EE 209 “Don’t Make the Same Mistake”

Redekopp

No calculators are allowed. Show all your work to get full credit.

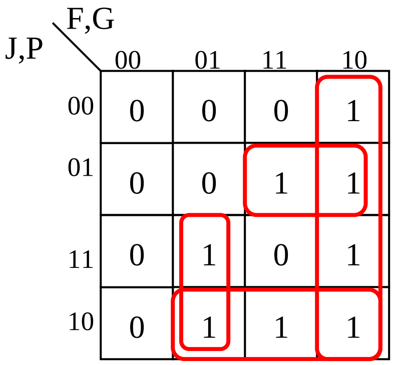
Note: The following are all INCORRECT answers meant to highlight common misunderstandings! Try to figure out where there answer came from and the misconception that caused it. Correct answers are intentionally not given and will not be so that you think about how to solve the problem correctly.

1. (17 pts.) Bob is throwing a party for his friends: Fred, Greg, Jill and Peter. He can only have the party if at least 2 of his friends attend. He invites them and his friends first check if they are free (able to come). However, some of his friends are also particular about who they associate with. Their conditions for attending are listed below.

a) Fred will only attend if at least one boy **besides** Bob is there.
b) Greg will only attend if Jill does.
c) Peter and Jill love parties and will attend w/o restriction if they are free.

For the different combinations of Bob’s friends (1 = free, 0 = busy), indicate which combinations will allow the party to occur (at least 2 attendees even with the restrictions a-c. Then find the minimal **SOP** expression for the party to occur.

F	G	J	P	PARTY
0	0	0	0	0
0	0	0	1	0
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	



J,P \ F,G	00	01	11	10
00	0	0	0	1
01	0	0	1	1
11	0	1	0	1
10	0	1	1	1

- Wrong 1 / 0 locations.
- Wrong grouping even with the given 1/0 locations

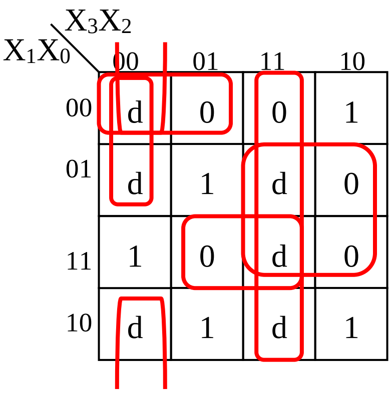
2. Consider a circuit that takes as input a single decimal digit, X[3:0], represented in **Excess-3 <u>decimal</u>** code, (X3X2X1X0), and produces an output P=1 indicating the number is prime (i.e. 2,3,5,…).

a) Fill in the truth table using don’t care’s where appropriate. (8 pts.)

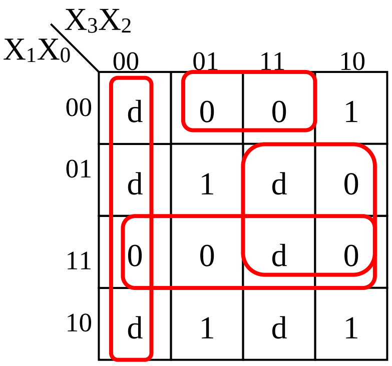
b) Use the K-Map’s below to find a minimal 2-level **POS** implementation for P. (10 pts.)

Recall, X[3:0] should be interpreted as an <u>Excess-3 decimal code</u> value.

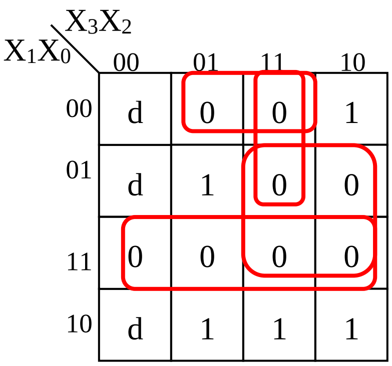
X3	X2	X1	X0	P	(Wrong answer 1)	Wrong answer 2
0	0	0	0		d	0
0	0	0	1		d	0
0	0	1	0		d	0
0	0	1	1		0	0
0	1	0	0		0	0
0	1	0	1		1	1
0	1	1	0		1	1
0	1	1	1		0	0
1	0	0	0		1	1
1	0	0	1		0	0
1	0	1	0		1	1
1	0	1	1		0	0
1	1	0	0		0	0
1	1	0	1		0	0
1	1	1	0		1	1
1	1	1	1		0	0



