6 Sep 2023

$2^3 = 8$ $8 \cdot 8 = 64$

$64k = 2^6 \cdot 2^{10} = 2^{16}$ ↑ value

$2^{16} \cdot 2^3 = 2^{19}$

Question 1 (20 points)

a) (5 points) You have a SRAM memory chip with a capacity of 64k x 8

1) How many data lines does it have? *Answer 1)* .... 8 ....
2) How many address lines does it have? *Answer 2)* .... 16 ....
3) What is its capacity expressed in "*bits*"? *Answer 3)* $2^{19} = 2^9 k =$

→ 64K x 8

$2^{\# \text{ addresses OR Variables}} \times$ # bits OR # datalines OR # functions

Size of word OR

b) **15 points** Using D-type flip-flops, design a 3-bit Gray code counter which has the following counting sequence:

000 → 001 → 011 → 010 → 110 → 111 → 101 → 100

PS $Q_2Q_1Q_0$	NS $Q_2^+Q_1^+Q_0^+$	$D_2D_1D_0$
000	001	001
001	011	011
010	110	110
011	010	010
100	000	000
101	100	100
110	111	111
111	101	101

D ff

$D = Q_{t+1}$

$Q_2 \backslash Q_1Q_0$	00	01	11	10
0	0	0	0	1
1	0	1	1	1

$D_2 = Q_1Q_0' + Q_0Q_2$

Q_1 \ Q_1Q_0	00	01	11	10
0	0	1	1	1
1	0	0	0	1

$D_1 = Q_1Q_0' + Q_2'Q_0$

$D_2 = Q_1Q_0' + Q_0Q_2$

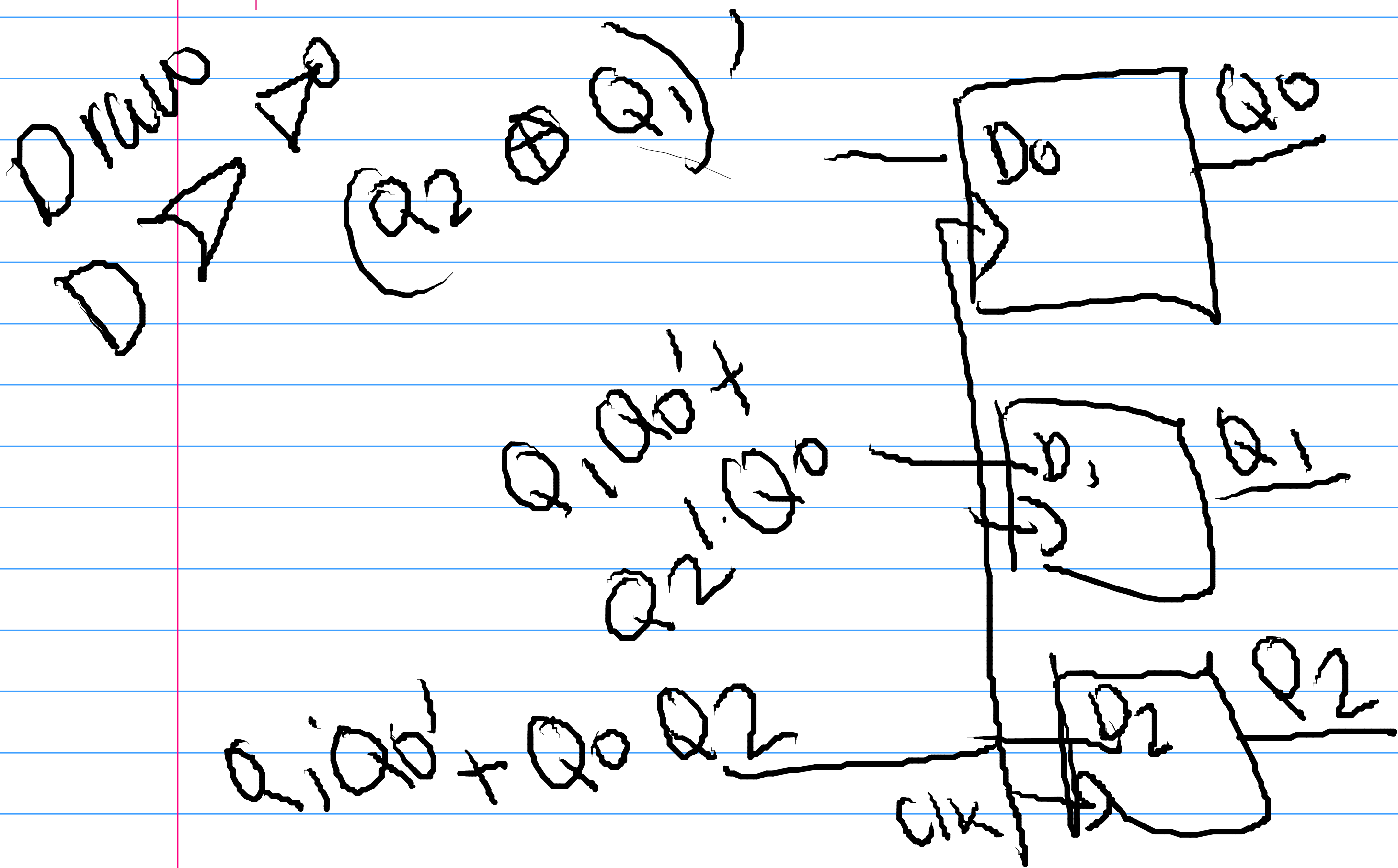
Q_2 \ Q_1Q_0	00	01	11	10
0	1	1	0	0
1	0	0	1	1

