1 Basics (1 point each subproblem, 15 pts total)

1.1 How would you represent the decimal value of -19 using ...

1.1(a). ... 6 bit 2s complement notation in binary?

1 0 1 1 0 1 → 101101

-2^5 2^4 2^3 2^2 2^1 2^0

-32 16 8 4 2 1

1.1(b). ... 6 bit signed magnitude notation in hexadecimal?

1 1 0 0 1 1 → 0x33

s 2^4 2^3 2^2 2^1 2^0

- 16 8 4 2 1

1.2 What is the decimal value of 10101011 assuming ...

1.2(a). ... 5.3 fixed point (e.g. xxxxx.xxx), 2s complement notation?

1 0 1 0 1 . 0 1 1

2^4 2^3 2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3}

$-16 + 4 + 1 + \frac{1}{4} + \frac{1}{8} = -10\frac{5}{8} = -10.625$

1.2(b). ... 4E3 floating point (e.g. S EEE MMMM) notation?

S = 1 → −

EEE = 010 → e = 2 − 3 = −1

MMMM = 1011 → m = 1.1011

$-(1.1011 \times 2^{-1}) = -(0.11011) = -\frac{27}{32}$

1.3 If you were trying to represent the decimal value 9/32, what is the closest value you could represent using ...

1.3(a). ... 5.3 fixed point (e.g. xxxxx.xxx) notation (as a decimal fraction)?

00000.010 = $\frac{2}{8} = \frac{8}{32}$ ←

00000.011 = $\frac{3}{8} = \frac{12}{32}$

$\frac{8}{32} = \frac{1}{4} = 0.25$

1.3(b). ... 4E3 floating point (e.g. S EEE MMMM) notation (as that 4E3 binary value)?

$\frac{9}{32} = 0.01001 = 1.0010 \times 2^{-2}$

S = 0

EEE = e + 3 = 001

MMMM = m − 1 = 0010

00010010

1.4 Consider two eight bit signals $X[7:0]$, $Y[7:0]$.

1.4(a). If you consider both as unsigned numbers, what is the numerical relationship of Y as a function of X if you set $Y[5:0]=X[7:2]$; $Y[7:6]$ = 2'b00 ?

$Y = X >> 2 = \left\lfloor \frac{X}{4} \right\rfloor$

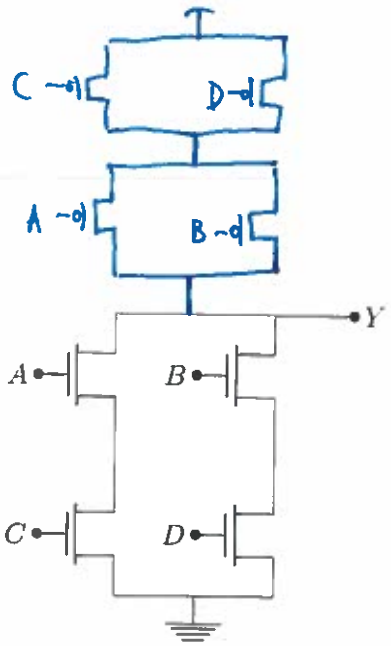
1.4(b). How would you create that same numerical relationship of Y as a function of X if they were both signed numbers instead?

Sign extend:

$Y[5:0] = X[7:2]; \quad Y[7:6] = \{X[7], X[7]\}$

1.5 Consider the given the pull-down network for a CMOS logic gate.

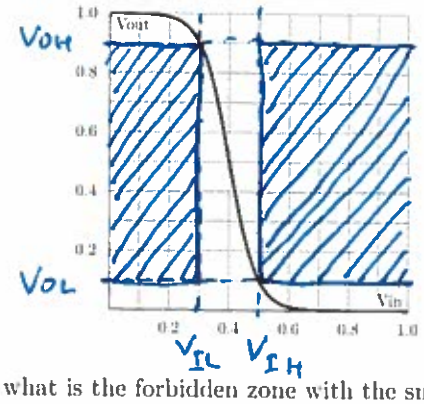
1.5(a). Draw in the pull-up network.