X_1X_0 \ X_3X_2	00	01	11	10
00	d	0	0	1
01	d	1	d	0
11	1	0	d	0
10	d	1	d	1

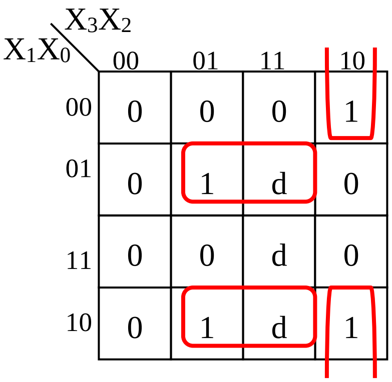


X_1X_0 \ X_3X_2	00	01	11	10
00	d	0	0	1
01	d	1	d	0
11	0	0	d	0
10	d	1	d	1



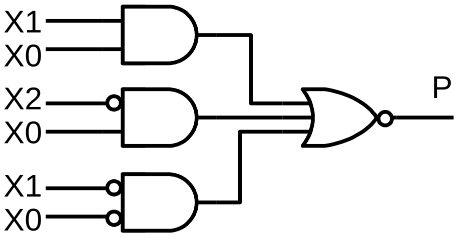
X_1X_0 \ X_3X_2	00	01	11	10
00	d	0	0	1
01	d	1	0	0
11	0	0	0	0
10	d	1	1	1

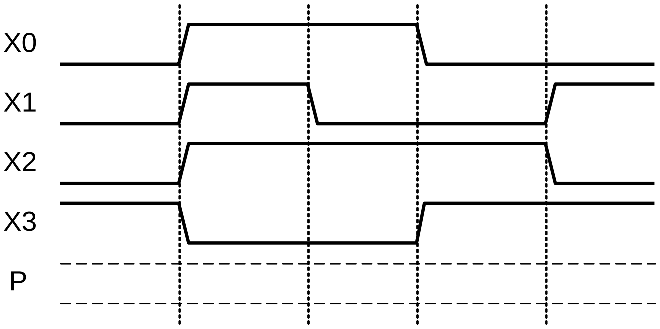
- Wrong K-Map groupings



X_1X_0 \ X_3X_2	00	01	11	10
00	0	0	0	1
01	0	1	d	0
11	0	0	d	0
10	0	1	d	1

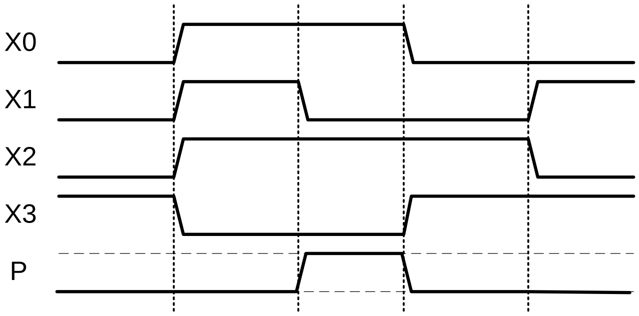
c) Billy Bruin said he could implement the same function using an AND-NOR implementation. He arrived at the circuit below. However, he made a mistake in his design and the circuit produces the **WRONG** value of P for **one** (*and only one*) input combination. Complete the waveform for P and determine which input combination of X[3:0] causes the error. (i.e. For what combo does P not match your expectation) (12 pts.)





Error-Producing Bit Combination {X3,X2,X1,X0} = ______________ (e.g. 1001)

Wrong Waveform and combination:



Error-Producing Bit Combination {X3,X2,X1,X0} = ___1010______ (e.g. 1001)

3. a.) (12 pts.) Using your Boolean algebra theorems, convert the following equation to a **minimal** POS form. (Circle that answer) Indicate which theorem(s) you are using in each step.

b.) **(16 pts.)** Only after it is in minimal POS form, expand the expression to maxterms and the canonical product, **showing your intermediate steps**. You do NOT need to show which theorems you are using. Write the canonical product in **both** *algebraic* form and ∏ *notation*. (Note: You may **NOT** find the canonical sum and then just write out the canonical product.)

$$G = (A\overline{B} + CB)(\overline{A} + C)$$

a.) Show work here:

- Wrong answer 1
  = C(B’+C)(A+B)
- Wrong answer 2
  (multiple mistakes)
  = AB’(A’+C) + CB(A’+C)
  = AB’A’ + AB’C + A’BC + BC
  = B’ + AB’C + A’BC + BC…
  = B(A’+B+C’)(A+B’+C’)(B’+C’) [DeMorgan’s]
- Wrong answer 3
  = AA’B’+AB’C + A’BC + CCB
  = AB’C + A’BC + CB
  = AB’C = A’
  = (A’+A)(A’+B’)(A’+C)
  = (A’+B’)(A’+C)
- Wrong answer 4
  = (AB’+CB)(A’+C)
  = (A’+B)(B’+C’)(A’+C) [DeMorgan’s]
- Wrong answer 5
  = AB’A’ + AB’C + A’BC + BCC
  = 0 + ABC + A’BC + BC
  = C(AB’ + A’B + B)
  = C(AB’ +B)
  = AB’C + BC
- Wrong answer 6
  Minimal$_{POS}$ = C(A+B)(A+C)(C+B)(B’+C)
- Wrong answer 7
  Minimal$_{POS}$ = C(A+B)(B’+B)

b.) Show work here:

- Wrong answer 1
  (starting here…)

$$= (A'+B'+C)(A'+B'+C')(A'+B+C)(A'+B'+C)$$
$$= (A'+B'+C)(A'+B'+C')(A'+B+C)$$
$$= \Pi_{ABC}(0,1,3)$$

- Wrong answer 2

$$= AB'C + BC$$
$$= AB'C + (A'A)BC$$
$$= AB'C + A'BC + ABC$$
$$= \Pi_{ABC}(0,2,4)$$

Single-Variable Theorems				
(T1)	$X + 0 = X$	(T1')	$X \bullet 1 = X$	(Identities)
(T2)	$X + 1 = 1$	(T2')	$X \bullet 0 = 0$	(Null elements)
(T3)	$X + X = X$	(T3')	$X \bullet X = X$	(Idempotency)
(T4)	$(X')' = X$			(Involution)
(T5)	$X + X' = 1$	(T5')	$X \bullet X' = 0$	(Complement)
Two- and Three-Variable Theorems				
(T6)	$X + Y = Y + X$	(T6')	$X \bullet Y = Y \bullet X$	(Commutativity)
(T7)	$(X+Y)+Z = X+(Y+Z)$	(T7')	$(X\bullet Y) \bullet Z = X\bullet (Y\bullet Z)$	(Associativity)
(T8)	$X\bullet(Y+Z) = X\bullet Y + X\bullet Z$	(T8')	$X+(Y\bullet Z) = (X+Y) \bullet (X+Z)$	(Distributivity)
(T9)	$X + X\bullet Y = X$	(T9')	$X \bullet (X + Y) = X$	(Covering)
(T10)	$X\bullet Y + X\bullet Y' = X$	(T10')	$(X+Y) \bullet (X+Y') = X$	(Combining)
(T11)	$X\bullet Y+X'\bullet Z+Y\bullet Z = X\bullet Y+X'Z$	(T11')	$(X+Y)\bullet(X'+Z)\bullet(Y+Z) = (X+Y)\bullet(X'+Z)$	(Consensus)
DeMorgan's Theorem				
	$(X \bullet Y)' = X' + Y'$	(T6')	$(X + Y)' = X' \bullet Y'$	(DeMorgan's)

4. Use Boolean algebra to find the simplest **SOP** or **POS** form of the following logic equation. For major steps, indicate which theorems you are using. Circle your answer. (8 pts.)

$$F = (\bar{A} + (\overline{A + C}) + \bar{B}) \cdot \overline{((\bar{A} + \bar{C} + AC)\bar{B}D)}$$

= (A’+(A’C’)+B’) · ((A’+C’+AC)’ + (B’D)’)

= (A’+(A’C’)+B’) · ((ACA’+C’) + (B+D’))

= …

= A’+ A’C’ + B’ · (AC(A’+C’) + (B+D’))

= A’(1+C’) + B’ · (ACA’+ACC’ + (B+D’))

= A’ + B’ · (B+D’)

= A’ + B’B + B’D’

= A’ + B’D’

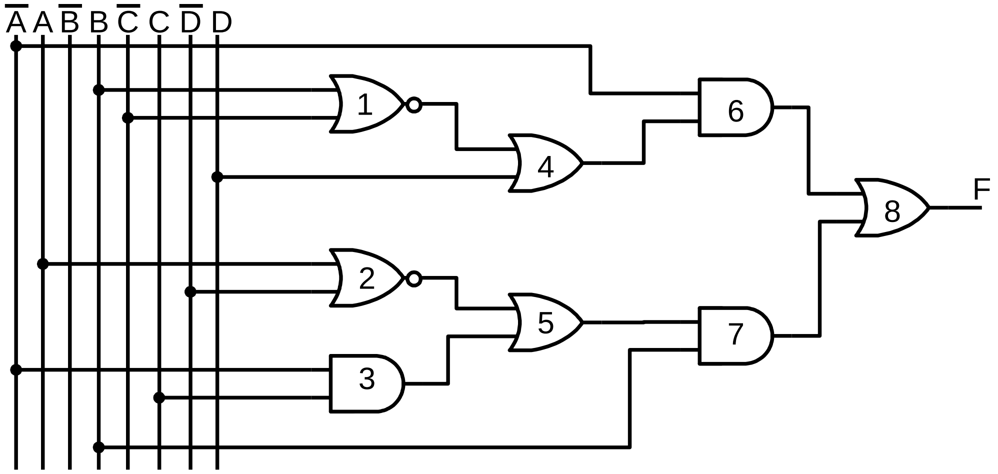
= (A’+B’)(A’+D’)