$D_0 = Q_2Q_1 + Q_2'Q_1'$

$= (Q_2 \oplus Q_1)'$

b) **15 points** Using D-type flip-flops, design a 3-bit Gray code counter which has the following counting sequence:

000 → 001 → 011 → 010 → 110 → 111 → 101 → 100
↑← ← ← ← ← ← ← ← ← ← ← ← ← ← ←↓



Question 2 (20 points) Design a 3-bit register whose function is described in the following table, where M and N are two control bits. Using the proper digital components (encoders, decoders, multiplexers, etc.) logic gates, and D flip-flops, draw a detailed diagram of the logic circuit of the register.

f^{MN}	M N	Operation	With counters	Next state	D_i
f^0	0 0	No change		$Q_2\ Q_1\ Q_0$	Q_i
f^1	0 1	Loading external inputs $I_2I_1I_0$		$I_2\ \ I_1\ \ I_0$	I_i
f^2	1 0	Decrement by 1 (count down)		$Q_2\ Q_1\ Q_0$ -1	?
f^3	1 1	Increment by 3 (count up)		$Q_2\ Q_1\ Q_0$+3	?

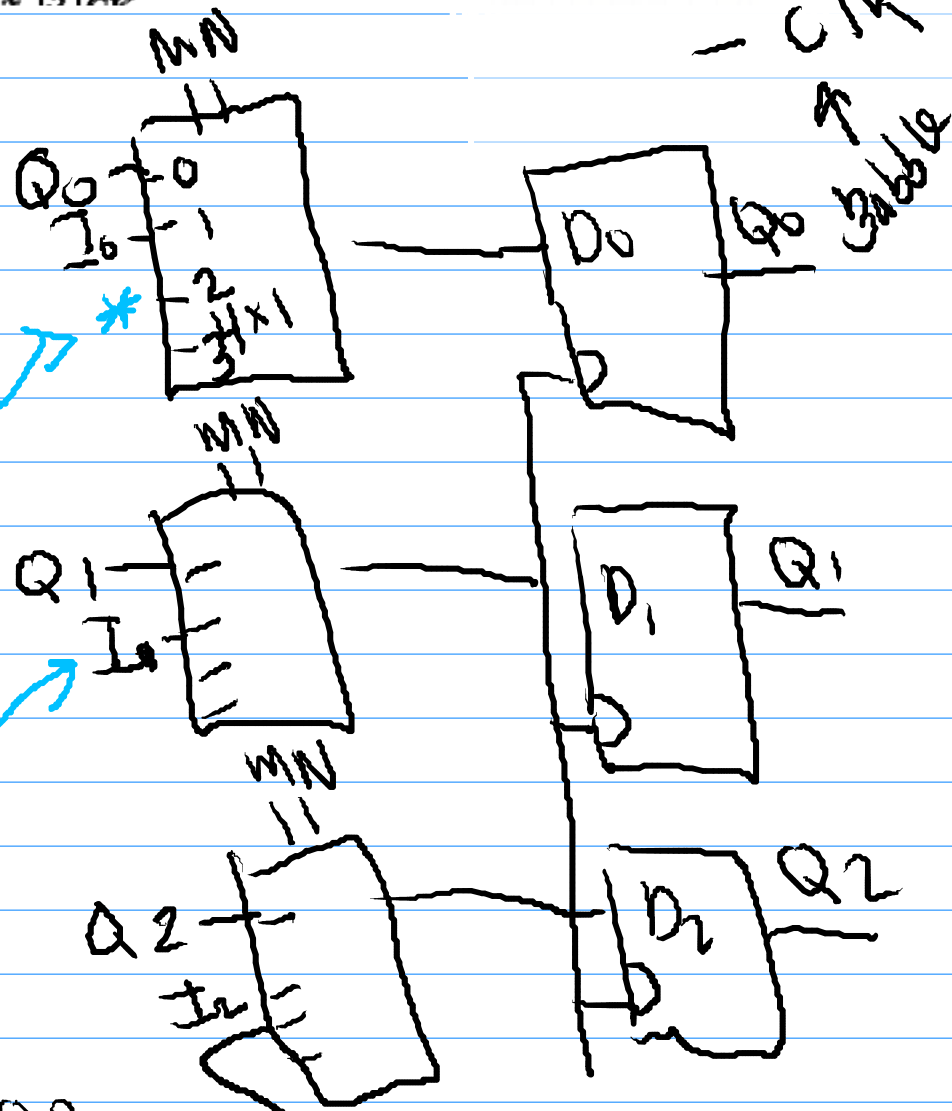
Using the D FF Excitation equation $D_i = Q_i^{n+1}$ for i=0,1,2