1.5(b). What is a Boolean expression for Y as a function of A, B, C, D? (You do not need to simplify or expand this expression.)

$Y = \neg((A \wedge C) \vee (B \wedge D))$

1.6 Consider the following VTC for a CMOS inverter.



1.6(a). If V_{OL}=0.1 and V_{OH}=0.9, what is the forbidden zone with the smallest width that we could set for V_{in}?

$V_{IL} = 0.3$
$V_{IH} = 0.5$ (width = 0.5 − 0.3 = 0.2)

1.6(b). On the VTC above, shade in the rectangular forbidden region(s) required for any CMOS gate to obey the resulting static discipline.

1.6(c). What is the noise immunity (smallest noise margin) in this static discipline?

$V_{NM_L} = V_{IL} - V_{OL} = 0.2$ ←
$V_{NM_H} = V_{OH} - V_{IH} = 0.4$

NI = 0.2

1.7 Consider a weighted six-sided die with the given probabilities.

i	p_i	bits
1	1/4	2
2	1/4	2
3	1/4	2
4	1/8	3
5	1/16	4
6	1/16	4

$\log_2 \frac{1}{1/4} = \log_2 4$
$\log_2 \frac{1}{1/8} = \log_2 8$
$\log_2 \frac{1}{1/16} = \log_2 16$

1.7(a). Complete the table above with the information content associated with each outcome.

1.7(b). What is the entropy of this system?

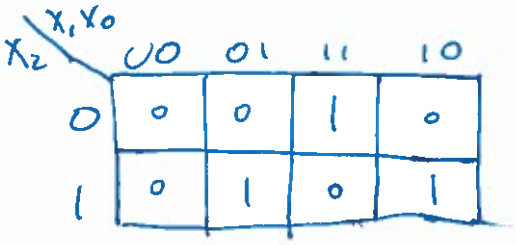
$H = 2\left(\frac{1}{4}\right) + 2\left(\frac{1}{4}\right) + 2\left(\frac{1}{4}\right) + 3\left(\frac{1}{8}\right) + 4\left(\frac{1}{16}\right) + 4\left(\frac{1}{16}\right)$

$H = 2\frac{3}{8} = 2.375$

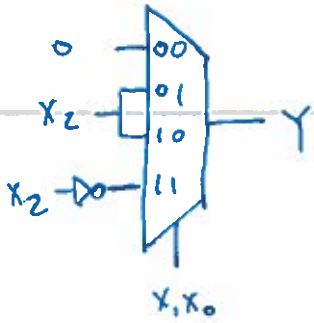
2 Universal gates (8pts)

Consider the three input gate $Y = 2OF3(X_2, X_1, X_0)$, defined as follows:
$Y = 1$ if exactly 2 of the 3 inputs are 1, and 0 otherwise.

2.0(a). Draw the k-map for this function. (1pt)



2.0(b). Implement this gate, given only the non-inverted inputs, using a single inverter and a 4-1 mux. (3pts)

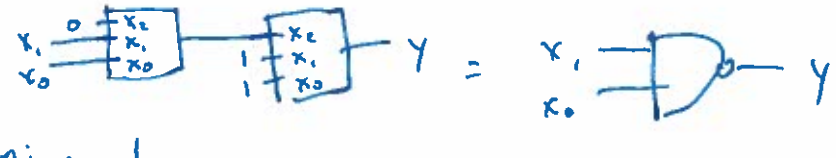


2.0(c). Is this gate universal? Prove your answer. (4pts)

Yes!

Not: 

And: 

NAND = 

↑ universal

3 Superfunctions (12pts)

Consider a single valued function of n inputs $Y = f(X) = f(X_{n-1}, X_{n-2}, \cdots, X_0)$.