Single-Variable Theorems				
(T1)	X + 0 = X	(T1’)	X • 1 = X	(Identities)
(T2)	X + 1 = 1	(T2’)	X • 0 = 0	(Null elements)
(T3)	X + X = X	(T3’)	X • X = X	(Idempotency)
(T4)	(X’)’ = X			(Involution)
(T5)	X + X’ = 1	(T5’)	X • X’ = 0	(Complement)
Two- and Three-Variable Theorems				
(T6)	X +Y = Y + X	(T6’)	X • Y = Y • X	(Commutativity)
(T7)	(X+Y)+Z = X+(Y+Z)	(T7’)	(X•Y) •Z = X• (Y•Z)	(Associativity)
(T8)	X•(Y+Z) = X•Y + X•Z	(T8’)	X+(Y•Z) = (X+Y) • (X+Z)	(Distributivity)
(T9)	X + X•Y = X	(T9’)	X • (X + Y) = X	(Covering)
(T10)	X•Y + X•Y’ = X	(T10’)	(X+Y) • (X+Y’) = X	(Combining)
(T11)	X•Y+X’•Z+Y•Z = X•Y+X’Z	(T11’)	(X+Y)•(X’+Z)•(Y+Z) = (X+Y)•(X’+Z)	(Consensus)
DeMorgan’s Theorem				
	(X • Y)’ = X’ + Y’	(T6’)	(X +Y)’ = X’ • Y’	(DeMorgan’s)

5. **(25 pts.)** Tim needs to construct the function, F, shown below but he does not have a lot of money. Tim was offered a great deal on NAND gates: **5 cents each**. All other gates cost him **10 cents each**. Assuming he can reconnect the 1st level gate inputs to either their true or inverted form, show how Tim achieved the cheapest implementation of F.

In other words, without changing the function, which gates can be **REPLACED WITH A NAND** or need to be **KEPT AS IS**. Also, for each input to the 1st level gates, indicate whether the true or inverted form should be used for your new circuit implementation.

Original Circuit:

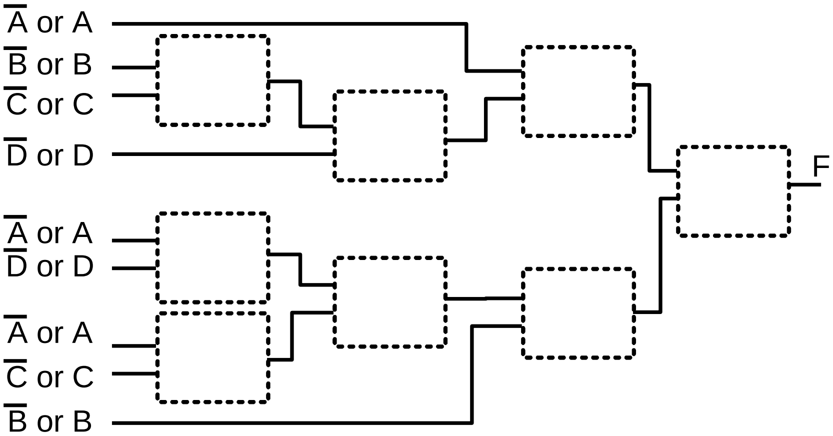


$\overline{A}$ A $\overline{B}$ B $\overline{C}$ C $\overline{D}$ D
1
2
3
4
5
6
7
8
F

Mistaken Answer:

Cheapest Implementation:

Circle either the variable in its true or inverted form as necessary →



Leave all connections as is.

