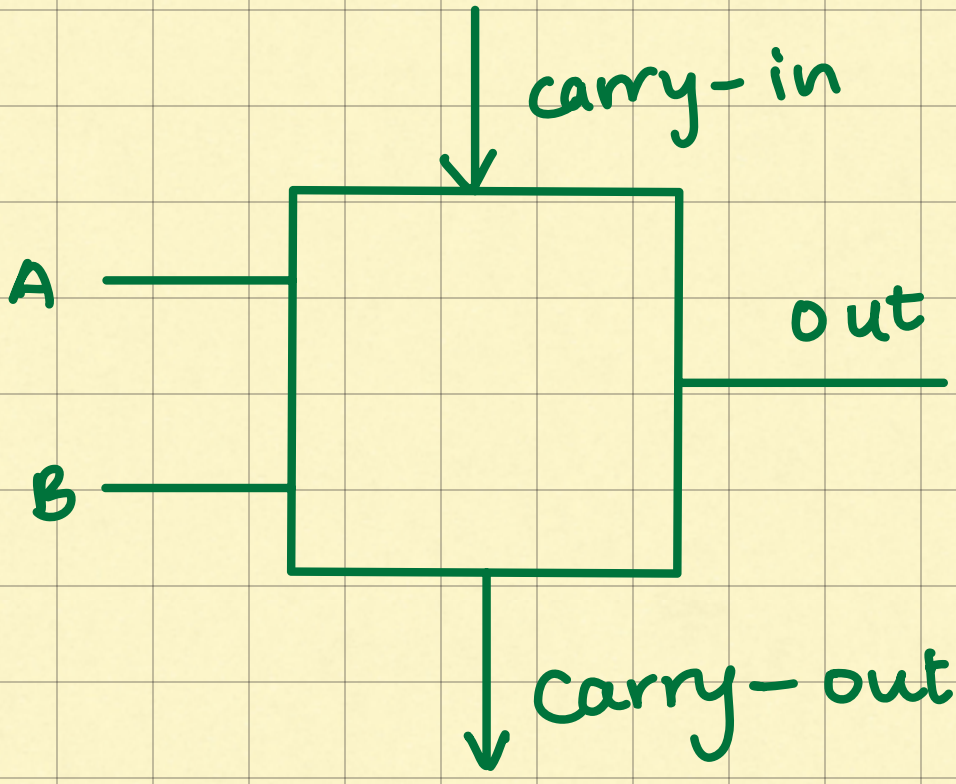
$3 \cdot 12 = 36T$

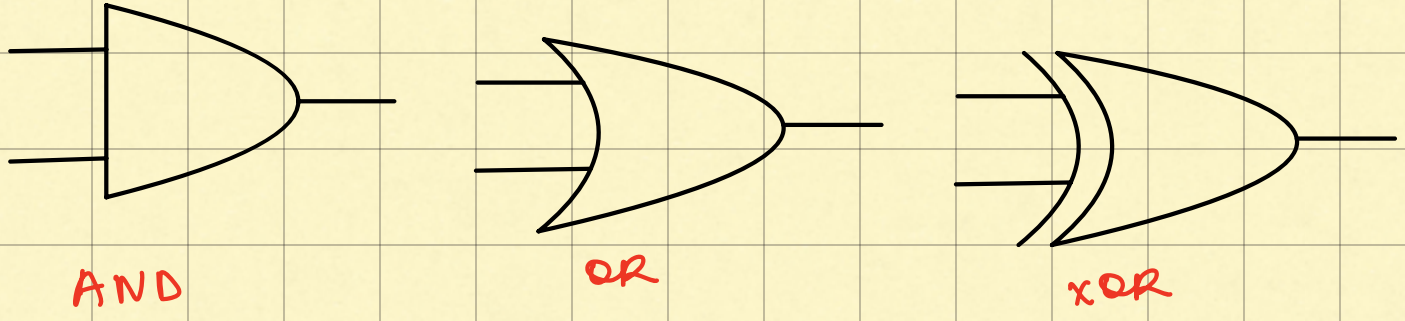
(ii) $AB + C(A \oplus B)$