3.0(a). There are M different such functions in total. If $n = 3$, what is M? (2pts)

2^n input combinations, each w/ 2 possible output values $\Rightarrow$ $M = 2^{2^n}$ functions

3.0(b). If we label each of the M possible function with a unique (binary) integer, we need m bits for this label. What is m? (1pt)

$m = \log_2 M = \log_2 2^{2^n} =$ 2^n bits

3.0(c). Consider a specific function f_i, which we know can be implemented by a single CMOS gate. If there is a specific input $A = A_{n-1}, A_{n-2}, \cdots, A_0$, such that $A_{n-1} = 0$ and $f_i(A) = 0$, what, if anything, can we say about $f_i(A')$, where $A' = 1, A_{n-2}, \cdots, A_0$? Why? (2pts)

If $A_{n-1} = 0$, any NMOS transistors driven by A_{n-1} act as open switches. But if $f_i(A) = 0$, there must be some pull down path, and therefore this pulldown path only passes through MOSFETS not driven by A_{n-1}. This path remains unchanged when $A_{n-1} \to 1$, and so

$f_i(A') = 0$ as well.

3.0(d). We can define a superfunction $g(I, X)$ that takes in an m bit input I identifying the function and the n bit input X, and returns $Y = f_I(X)$ for the specific function f_I labeled by the value I. Come up with the mapping from I to f_I that lets you implement this function with a single mux. For what value I is f_I an n-input NAND gate? (3pts)

We can define the function $f_I(x)$ by its minterm expansion, saying $f_I(x) = 1$ — e.g. m_x is a minterm of f_I — corresponds to the x'th bit of I being a "1", and $f_I(x) = 0 \iff x$'th bit of I is "0". That is, in truth table form:

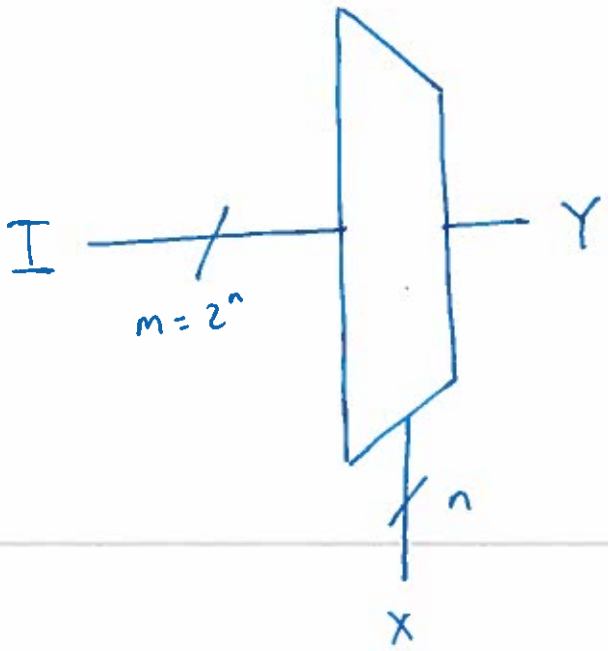
X	Y
0	I_0
1	I_1
⋮	⋮
$2^n - 1$	I_{m-1}

I

With this mapping, NAND →

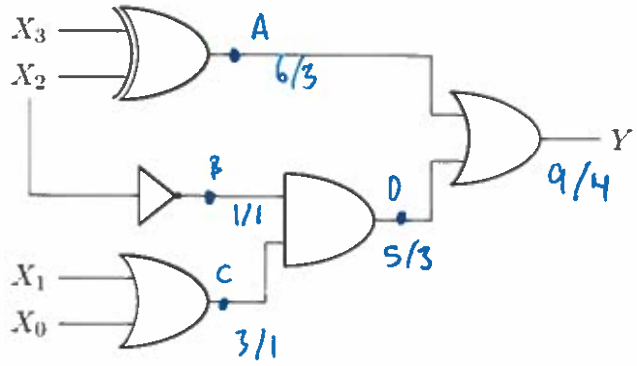
$I = 011\ldots1 = 2^{m-1} - 1$

3.0(e). Draw and label this mux-based implementation of $Y = g(I, X)$. (4pts)



4 Boolean gates (8pts)

Consider the following Boolean logic system, with gate delays as in the corresponding table.



