Tyler Goodwyn

Erik Vallejo

Stewart Dulaney

Israel

We tried to implement a 4-bit adder/subtractor on a breadboard too and assembled all the parts only to find out that the AND and XOR gates we ordered online did not work.

Our multifunctional ALU has four modes:

S_1 S_0	Mode
0 0	Add
0 1	Subtract
1 0	Multiply
1 1	Divide

Multifunctional ALU

Adder

See 2-bit Full Adder

If A were 11 (-1 in decimal)

+B were 11 (-1 in decimal)

```
  1
  11
+ 11
-----
 110
```

The inputs would be A_0: 1

A_1: 1

B_0: 1

B_1: 1

The ouputs would be S_0: 0

C_0: 1

S_1: 1

C_1: 1

The Sum values + last Carry value correspond to the digits of the result, which would be 1110 (-2 in decimal) sign extended to fit a four bit result

Subtraction

See 2-Bit Ripple Carry Full Adder w/ Control Signal.

If A were 11 (-1)

and B were 01 (1), after passing through the XOR gate, and control signal S were 1

The inputs would be:

A_0: 1
A_1: 1
B_0: 1
B_1: 0

```
  11          11
+ 01        + 11
----        ----
            110
```

The outputs would be:

S_0: 0
C_0: 1
S_1: 1
C_1: 1

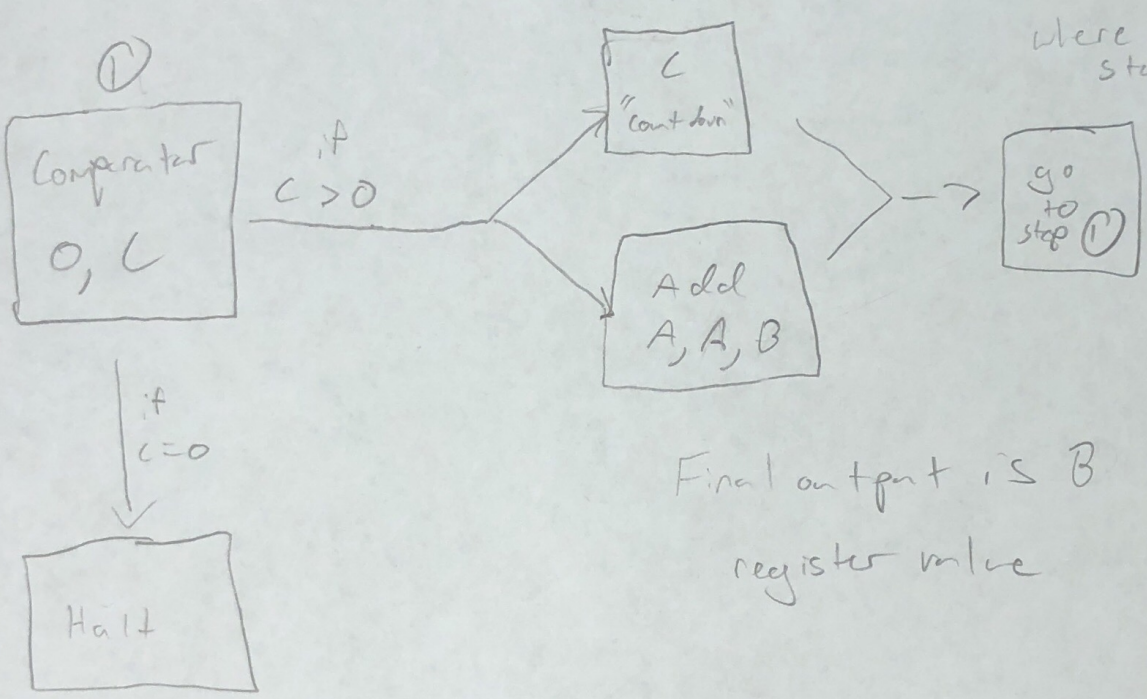
The sum values and last carry value, sign extended to fit a four bit result would be 1110 (-2 in decimal)

Multiplication by repeated addition

C register is register where counter value is stored (for multiplying, one of the operands is the counter)