2 AND + 1 OR + 1 XOR

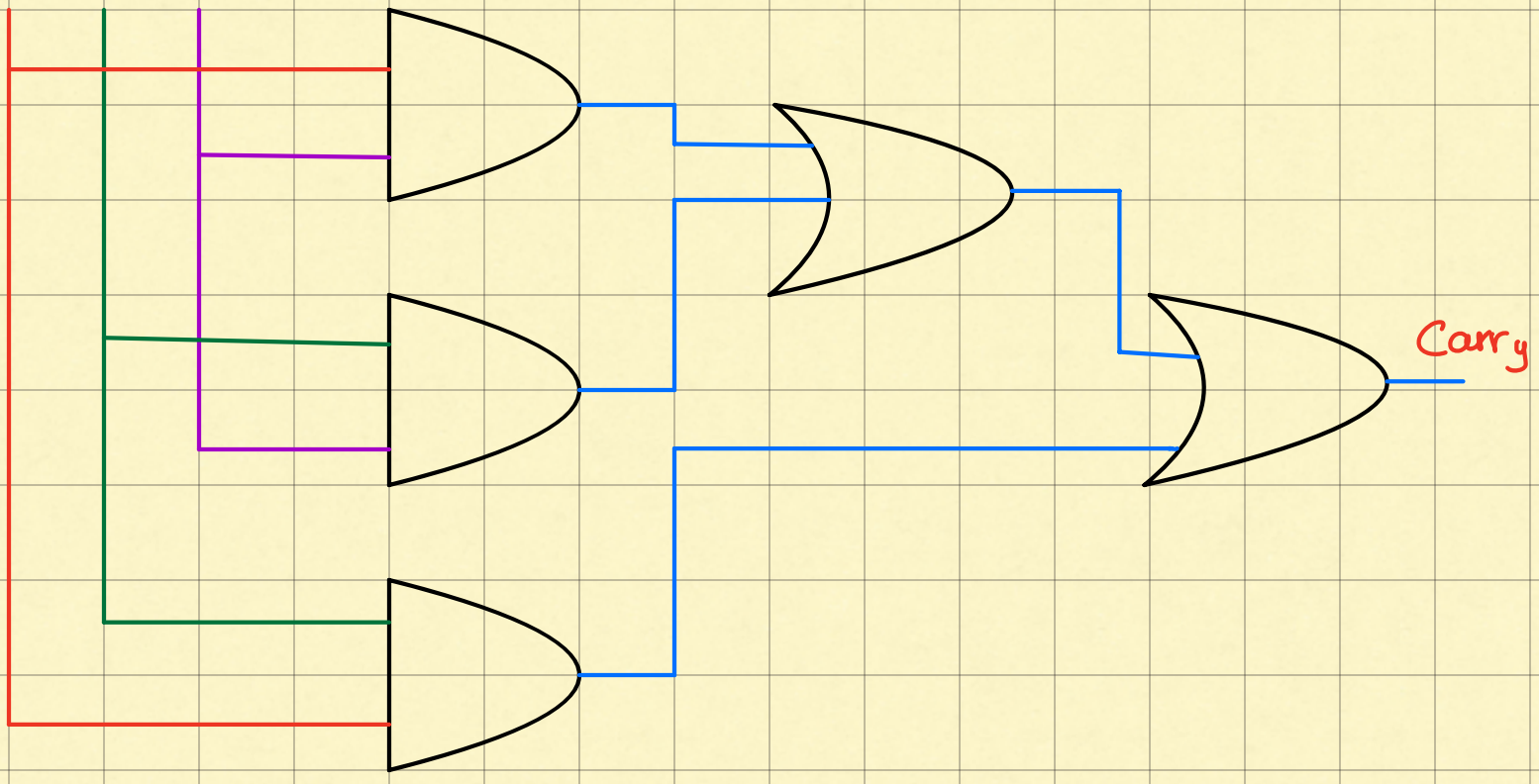
$2 \cdot 6 + 6 + 12 = 30T$

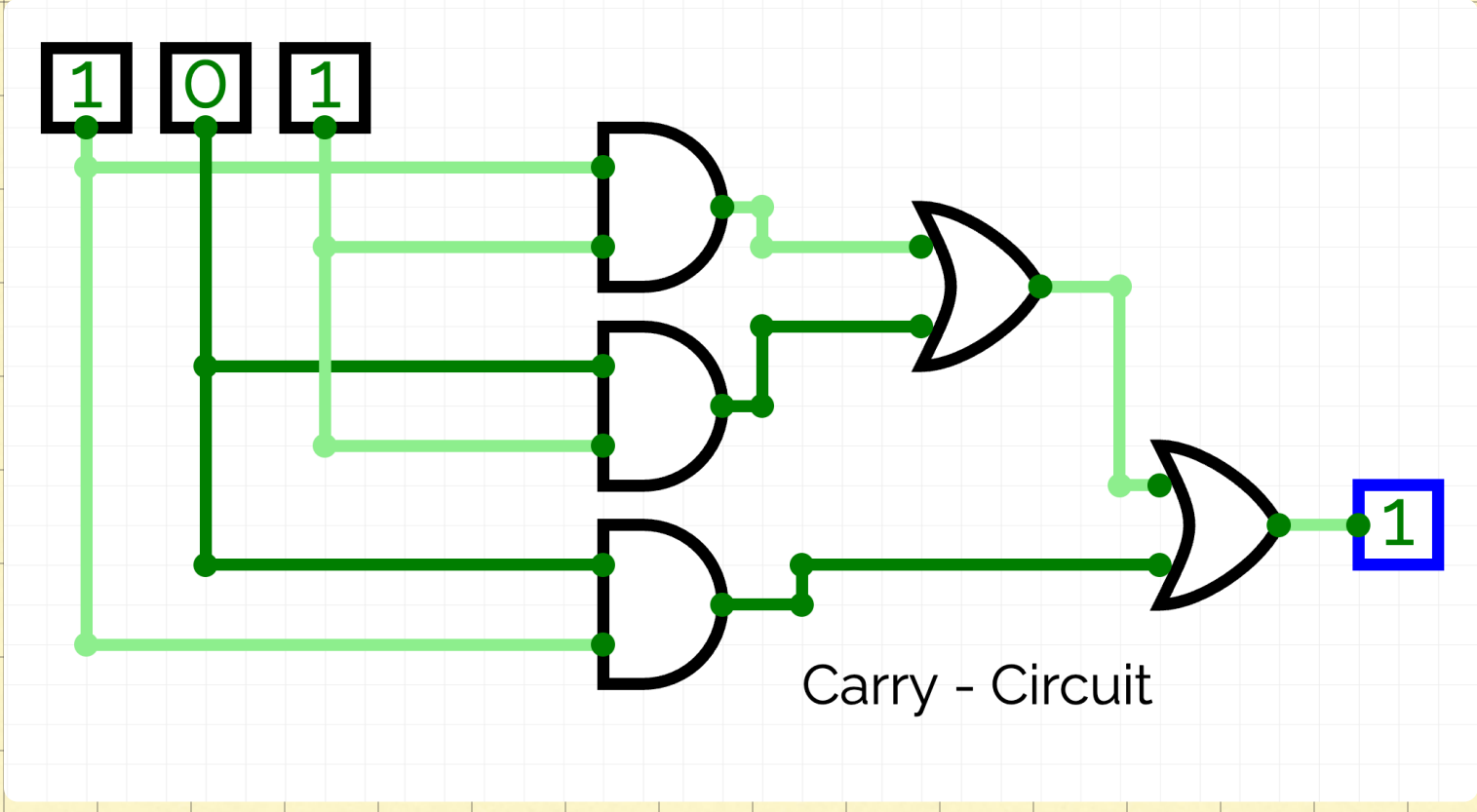