Gate	t_{PD}	t_{CD}
NOT	1	1
OR	3	1
AND	2	2
XOR	6	3

X_3	X_2	X_1	X_0	n	Y
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	1
0	0	1	1	3	1
0	1	0	0	4	1
0	1	0	1	5	1
0	1	1	0	6	1
0	1	1	1	7	1
1	0	0	0	8	1
1	0	0	1	9	1
1	0	1	0	10	1
1	0	1	1	11	1
1	1	0	0	12	0
1	1	0	1	13	0
1	1	1	0	14	0
1	1	1	1	15	0

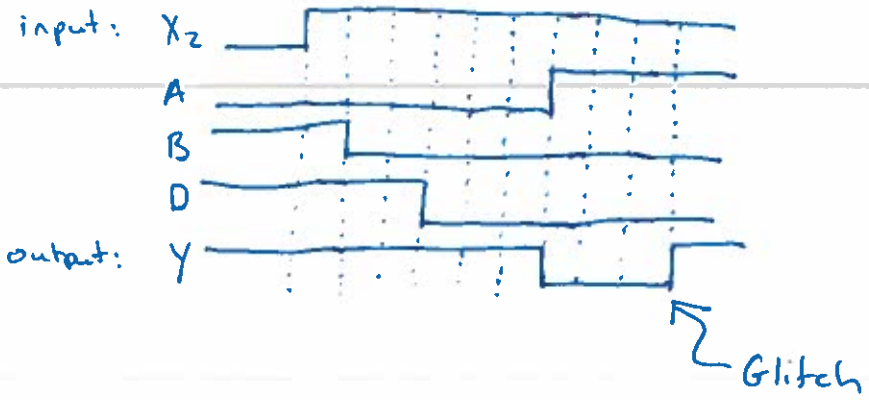
4.0(a). Assume we are able to implement the Boolean gates directly as shown in the diagram above. Given the propagation and contamination delays of each of those gates, what are the propagation and contamination delays of this system? (3pts)

$t_{PD} = 9$

$t_{CD} = 4$

4.0(b). If each Boolean gate in the system is lenient, is the entire system of part 4.0(a) lenient? If it is, explain why. If it is not, draw a timing diagram proving that the system can glitch. (3pts)

No. Consider $X_3 = 0$, $X_1 = 1$, $X_0 = 1$:



4.0(c). Complete the truth table above for Y as a function of X_3, X_2, X_1, X_0. (1pt)

4.0(d). Write the function using either minterm or maxterm shorthand, that is, in the form $Y = m(n_1, n_2, \cdots)$ or $Y = M(n_1, n_2, \cdots)$, where $n_1, n_2, \cdots$ are integers. (1pt)

$Y = M(0, 12, 13, 14, 15)$

5 Privileged Encoder (17 pts)

Recall that a decoder goes from an n-bit binary encoded input to an m-bit one-hot output. We would like to build an **encoder** to go the other way, generating an n-bit binary encoded integer $Y = Y_{n-1}Y_{n-2}\ldots Y_0$ from an m-bit input $X = X_{m-1}X_{m-2}\ldots X_0$.

However, since we can't guarantee that the m-bit input will be one-hot, we will need to handle the case when either several or none of the input bits are high. To do so, we will have an additional one-bit input P indicating which direction is privileged, and an additional one-bit output H indicating whether any of the input bits were high:

- If none of the input bits are high, we *don't care* what the n-bit output Y is, but the one bit output $H = 0$.
- If exactly one of the input bits is high, then Y is the index of the input that is high, and $H = 1$.
- If multiple of the input bits are high, and $P = 1$, then Y is the largest index of the inputs that are high, and $H = 1$.
- If multiple of the input bits are high, and $P = 0$, then Y is the smallest index of the inputs that are high, and $H = 1$.