operands: A + C

B is register where value is stored



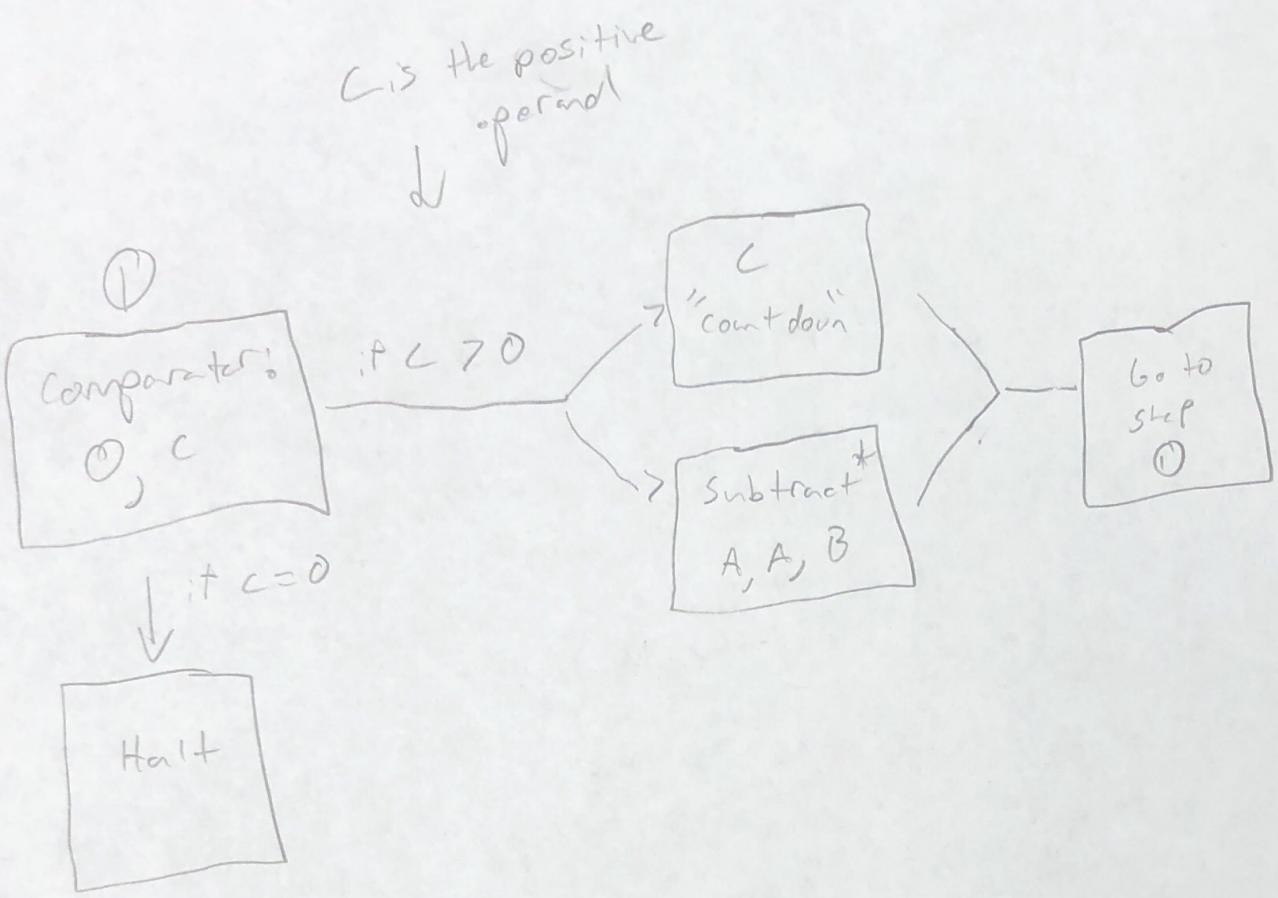
Final output is B register value

Blocks used:

- Comparator
- "count down" counter
- Adder

This works when A + B are same sign

For different sign, see back

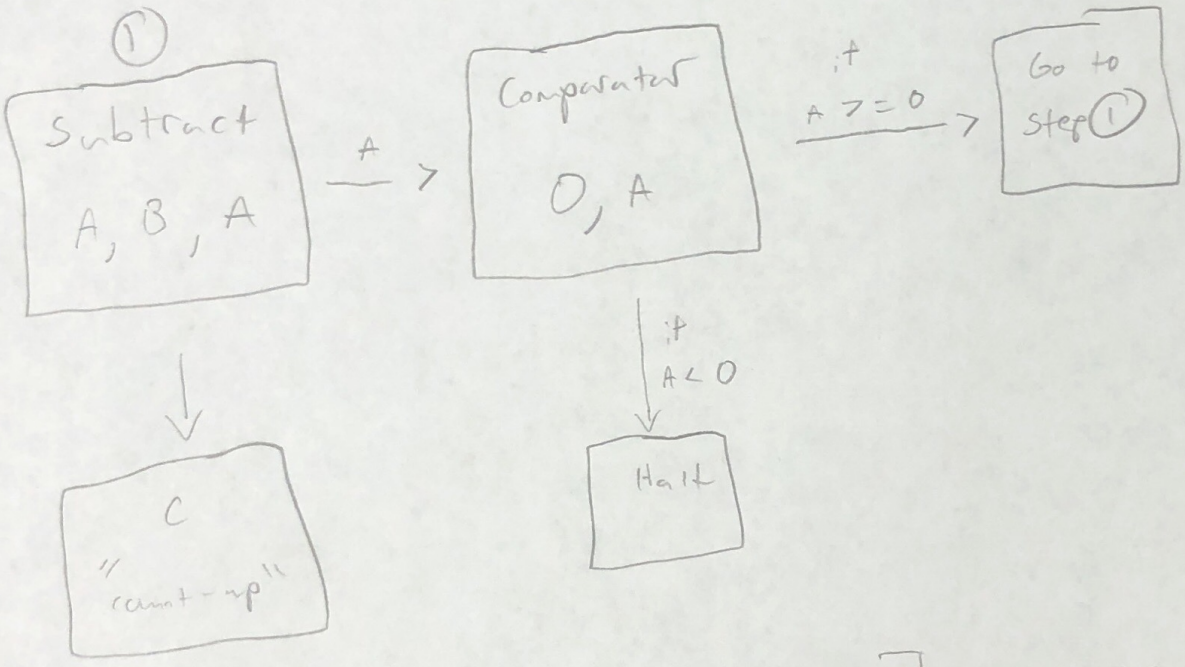


* A is the negative operand, unlike w/ same sign multiplication, we subtract the negative operand from itself repeatedly to get the final result

Division by repeated subtraction

Operands
A + B

C is register where count is stored + location of final output



final output,
# of subtractions

This works when A + B are same sign

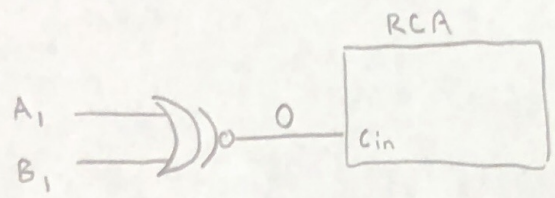
Different signs or back

Blocks used:
- Comparator
- "Count up" counter
- Subtractor

Assume we're dividing A_1A_0 by B_1B_0

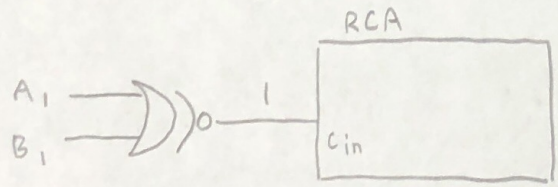
Case 1 The sign of operands are different

(checked by XNORing the sign bits and feeding the output as Cin to the adder subtractor)



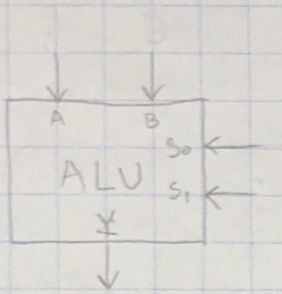
Therefore when the signs are different, perform repeated addition until you reach zero, then the result will be held in the upcounter

Case 2 The signs of operands are the same

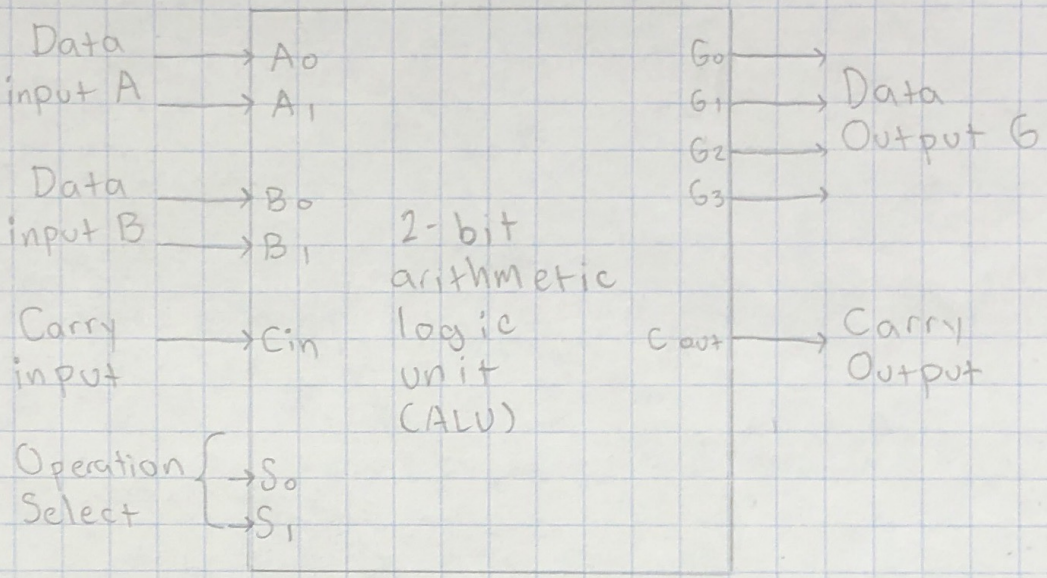


When the signs are the same, you get a 1 from the XNOR, so the adder/subtractor performs repeated subtraction, then the result will again be held in the upcounter