Muti fN

+ clk

− clk

0 0	No Change
0 1	I
1 0	Q −1
1 1	Q + 3



$Q_0 \rightarrow Q_0$

$Q_0 \rightarrow I_0$

EQN

kmap

Dec by 1

$Q_2 Q_1 Q_0$	$Q_2^+ Q_1^+ Q_0^+$
0 0 0	1 1 1
0 0 1	0 0 0
0 1 0	0 0 1
0 1 1	0 1 0
1 0 0	0 1 1
1 0 1	1 0 0
1 1 0	1 0 1

Q_2 \ Q_1Q_0	00	01	11	10
0	1	0	0	0
1	0	1	1	1

$D_2 = Q_2'Q_1'Q_0' + Q_2 Q_0 + Q_2 Q_1$

111 | 110

	00	01	11	10
0	1	0	1	0
1	1	0	0	1

$D_1 = Q_1'Q_0' + Q_1Q_0Q_2' + Q_2Q_0'$

PS $Q_2Q_1Q_0$	NS $Q_2^tQ_1^tQ_0^t$
0 0 0	0 1 1
0 0 1	1 0 0
0 1 0	1 0 1
0 1 1	1 1 0
1 0 0	1 1 1
1 0 1	0 0 0
1 1 0	0 0 1
1 1 1	0 1 0

+3

D's

→ kmap

→ EqN

→ link

③ on

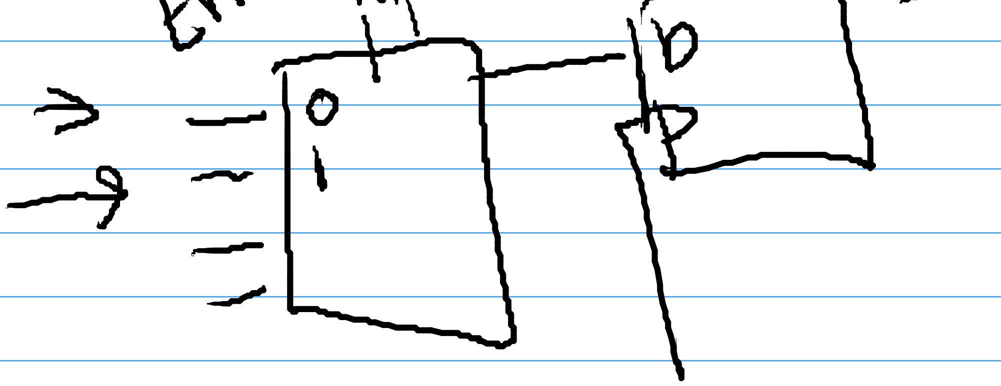
to

mux

ff → start

Enablers

MN
00
01



The 2's complement representation is used in an 8-bit register which contains the binary value 11011000.

a. What is the decimal value of the number stored initially in the register?

11011000 $\xrightarrow{\text{2's com}}$ 00101000 $= -(40)$

sign — neg

$2^5 \; 2^4 \; 2^3 \; 2^2 \; 2^1 \; 2^0$

$2^5 + 2^3 = 8 + 32$

b. What is the register value after an arithmetic shift right? Give your result both in binary and decimal.