Final = $36 + 30 = 66T$





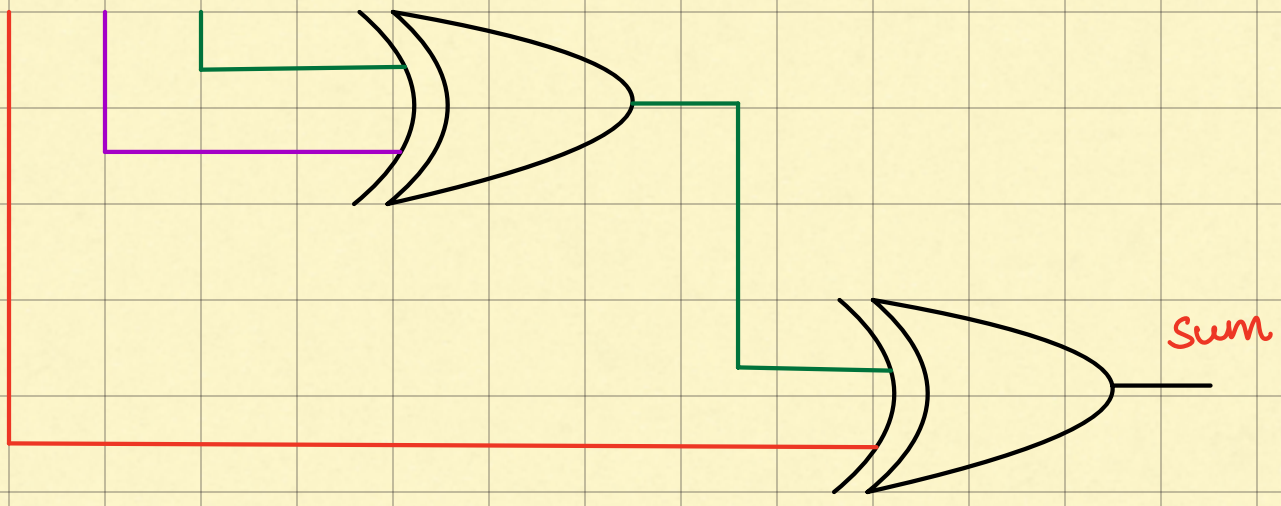
Designing AB + BC + AC, Carry

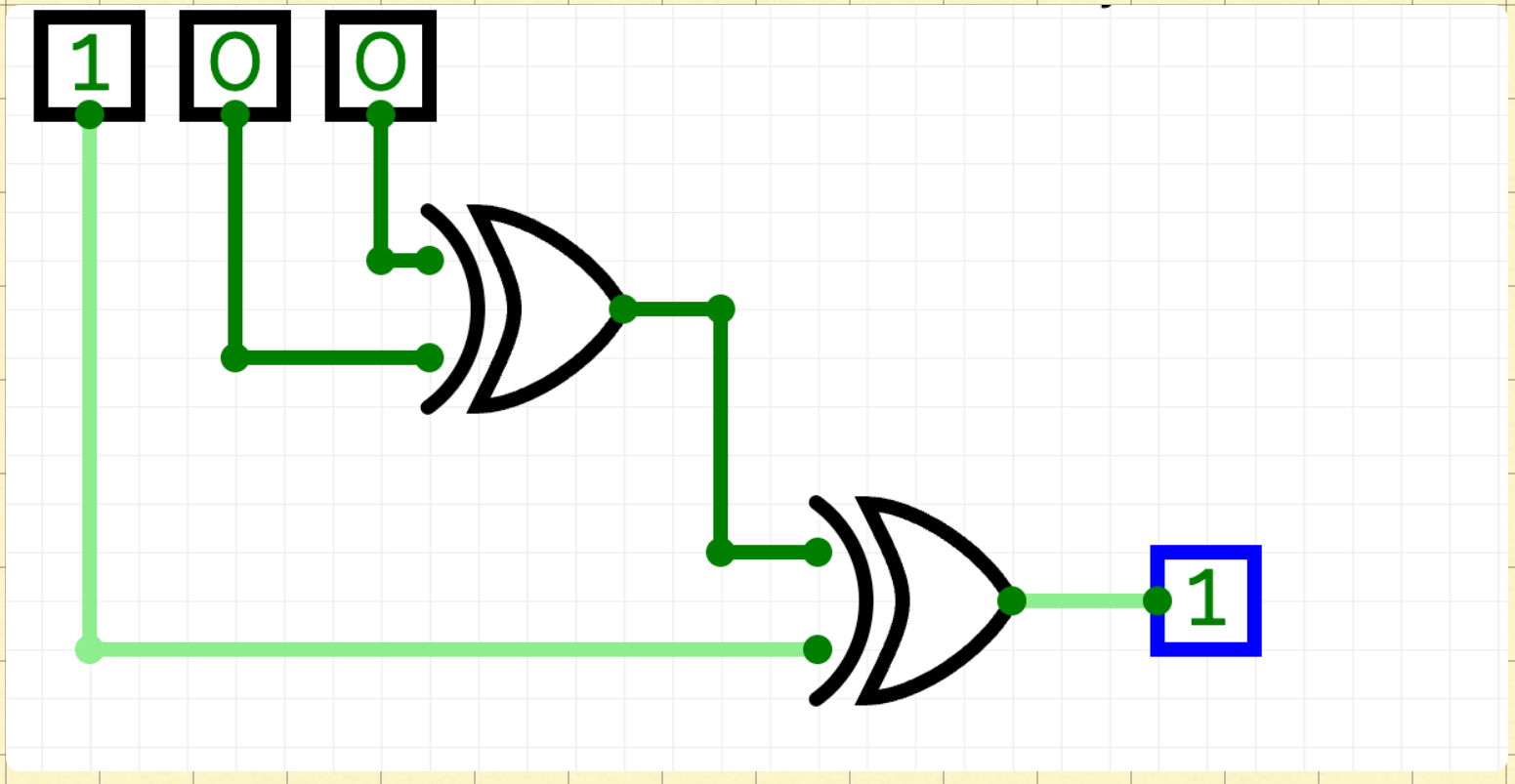