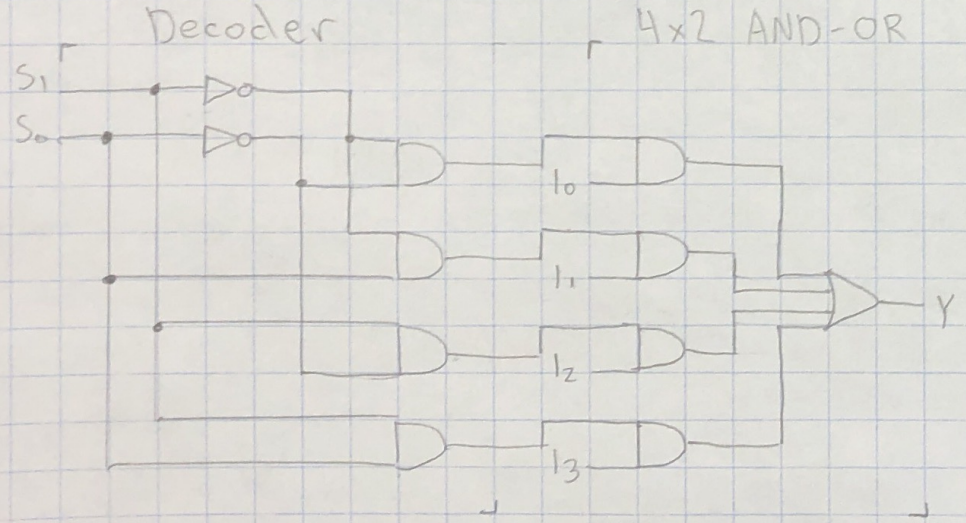
CS42 - Project - ALU Design (2-bit)