L ⟶ R

11011000

→ 11101100

00010100 $\quad 2^2 + 2^4 \quad 4+16 \quad -(20)$

Ar RS

take sign bit

add to left

c. Starting again from the initial number 11011000, determine the register value after an arithmetic shift left, both in binary and decimal.

→ 11011000

L ⟵ R

sign → 10110000 → $-(80)$

2's → 01010000

Ar LS

add Ø to Righ

& shift

d. What arithmetic operations are performed by these shifts?

Ar L Shift $\times 2$

Ar R Shift $\div 2$

e. Is there any overflow?

Not in this case

talk only in range

$[-2^{N-1}, 2^{N-1}-1]$

$[-2^7, 2^7-1]$

fails in

if (Sign X = Sign Y) ≠ Sign S → overflow

FA takes 3 input

1's comp +1 → 2's comp (neg)

xy \ z	0	1
0 0	F = A + B (add)	F = A + B + 1
0 1	F = A + B'	F = A - B (subtract)
1 0	F = A (transfer)	F = A + 1 (increment)
1 1	F = A - 1 (decrement)	F = A' + 1 (2's complement of A)

x y z	F	P Q
0 0 0	A+B	$A_2A_1A_0 + B_2B_1B_0 + 0$
0 0 1	A+B+1	$A_2A_1A_0 + B_2B_1B_0 + 1$
0 1 0	A+B'	$A_2A_1A_0 + B_2'B_1'B_0' + 0$
0 1 1	A−B	$A_2A_1A_0 + B_2'B_1'B_0' + 1$
1 0 0	A	$A_2A_1A_0 + 000 + 0$
1 0 1	A+1	$A_2A_1A_0 + 000 + 1$
1 1 0	A−1	$A_2A_1A_0 + 111 + 0$
1 1 1	A'+1	$A_2'A_1'A_0' + 000 + 1$

001 → 111

111

x y

11 → 3

11

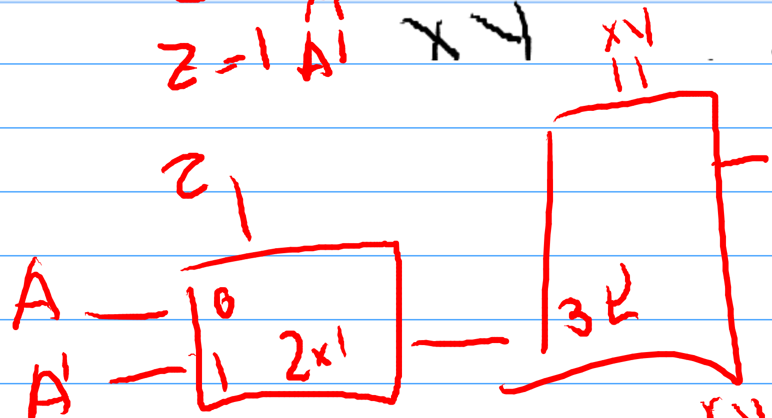
xyz	F	P + Q + c_{in}
000	A+B	$A_2A_1A_0 + B_2B_1B_0 + 0$
001	A+B+1	$A_2A_1A_0 + B_2B_1B_0 + 1$
010	A+B'	$A_2A_1A_0 + B_2'B_1'B_0' +$
011	A−B	$A_2A_1A_0 + B_2'B_1'B_0' +$
100	A	$A_2A_1A_0 + 000 + 0$
101	A+1	$A_2A_1A_0 + 000 + 1$
110	A−1	$A_2A_1A_0 + 111 + 0$
111	A'+1	$A_2'A_1'A_0' + 000 + 1$

Z=0 A

Z=1 A'

xy

why 2×1 mux

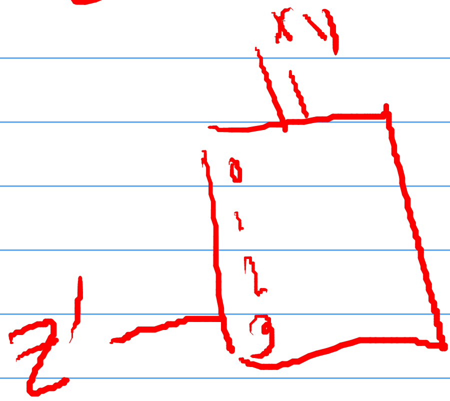


x y z Add by

1 1 0 → A

1 1 1 → A'