Circle the desired gate replacements:

Gate 1	[✓ Keep as is] / Replace with NAND
Gate 2	[✓ Keep as is] / Replace with NAND
Gate 3	[✓ Keep as is] / Replace with NAND
Gate 4	[✓ Keep as is] / Replace with NAND
Gate 5	[✓ Keep as is] / Replace with NAND
Gate 6	Keep as is / [✓ Replace with NAND]
Gate 7	Keep as is / [✓ Replace with NAND]
Gate 8	Keep as is / [✓ Replace with NAND]

6. a.) **(12 pts.)** Using your Boolean algebra theorems, convert the following equation to a **minimal** POS form. (Circle that answer) Indicate which theorem(s) you are using in each step.

$$G = (A\overline{B} + CB)(\overline{A} + C)$$

a.) Show work here:

Mistake:

G = AB’A’ + AB’C + A’BC + BCC [T8 / FOIL]
= 0 + AB’C + A’BC + BC
= AB’C + BC(A’ + 1)
= AB’C + BC
= C(AB’+B)
= C(B+A)(B+B’) [T8’]
= C(B+A)
= AC + BC Done! Simplified POS!

7. Full and Half Adders

(

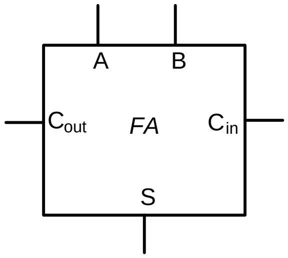
15 pts.)

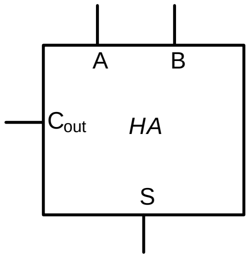
Tim needs a circuit to count how many 1’s are present in a 6-bit number, **X[5:0]**.. This can be accomplished by adding the six bits of X. Produce an output binary number, **Z**, indicating how many 1’s are in the number, X. (Ex.: X = 100110 => three 1’s => Z = 3_{10} = $?_2$)

a. How many bits of output are needed for Z? _______________

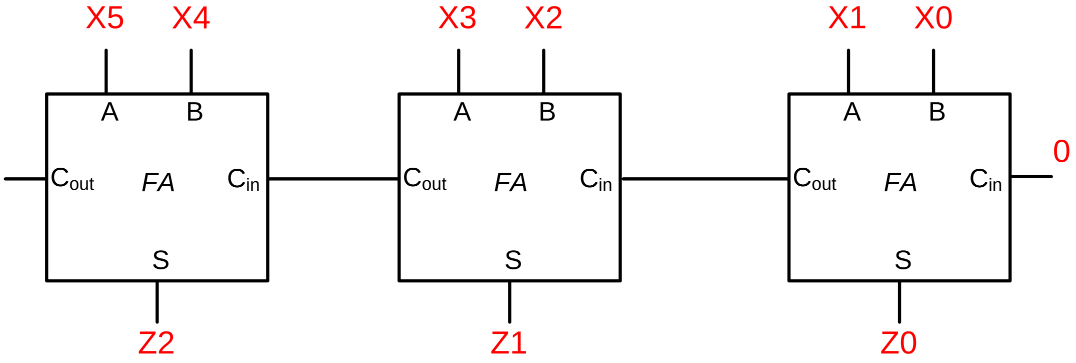
b. Write out the column(s) of addition that need to be performed?

c. Use the **minimal number** of full adders and half adders to implement your design, though your selection of either full or half adders will not be graded (i.e. If only a HA is needed and you use a FA, no points will be deducted.) . A full and half adder have been drawn for you that you can feel free to use. You will need to add more…