	S_1	S_0	Y
I_0	0	0	A+B
I_1	0	1	A-B
I_2	1	0	A×B
I_3	1	1	A/B

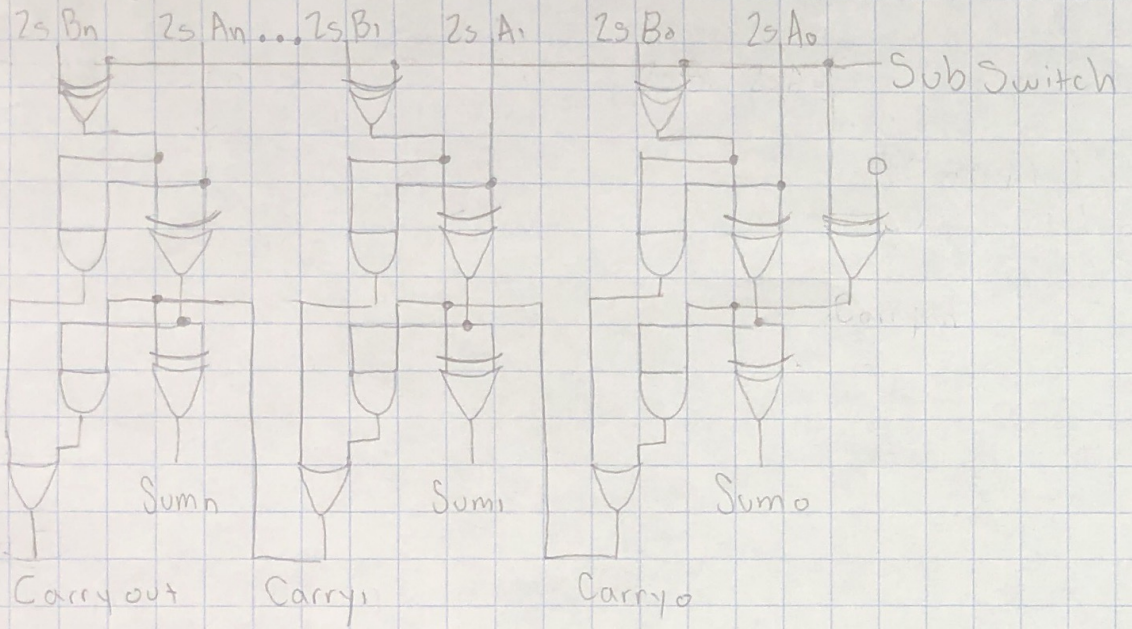


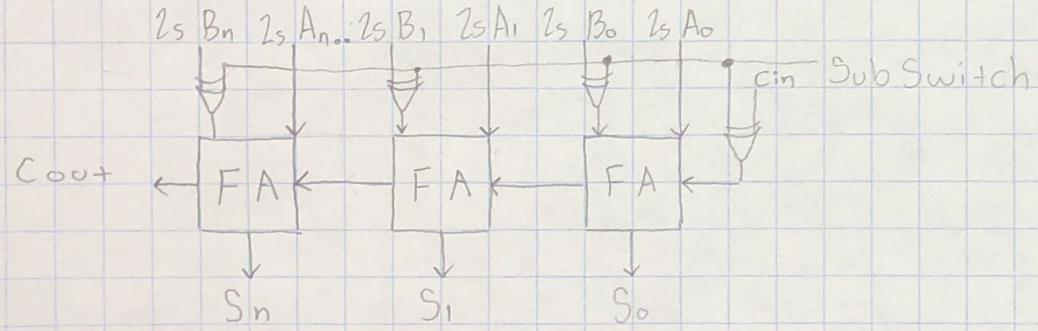
4-to-1-Line Multiplexer



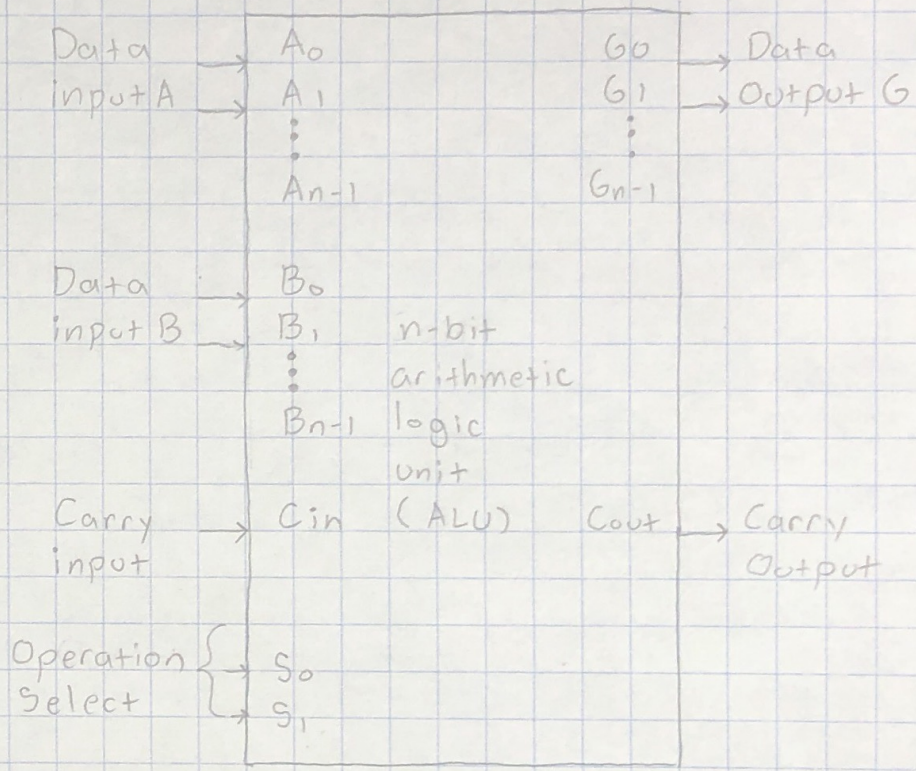
CS42 - Project - Arithmetic Circuit (n-bit) - ALU Design (n-bit)