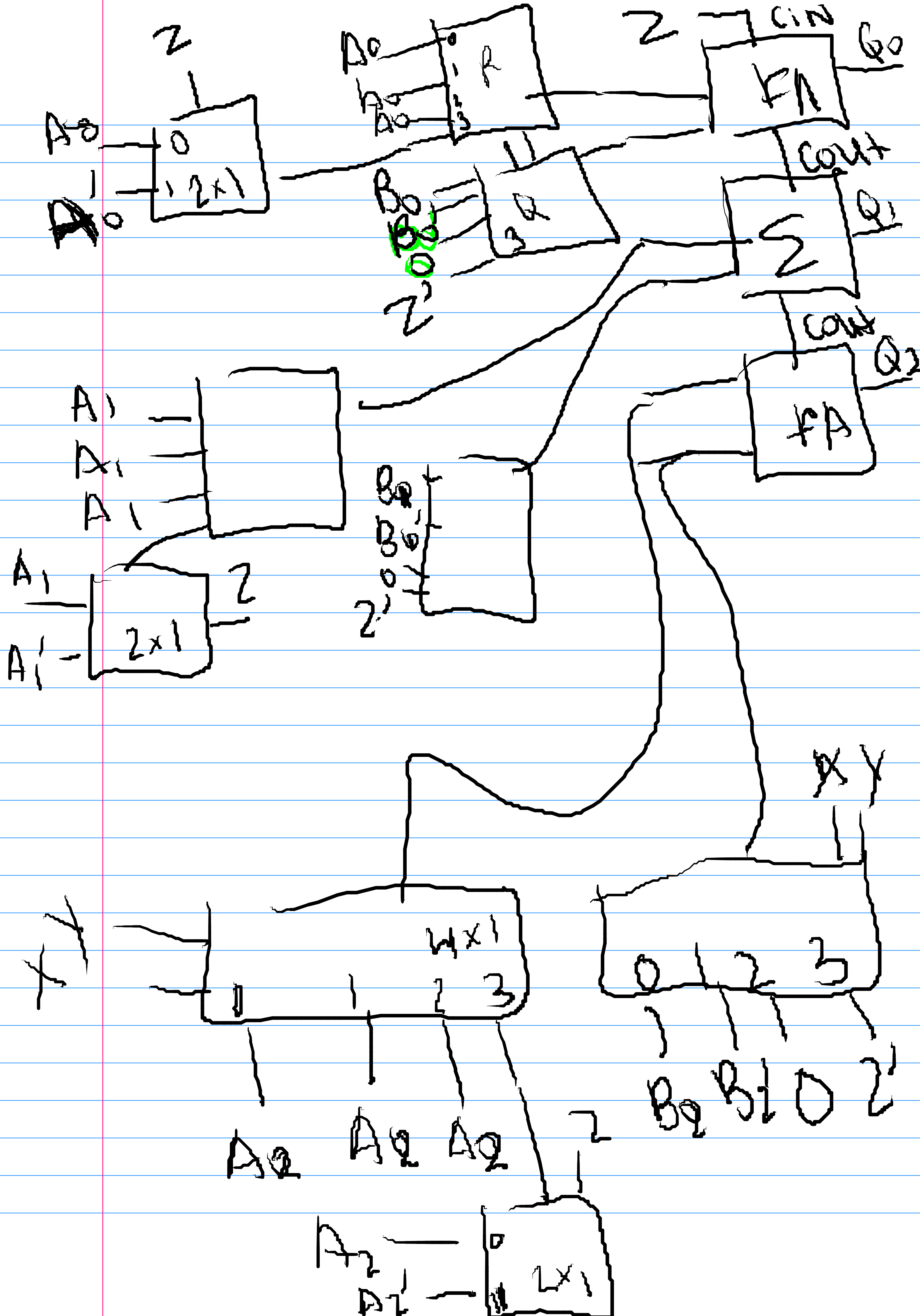
Why z'



x y z Add by

1 1 0 +1

1 1 1 +0



2
2x1
FA
Cin
Cout
Q0
Q1
Q2
4x1
X Y

Clock	M	N	Operation
↑	0	0	Store current value / No change
↑	0	1	Decrement by 3
↑	1	0	Load external inputs (I_2 I_1 I_0)
↑	1	1	Increment by 2

Design a synchronous modulo-4 binary counter that has one control input *W* and that counts forward or backward, as follows (assume an initial count value of 0):