Simulation for Carry

Designing Sum = A ⊕ B ⊕ C





Simulation of sum

Current implementation adds two 1-bits.

Scaling to 8-bit full-adder

Assume 1 bit full adder as black box

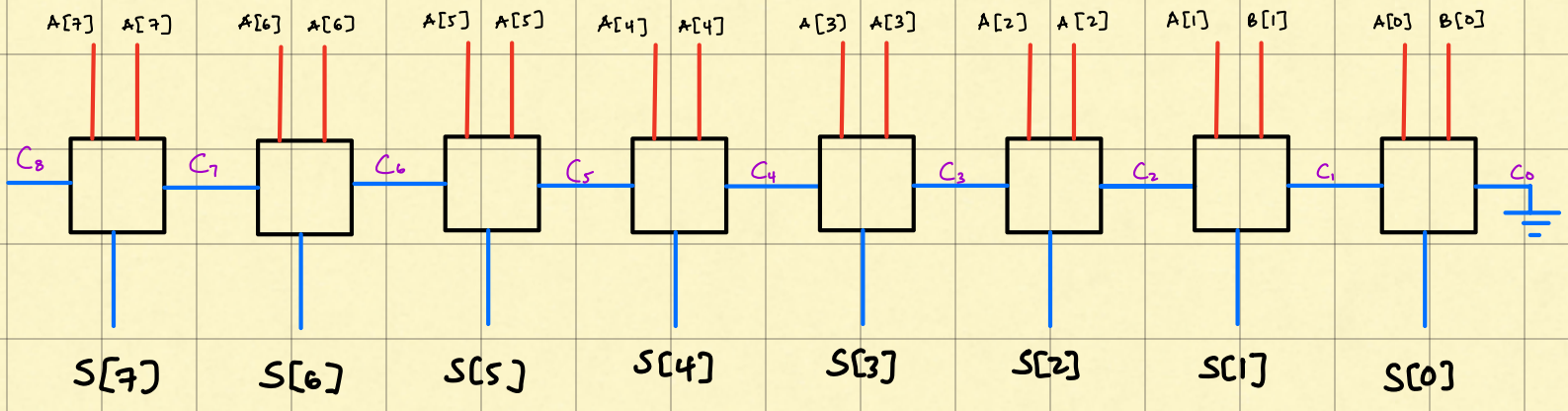
A is an 8-bit input A[7:0]