Full-Adder/Subtractor w/ signed numbers in two's



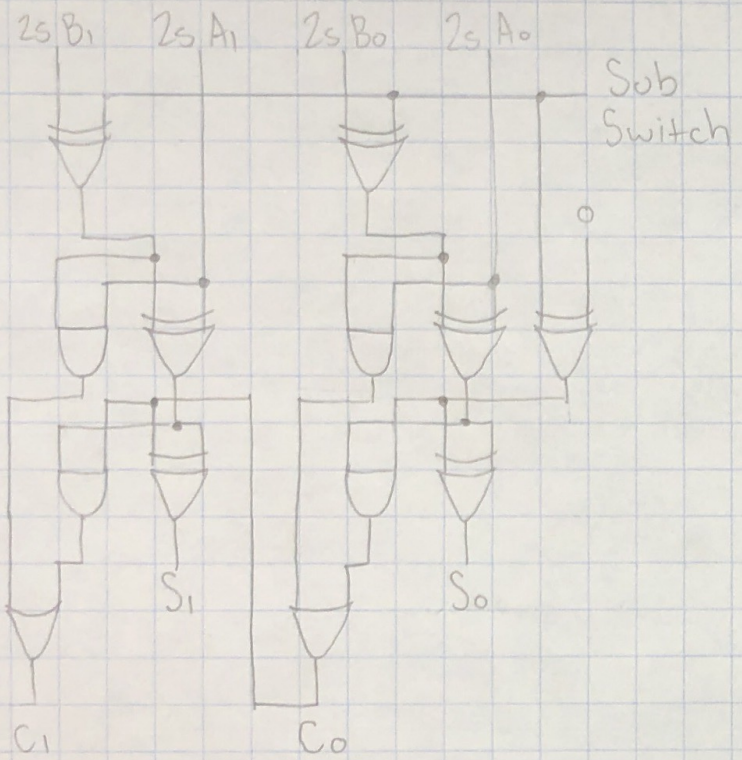


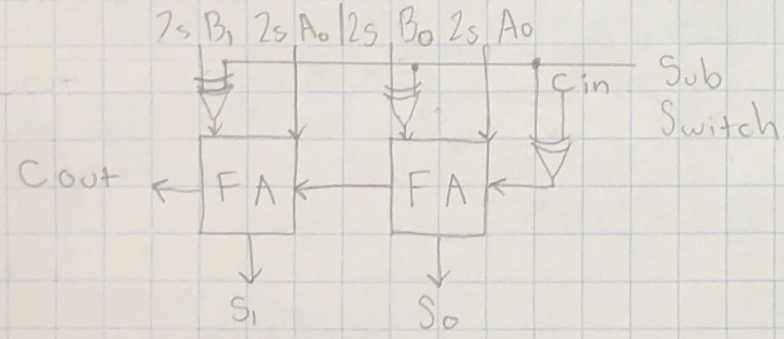
ALU



CS42 - Project - Arithmetic Circuits (2 bit)

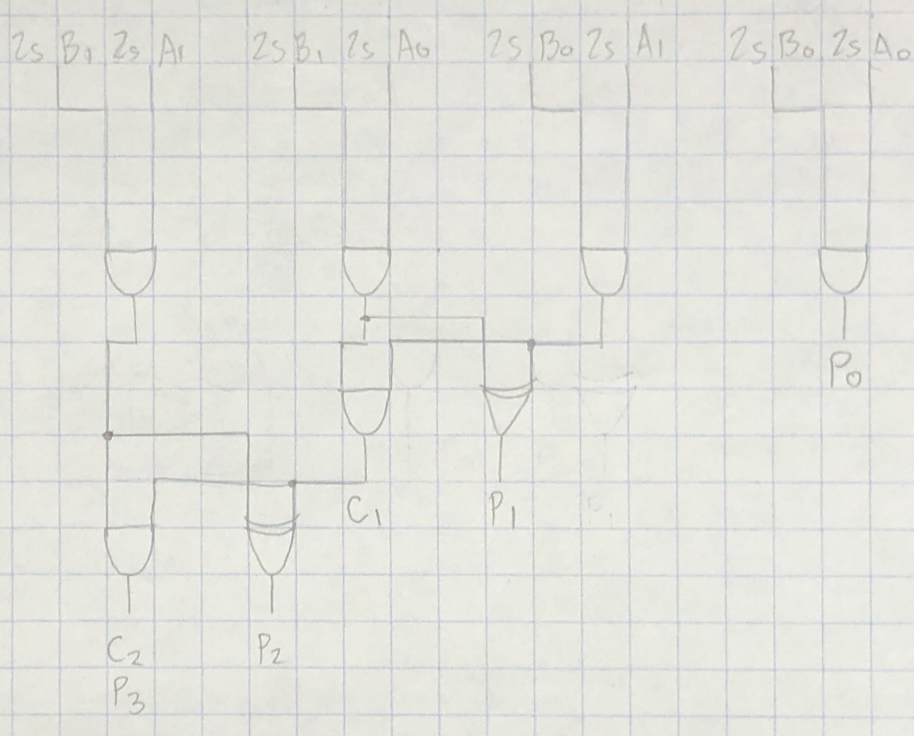
2-bit Full Adder w/Signed Numbers in 2's