- if *W*=0: counts backward by one (0→3→2→1→0…) at each clock pulse
- if *W*=1: counts forward by one (0→1→2→3→0…) at each clock pulse

x y	*z* = 0	*z* = 1
0 0	F = A’ + 1 (2’s complement)	F = A – 1 (decrement)
0 1	F = A – B (subtract)	F = A – B + 1 (subtract with borrow)
1 0	F = A + 1 (increment)	F = A (transfer)
1 1	F = A + B + 1 (add with carry)	F = A + B (add)

encoders, decoders, multiplexers, etc.) (Note: A' is the 1's complement

x y	***z* = 0**	***z* = 1**
0 0	F = A + B (add)	F = A + B + 1
0 1	F = A + B'	F = A - B (subtract)
1 0	F = A (transfer)	F = A + 1 (increment)
1 1	F = A - 1 (decrement)	F = A' + 1 (2's complement)

S_2	S_1	S_0	Operation	Description
0	0	0	F← A + B	Addition
0	0	1	F← A - 3	Decrement by 3
0	1	0	F← A + 1	Increment by 1
0	1	1	F← A - B (i.e., A + B'+1)	Subtraction
1	0	0	F← ashl A	Arithmetic shift - left
1	0	1	F← cshr A	Circular shift - right
1	1	0	F← (A ∧ B)'	NAND
1	1	1	F← A ⊕ B	Exclusive-OR

Arth

$S_2S_1S_0$	Function	
0 0 0	A + B	$A_2A_1A_0 + B_2B_1B_0 + 0$
0 0 1	A - 3	$A_2A_1A_0 + 101 + 0$
0 1 0	A + 1	$A_2A_1A_0 + 000 + 1$
0 1 1	A - B	$A_2A_1A_0 + B_2'B_1'B_0' + 1$