B is an 8-bit input B[7:0]

A+B = S, a 8-bit output, S[7:0]

C is a carry, 1 bit,

[$C_8 = C_{final}$ = 1 if overflow else 0] ** will address later



Currently, ADD is the only operation.

Improving to support subtraction SUB

By def, given ADD A B => A + B

SUB A B => A + (-B)

* Ignore storage register specs in assembly.

Since A & B are in 2's complement, SUB implements

Given B, $-B \Rightarrow NOT(B) + 1$, B is 8-bit

Demo, B = 0 0 0 1 1 0 1 1

NOT B = 1 1 1 0 0 1 0 0

$+1$

$-B = 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1$

⇒ ALU must know when to add / subtract.

A control line is created.

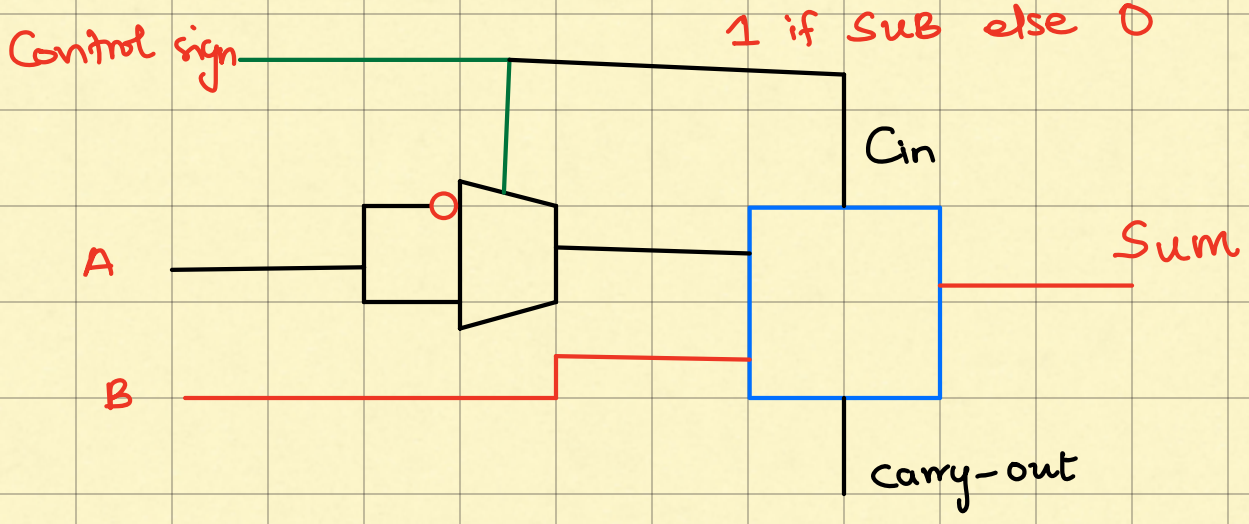
In current implementation, only 2 operations are supported * will be improved to support MUL (multiplication) and DIV (divide)

Also, same ALU will handly logical operations, not to be deficient in control signal bits, I implement 4 bit CS.