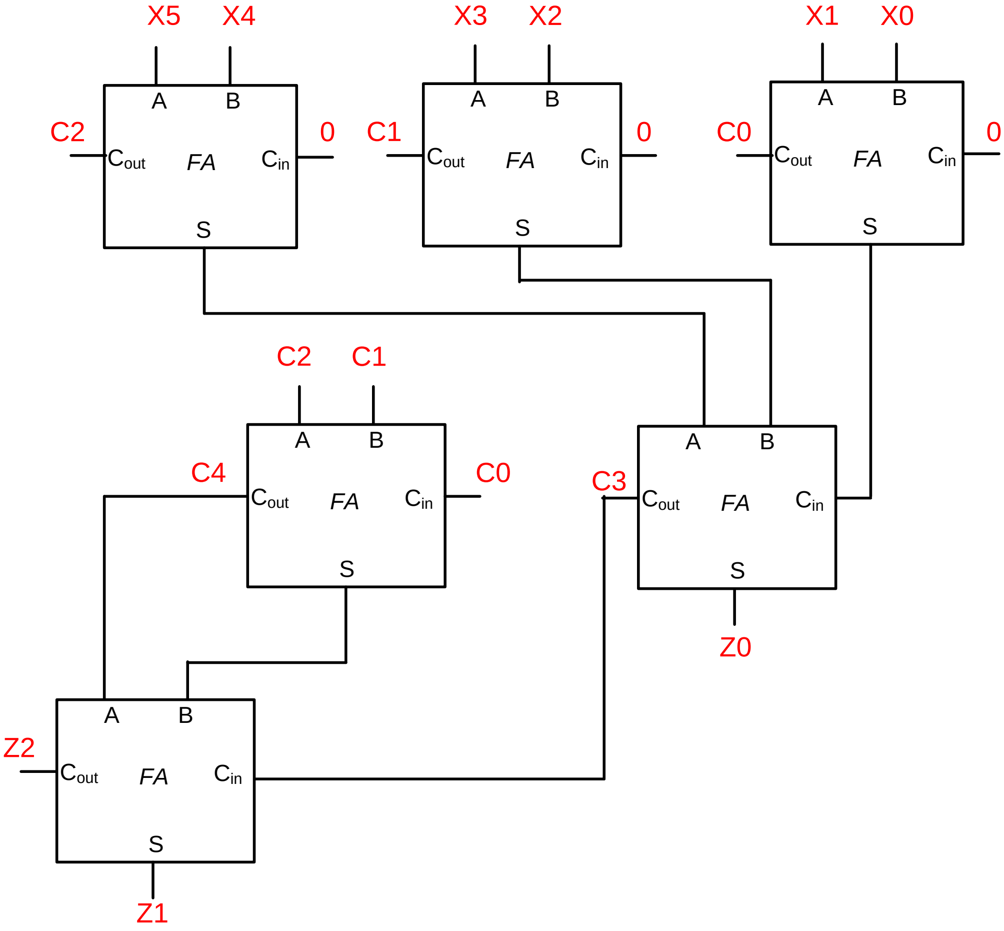




1.)


X5
X4
X3
X2
X1
X0
A
B
Cout
FA
Cin
S
A
B
Cout
FA
Cin
S
A
B
Cout
FA
Cin
S
0
Z2
Z1
Z0

2.)


X5
X4
X3
X2
X1
X0
C2
0
C1
0
C0
0
A
B
Cout
FA
Cin
S
C2
C1
C4
C0
C3
Z0
Z2
Z1

8. Design a circuit with the following inputs and outputs: (

15 pts.)

Inputs:

- **X[2:0]:** a 3-bit, 2's complement number
- **N[1:0]:** a 2-bit unsigned number

Outputs:

- **Z[5:0]:** a 6-bit, 2's complement number equal to **X * 2^N**. (Take X and multiply by the power of 2 indicated by N).

Example: X=100 (-4_{10}), N=01 (1_{10}) Z = 111000 = (-8_{10})

Complete the function table and then implement the design. To implement your circuit you should use the 4-to-1 muxes drawn below and a **minimal** amount of logic gates.

Function Table:

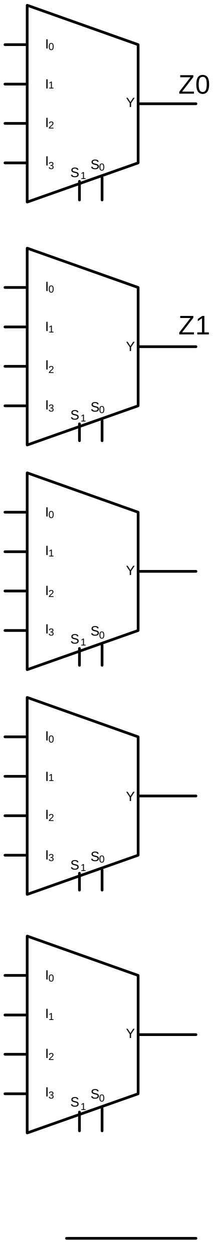
N1	N0	Z5	Z4	Z3	Z2	Z1	Z0
0	0						
0	1						
1	0						
1	1						

Be sure to label the outputs Z2 through Z5 (Z1 and Z0 are labeled for you.)!

Function Table:

N1	N0	Z5	Z4	Z3	Z2	Z1	Z0
0	0	0	0	0	X2	X1	X0
0	1	0	0	X2	X1	X0	0
1	0	0	X2	X1	X0	0	0
1	1	X2	X1	X0	0	0	0

Be sure to label the outputs Z2 through Z5 (Z1 and Z0 are labeled for you.)!