Subtract → 2's comp

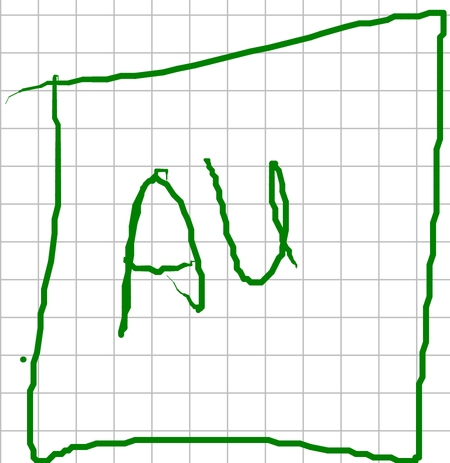
3 011
101

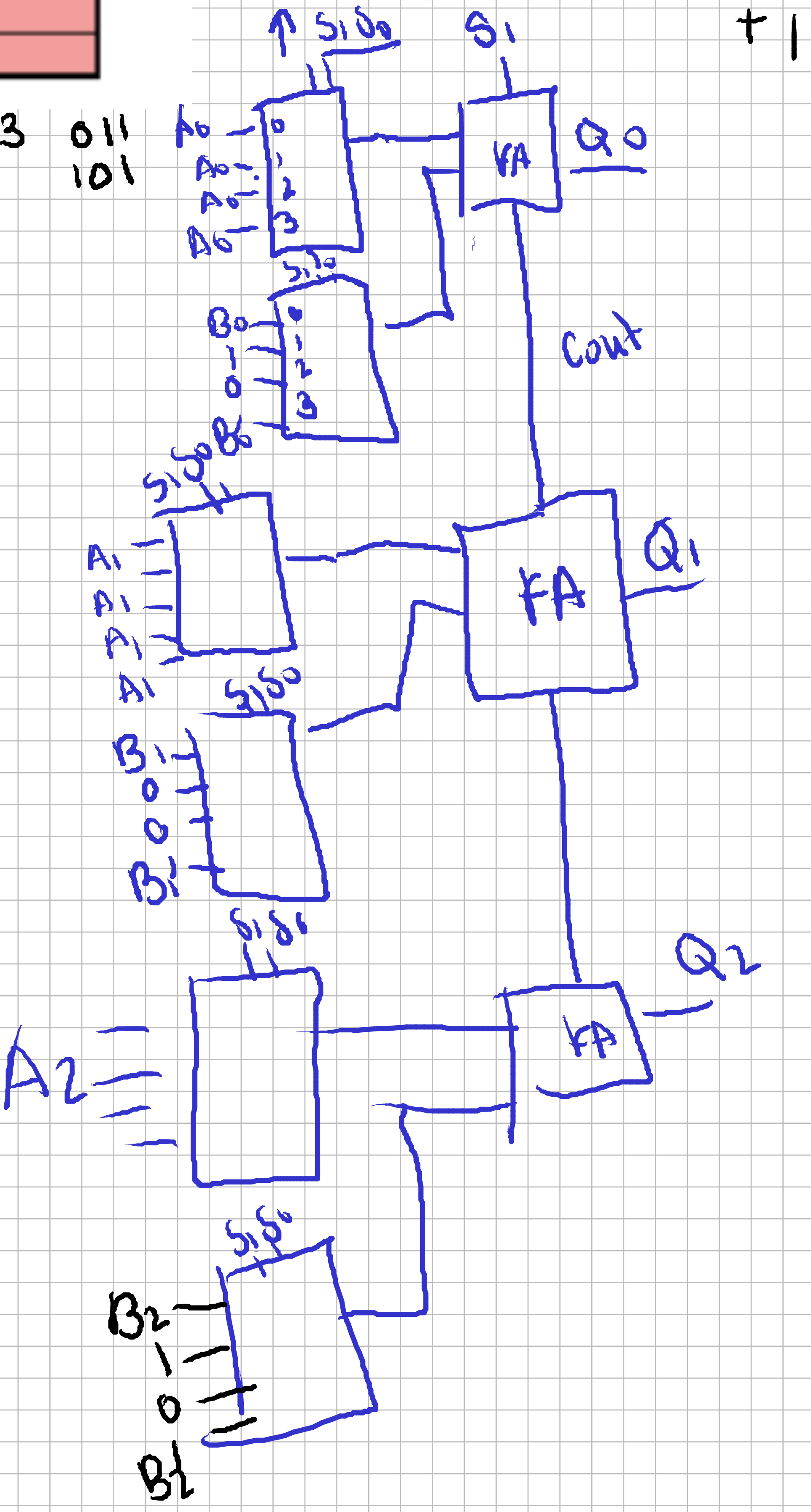
tice relationship

w Z

OR
1's comp + 1

AU Done !!)



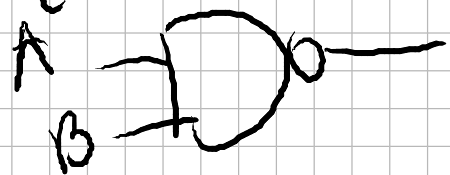


S_2	S_1	S_0	Operation	Description
0	0	0	F← A + B	Addition
0	0	1	F← A - 3	Decrement by 3
0	1	0	F← A + 1	Increment by 1
0	1	1	F← A - B (i.e., A + B'+1)	Subtraction
1	0	0	F← ashl A	Arithmetic shift - left
1	0	1	F← cshr A	Circular shift - right
1	1	0	F← (A ∧ B)'	NAND
1	1	1	F← A ⊕ B	Exclusive-OR