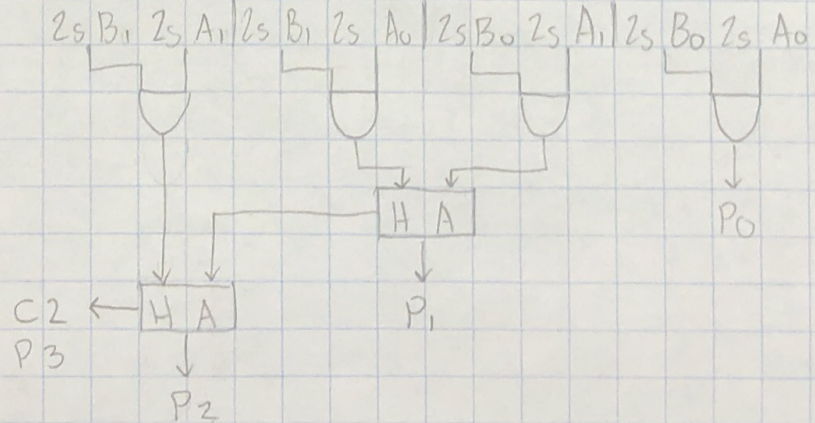




2-bit Multiplier (Boolean Algebra)

Not used in our design but an alternative



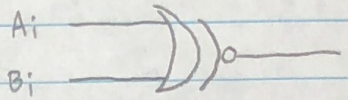


2 Bit Comparator

Assume we're comparing A_2A_1 vs. B_2B_1

$A_i = B_i$

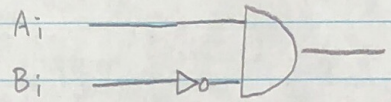
A_i	B_i	$A_i = B_i$
0	0	1
0	1	0
1	0	0
1	1	1



$= \overline{A_i}\,\overline{B_i} + A_iB_i = \overline{A_i \oplus B_i}$

$A_i > B_i$

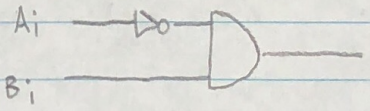
A_i	B_i	$A_i > B_i$
0	0	0
0	1	0
1	0	1
1	1	0



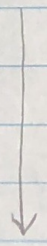
$= A_i\overline{B_i}$

$A_i < B_i$

A_i	B_i	$A_i < B_i$
0	0	0
0	1	1
1	0	0
1	1	0



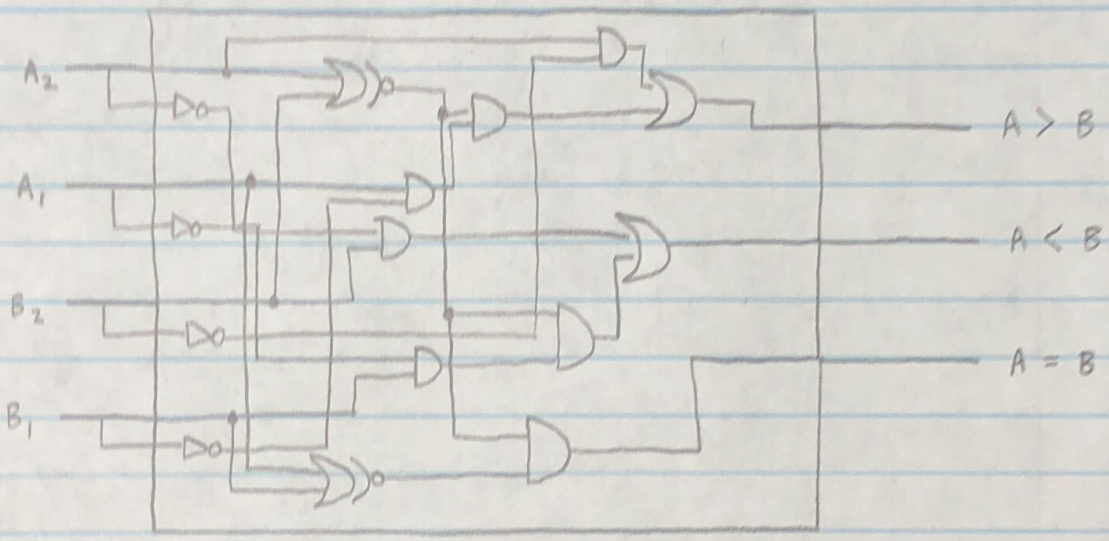
$= \overline{A_i}\,B_i$



$$A > B = A_2 \overline{B}_2 + \overline{(A_2 \oplus B_2)}\, A_1 \overline{B}_1$$

$$A < B = \overline{A}_2 B_2 + \overline{(A_2 \oplus B_2)}\, \overline{A}_1 B_1$$

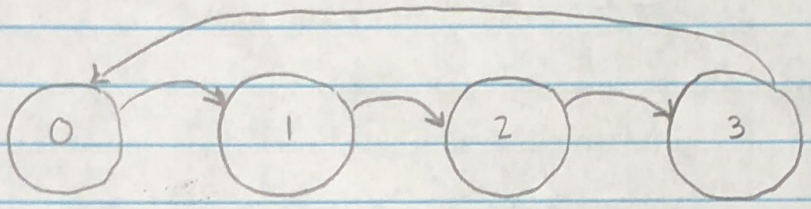
$$A = B = \overline{(A_2 \oplus B_2)}\ \overline{(A_1 \oplus B_1)}$$