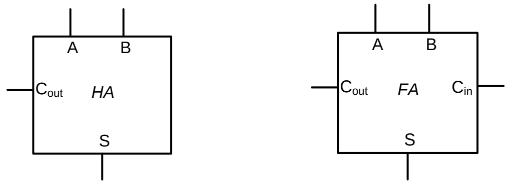


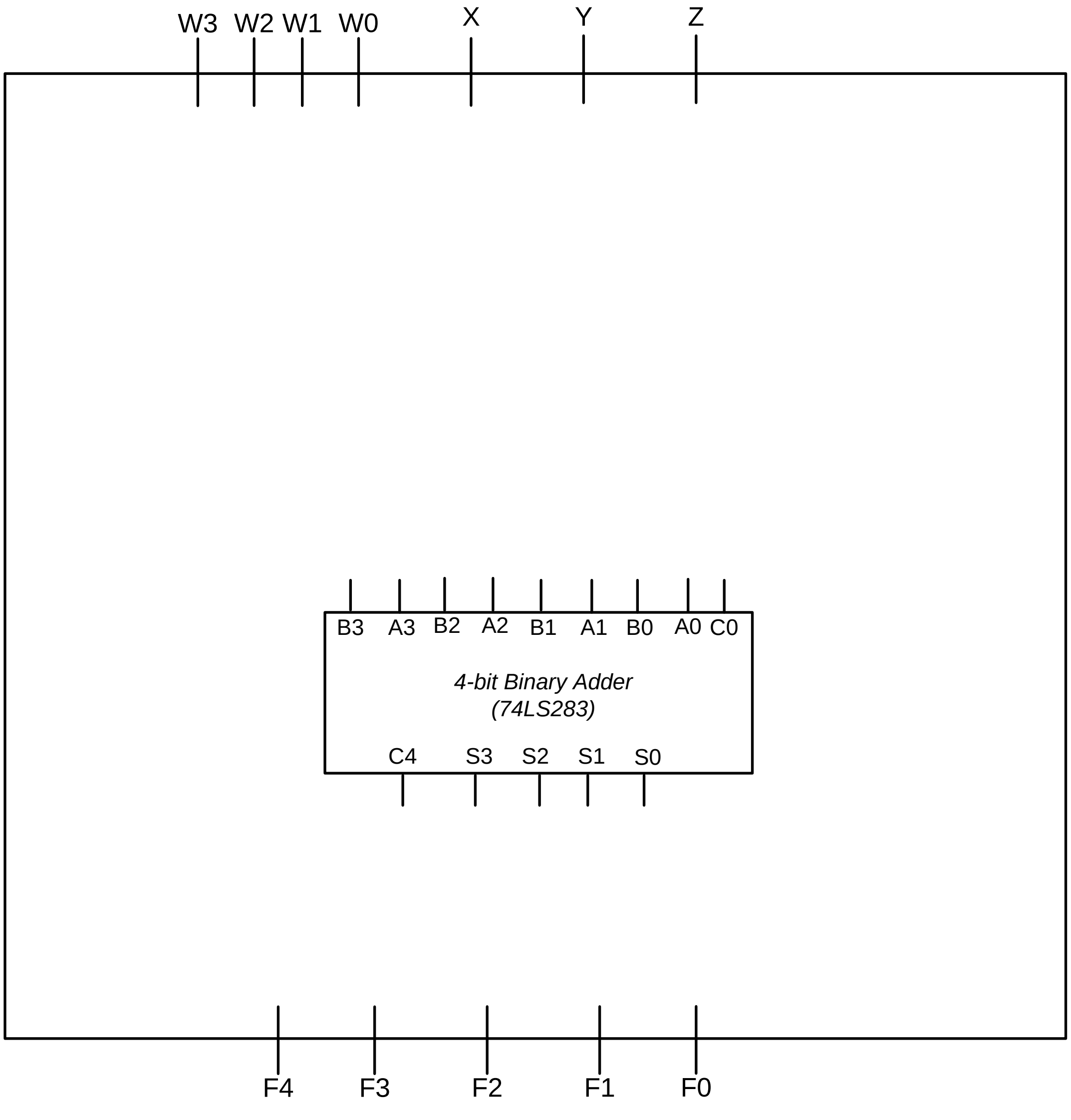
9. Adders (10 pts.)
You are given a 4-bit unsigned number, W[3:0], representing how many people have voted ‘yes’ on a proposition. However, three people vote late. Their vote is represented by three single-bit values: X, Y, and Z (‘1’ = yes / ‘0’ = no). Design a circuit to produce the total sum of yes votes (i.e. design a circuit that adds three one-bit number, X, Y, and Z to a 4-bit number, W[3:0]). The output should be a 5-bit number F[4:0] = W[3:0] + X + Y + Z.

To implement this circuit you will need a 4-bit 74LS283 adder drawn below. You may use additional half- and full-adders (shown below to jog your memory) as necessary. Implement your circuit with a **minimal** amount of logic beyond the ‘283 adder (i.e. use fewer building blocks and also try to substitute a single HA in place of an FA where possible).