2^4 = 16 possible operations, easily scalable

I'll continue however by building SUB hardware and defining output by a selector mux

Logic with Control signal



Logic

Control line if 1 will supply adder with 1 to add, which completes 2's complement

=> Given A - B

The SUB first NOT (B) while control adds 1 to complete final 2's comp negation of B

Cons of Design

It requires at least 8 of 3 input AND gates and 8 input OR gates and not gates.

~ $2 \cdot 8 \cdot 6T + (8-1) \cdot 6$

$(16 + 7) \cdot 6T = \sim 138T$ excluding NOT gates

Different approach

Instead of dedicated mux for sub, instead, ill implement the logic circuit switched by the control signals that allows using NOT operations to handle subtraction

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

After carefully examining the XOR operation, perhaps a logic that I found very "beautiful"
Assume the control signal instead is

ADD = 0000 0000

SUB = 1111 1111

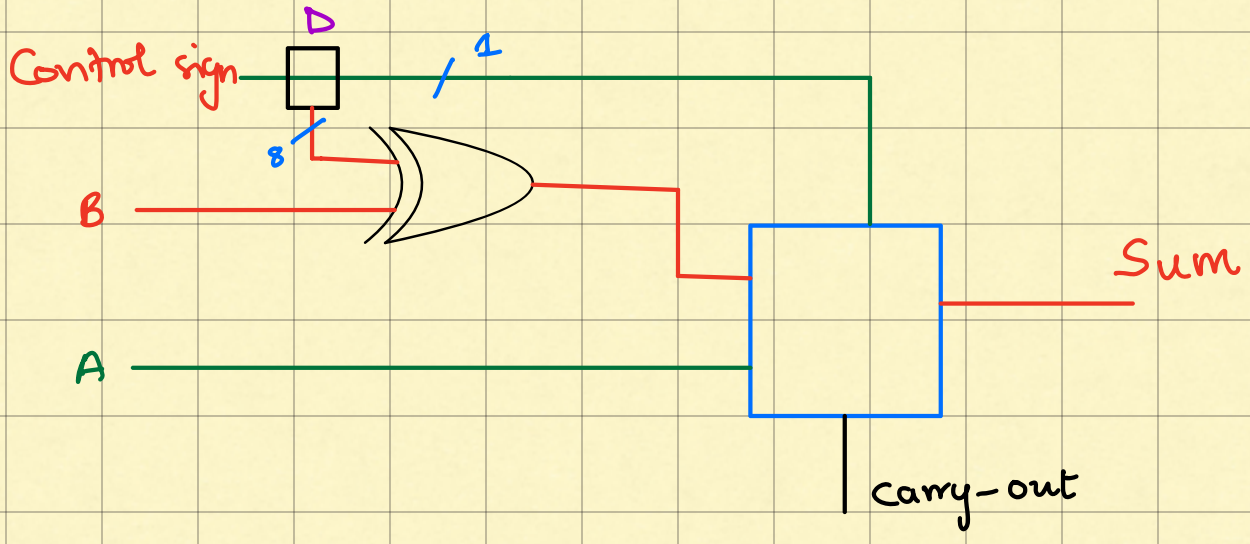
where B is always passed through an XOR with that given $B \oplus ADD = B$

$B \oplus SUB = B'$ and the SUB control (1bit) feeds the ALU to complete $B \rightarrow -B$

Advantage ; 8 bit XOR ⇒ 8 · 12 = 96T

96T < 138T

Logic with XOR design



D is a black box that given CT. Sig in 4 bits, sends 1 bit to adder, and sends 8 bit of 1s or 0s to XOR. Note its 1 bit extended by 1s or 0s given operation.

With the growing operations, it is obvious a control unit is needed with its decoder to process for the diff operators.

Building Control Unit