Stewart Dulaney

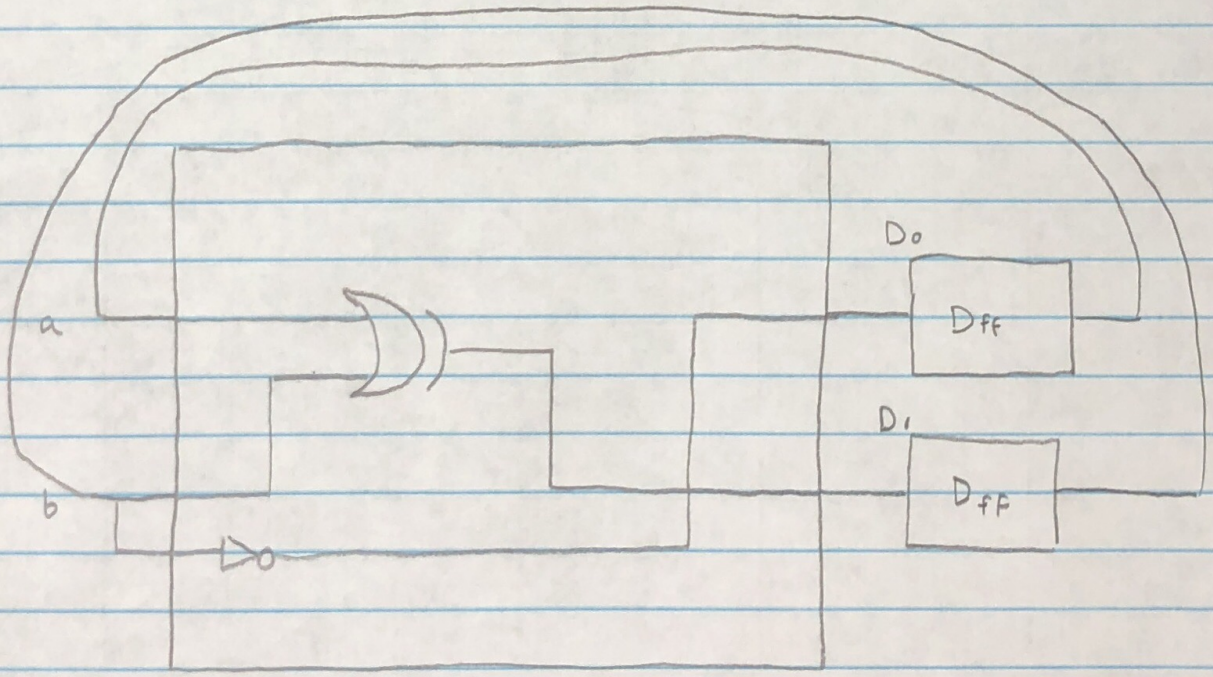
2 Bit Upcounter w/ D Flip Flops

	Current State a	b	Next State		D_1	D_0
m_0	0	0	0	1	0	1
m_1	0	1	1	0	1	0
m_2	1	0	1	1	1	1
m_3	1	1	0	0	0	0



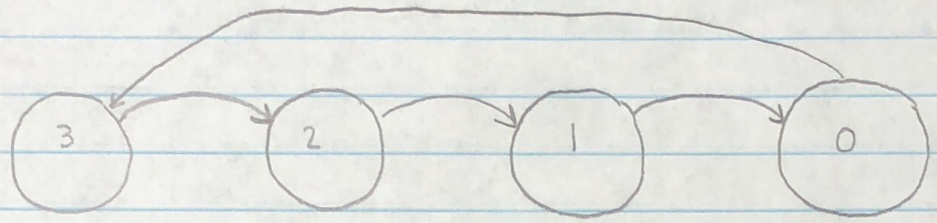
$$D_0 = m_0 + m_2 = \bar{a}\bar{b} + a\bar{b} = \bar{b}(\bar{a} + a) = \bar{b}$$

$$D_1 = m_1 + m_2 = \bar{a}b + a\bar{b} = a \oplus b$$



2 Bit Downcounter w/ D Flip Flops

	Current State a	b	Next State		D_1	D_0
m_0	1	1	1	0	1	0
m_1	1	0	0	1	0	1
m_2	0	1	0	0	0	0
m_3	0	0	1	1	1	1



$$D_0 = m_1 + m_3 = a\bar{b} + \bar{a}\bar{b} = \bar{b}(a + \bar{a}) = \bar{b}$$

$$D_1 = m_0 + m_3 = ab + \bar{a}\bar{b} = \overline{a \oplus b}$$