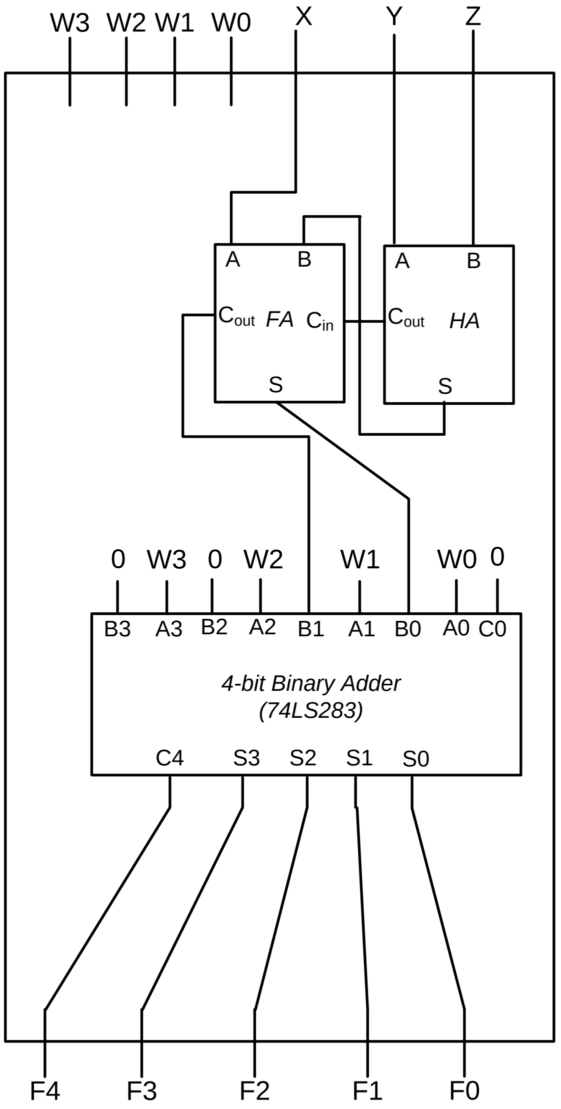
Hint: It may help to write out the column(s) of addition you need to perform.

HA and FA block diagrams…

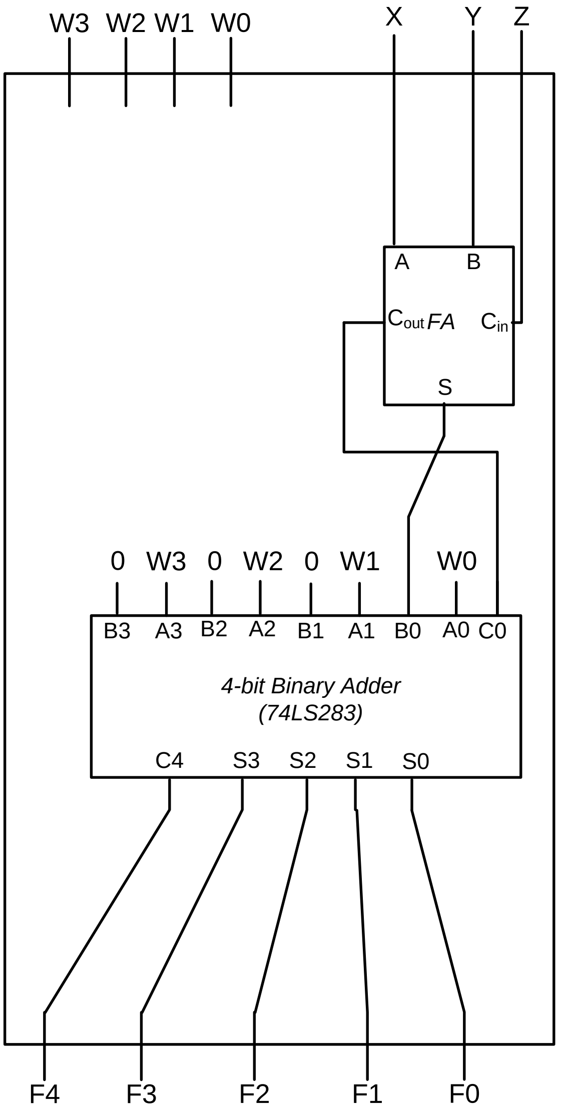




Wrong solution 1



Wrong Solution 2


W3 W2 W1 W0
X
Y
Z
A
B
Cout
FA
Cin
S
0 W3 0 W2 0 W1 W0
B3 A3 B2 A2 B1 A1 B0 A0 C0
4-bit Binary Adder
(74LS283)
C4 S3 S2 S1 S0
F4 F3 F2 F1 F0