$S_2 S_1 S_0$	
1 0 0	ashL
1 0 1	crR
1 1 0	$(A \wedge B)'$
1 1 1	$A \oplus B$

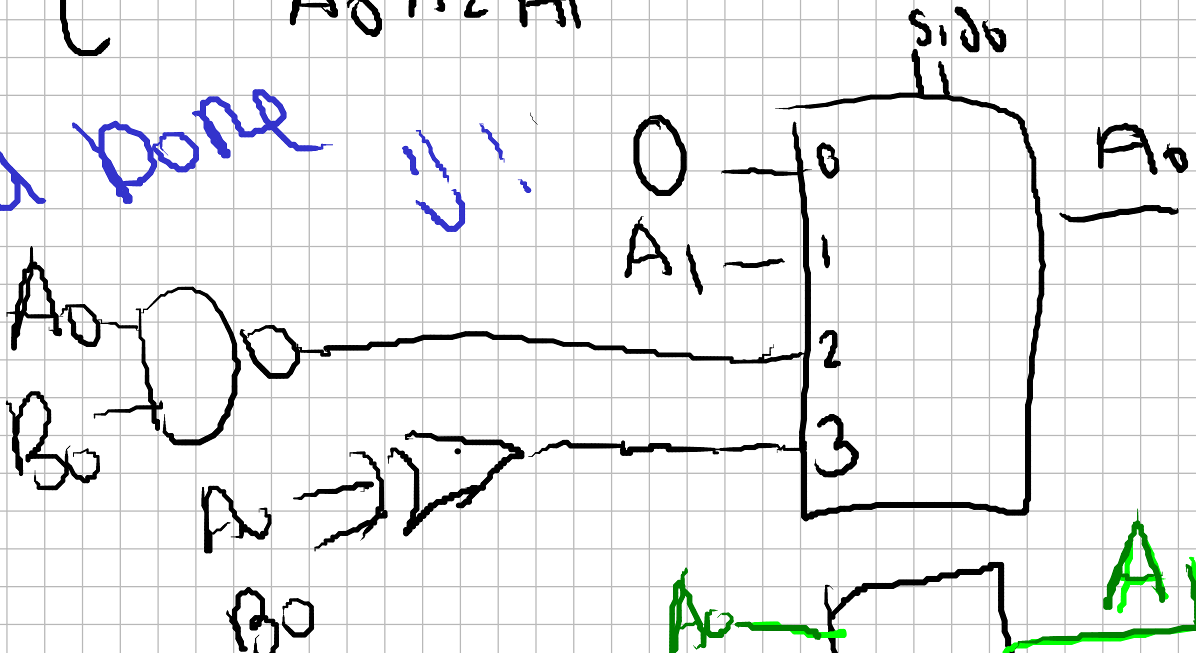
FN1 [L $A_2 A_1 A_0$ ← $A_1 A_0 0$ R

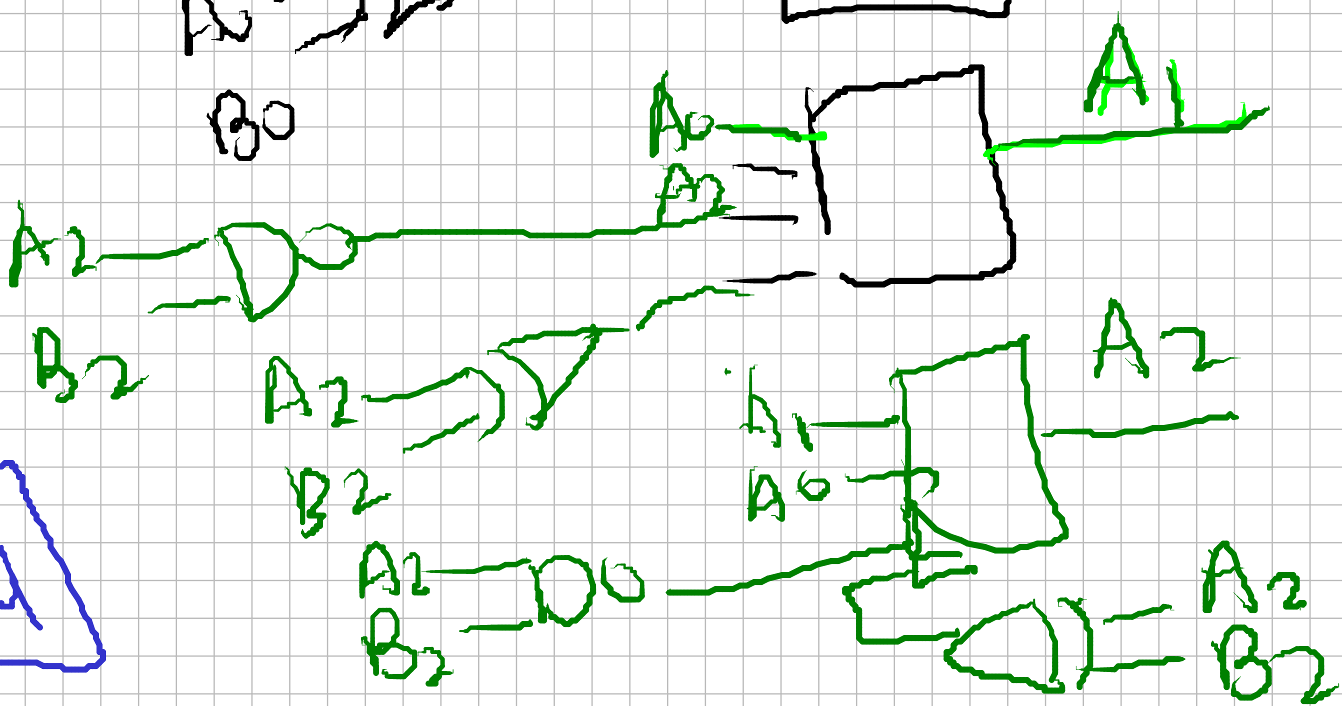
$(A \cap B)'$

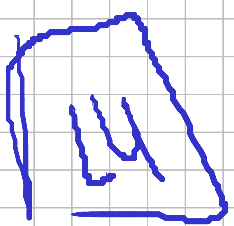


FN2 [R L $A_2 A_1 A_0$ → $A_0 A_2 A_1$ R

LU Done !!







ALU

1 unit
1 bit