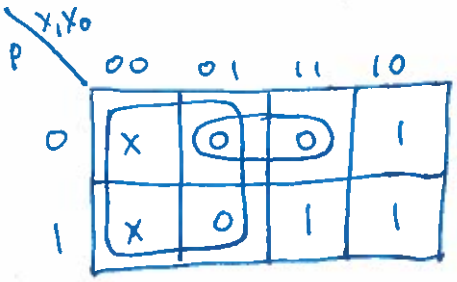
5.0(a). What must n be as a function of m? (1pt)

$$n = \lceil \log_2(m) \rceil$$

5.0(b). Let us first consider the case where $m = 2$. Draw the truth table for outputs Y and H as a function of inputs P and X. (1pt)

P	X_1	X_0	H	Y
0	0	0	0	X
0	0	1	1	0
0	1	0	1	1
0	1	1	1	0
1	0	0	0	X
1	0	1	1	0
1	1	0	1	1
1	1	1	1	1

5.0(c). Use a k-map for $m = 2$ to determine what the *don't care* values should be to minimize a product-of-sums implementation for Y. In this case, what is the minimal Boolean PoS expression for Y? (2pts)



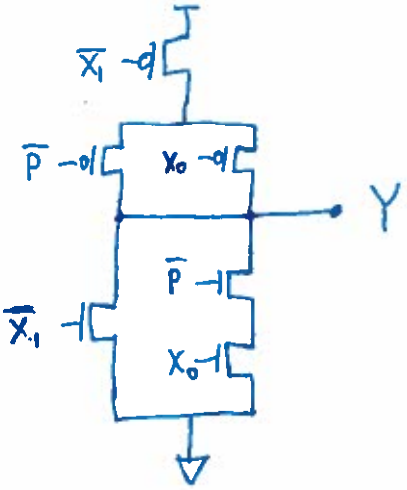
$$\overline{Y} = \overline{X_1} \vee (\overline{P} \wedge X_0)$$

$$\Rightarrow Y = X_1 \wedge \overline{(\overline{P} \wedge X_0)}$$

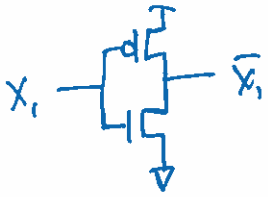
$$\Rightarrow \boxed{Y = X_1 \wedge (P \vee \overline{X_0})}$$

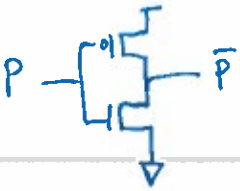
5.0(d). Implement this expression for Y using 3 CMOS gates: two inverters and one 6-transistor gate. (3pts)

$Y = X_1 \wedge (P \vee \overline{X}_0)$ → Pullup network:

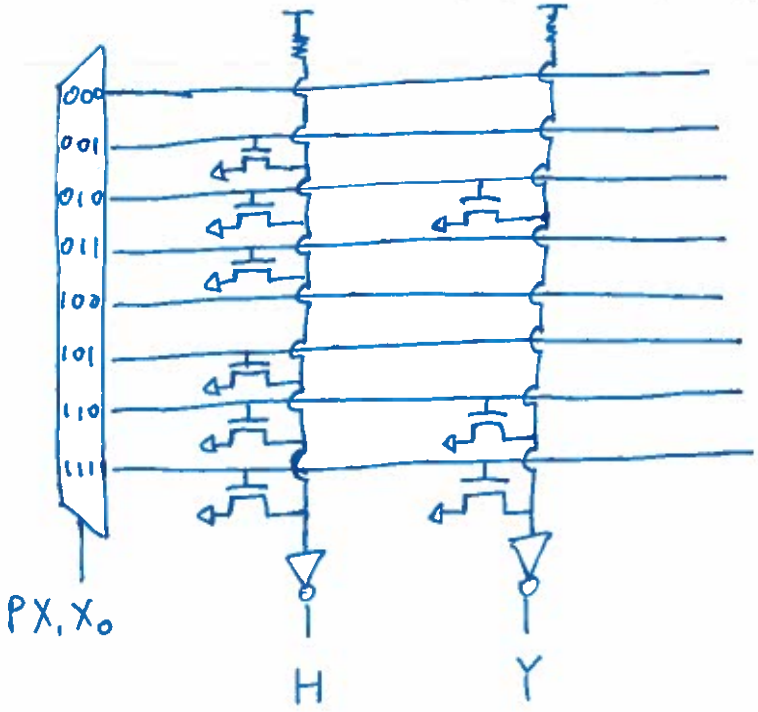


to generate inverted signals:

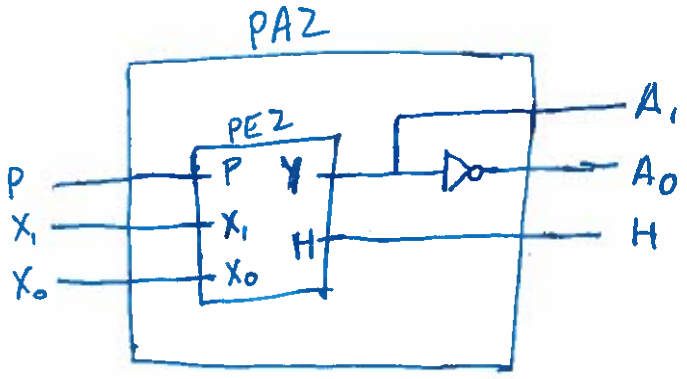




5.0(e). Implement PE2, the privileged encoder for $m = 2$, using a ROM to generate both Y and H from inputs P and X. Ensure that all generated outputs obey the CMOS static discipline. (3pts)



5.0(f). A **arbiter** generates an m-bit one-hot output from an m-bit input. Use the PE2 block plus any additional CMOS gates to implement the PA2 block, which outputs H as from the PE2 block and A, an $m = 2$-bit one-hot output that corresponds to the index Y as set from PE2. (2pts)



5.0(g). Use two PA2 blocks, plus any CMOS gates or muxes, to generate the PA4 block, which is the privileged arbiter generating an $m = 4$ bit one-hot output A and a one bit output H from $m = 4$ bit input X and one bit input P. (3pts)

We can divide our input X into two sets of 2 bits:
Upper = X_U = X[3:2] and Lower = X_L = X[1:0],
and decide what to do based on which of those hit, i.e. contain high values, or H=1. If only the upper half hit, or if both halves hit and P=1, then we will use the upper one-hot half and set the lower half to 2'b00. Otherwise, we will use the lower half one-hot output, and set the upper half to 2'b00. The 4 bit value hits if either half hits.

Use upper half if: $U = H_U \overline{H_L} + H_U H_L P = H_U(\overline{H_L} + H_L P)$
Use lower half if: $L = \overline{U} = \overline{(H_U)(\overline{H_L} + H_L P)} = \overline{(H_U)(\overline{(H_L)(\overline{H_L P})})}$

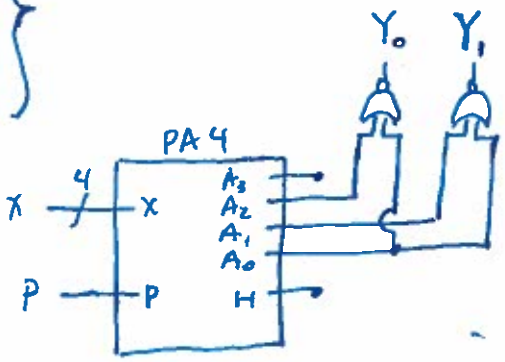
See next page →

5.0(h). Using any CMOS gates or muxes, generate the n-bit result Y from the $m = 4$-bit output A of the PA4 block. (2pts)

We know A is one-hot, so we get the following truth table:

A_3	A_2	A_1	A_0	Y_1	Y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

$Y_1 = \overline{A_1 + A_0}$; $Y_0 = \overline{A_2 + A_0}$



5.0 (g) cont'd: