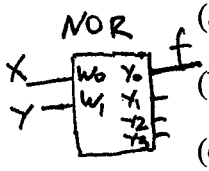
Sample Solutions

CprE 281: Digital Logic
Midterm 2: Friday Oct 28, 2022

Name: _____ **ID Number:** _____

Lab Section: **Tue 2-5 (#9)** **Wed 7-10 (# 6)** **Thur 8-11 (#15)**
(circle one) **Wed 11-2 (#13)** **Thur 11-2 (#11)**
Wed 6-9 (#12) **Thur 2-5 (# 8)**
Thur 5-8 (# 7)

1. True/False Questions (10 x 1p each = 10p)



(a) I forgot to write down my name, lab section, and student ID number. TRUE / (FALSE)

(b) Any Boolean function can be implemented using only 2-to-4 decoders. (TRUE) / FALSE

(c) A full-adder can be constructed using two half-adders and one XOR (OR) gate. TRUE / (FALSE)

(d) The phrase "This is the way" refers to a code. (TRUE) / FALSE

(e) A JK flip-flop has 2 inputs, 2 outputs, and can store 2 bits of information. TRUE / (FALSE)

(f) The characteristic tables of the SR latch and the $\overline{S}\,\overline{R}$ latch are the same. TRUE / (FALSE)

(g) A demultiplexer changes its outputs on the positive edge of the clock. TRUE / (FALSE)

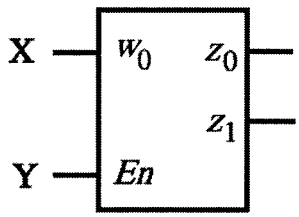
(h) A 32-bit hierarchical carry-lookahead adder sets the carries in 7 gate delays. (TRUE) / FALSE

(i) In 1's complement, the magnitude of a number with all bits set to 1 is 0. (TRUE) / FALSE

(j) The ALU of the i281 CPU has one 8-bit ripple-carry adder. (TRUE) / FALSE

2. Decoder Expressions (5p)

Draw the truth table for the 1-to-2 decoder with enable that is shown below. Then, write the Boolean expressions for the outputs Z_0 and Z_1 in terms of the inputs X and Y.



X	Y	z_0	z_1
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

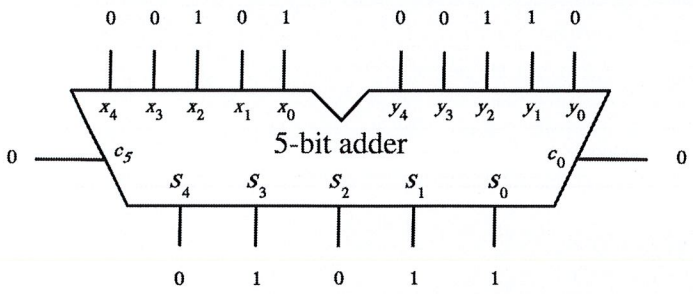
$z_0 = \overline{X}\,Y$

$z_1 = X\,Y$

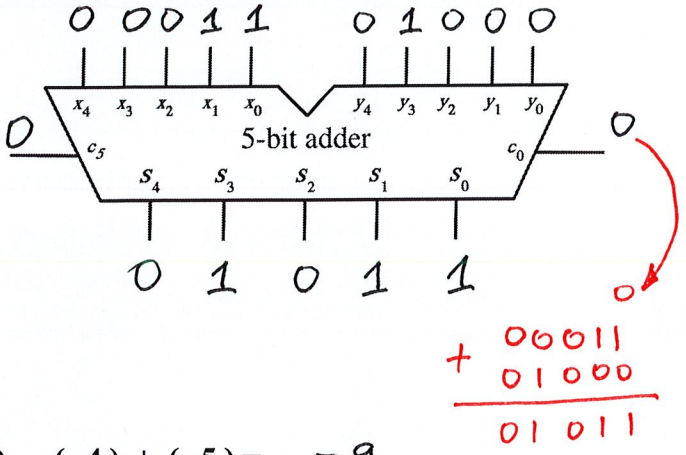
3. Computations with Adders (5 x 3p each = 15p)

In all problems below, the binary numbers are stored in <u>2's complement representation</u>. For each of the following, assign either a 0 or a 1 to each input and output of the 5-bit adder such that it computes the given expression. The problem in a) is already solved.

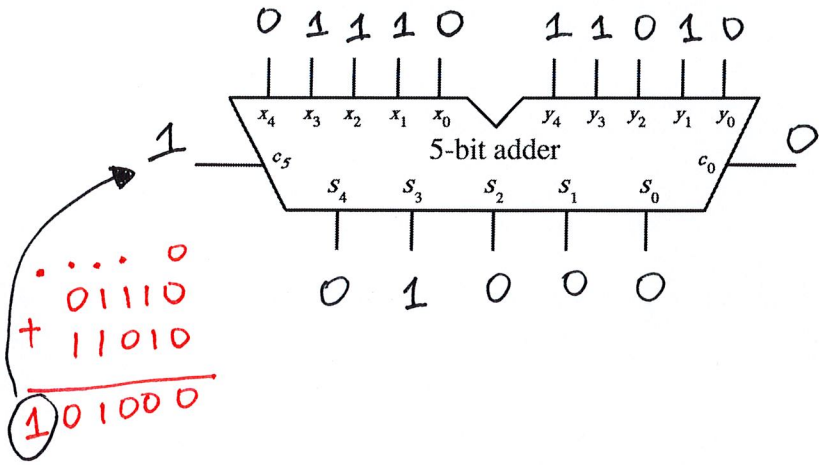
a) **(+5) + (+6) = +11**



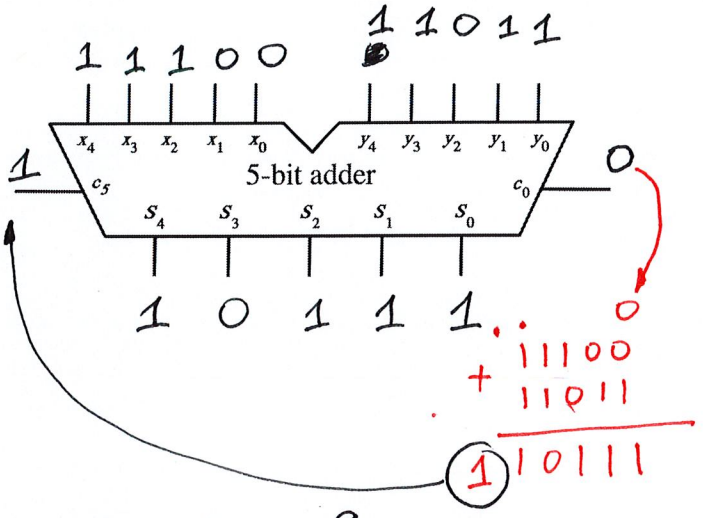
b) **(+3) + (+8) =** +11



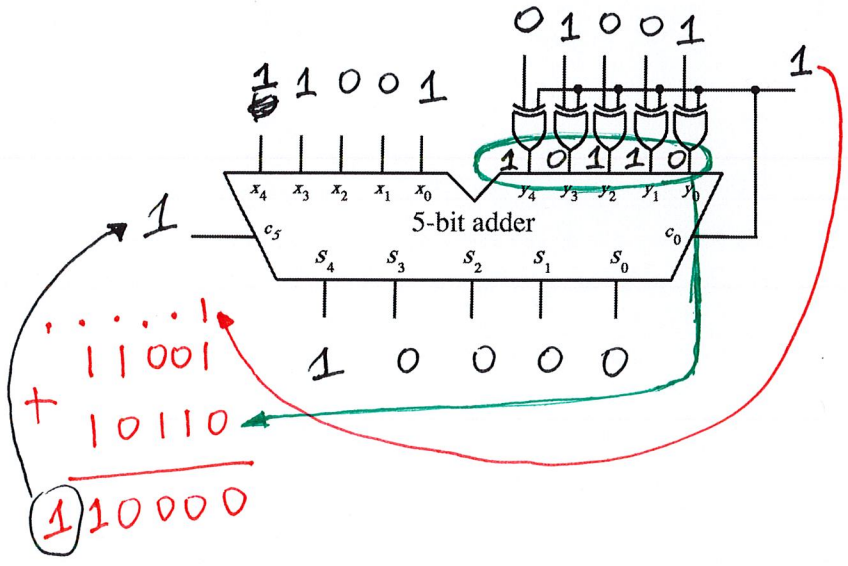
c) **(+14) + (-6) =** +8



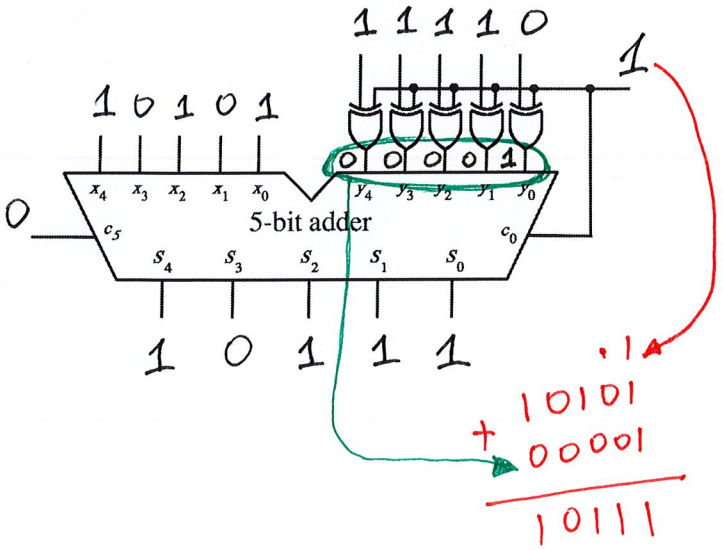
d) **(-4) + (-5) =** -9



e) **(-7) - (+9) =** -16

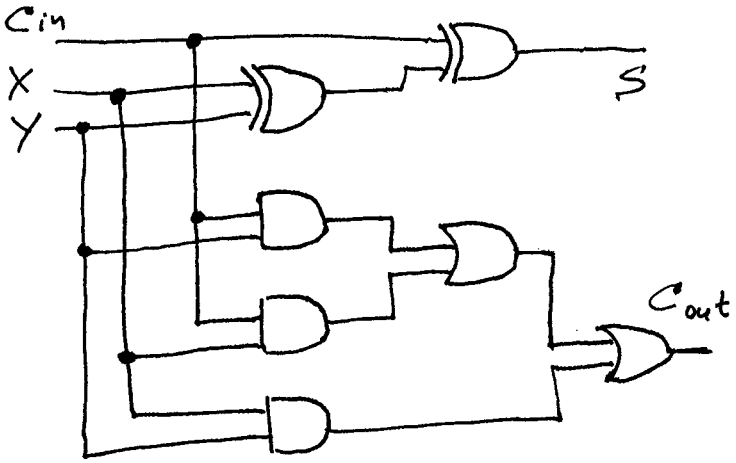


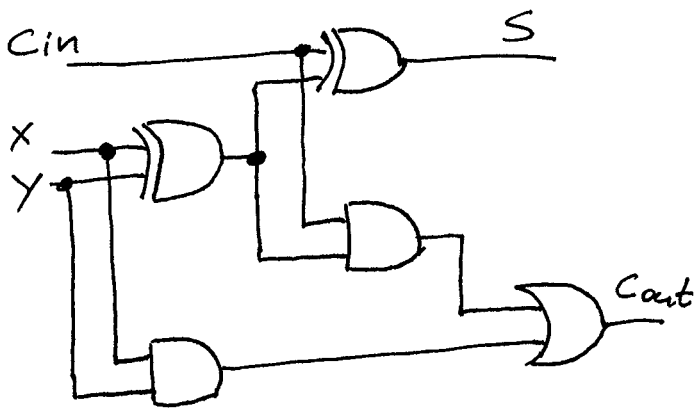
f) **(-11) - (-2) =** -9



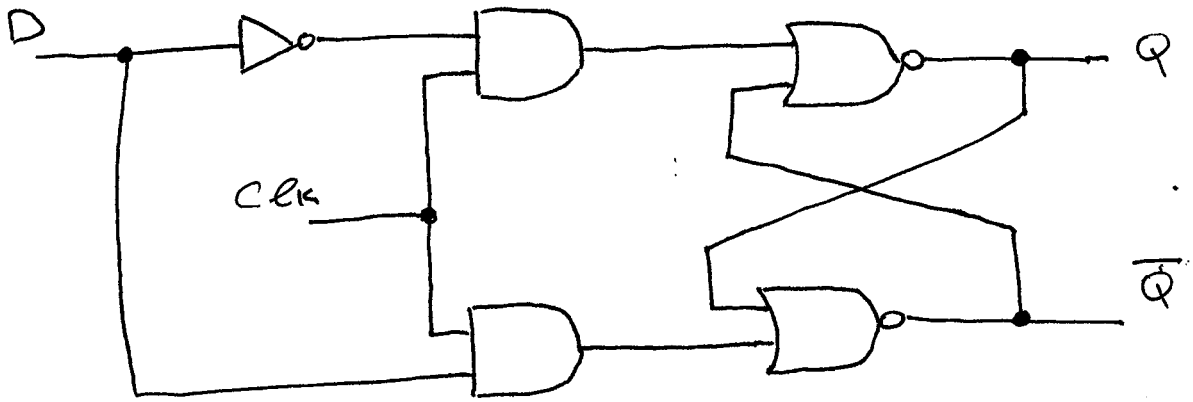
4. Basic Circuits (3 x 5p each = 15p).

(a) Draw the complete wiring diagram for a full-adder using only 2-input logic gates. Clearly label all inputs and outputs.

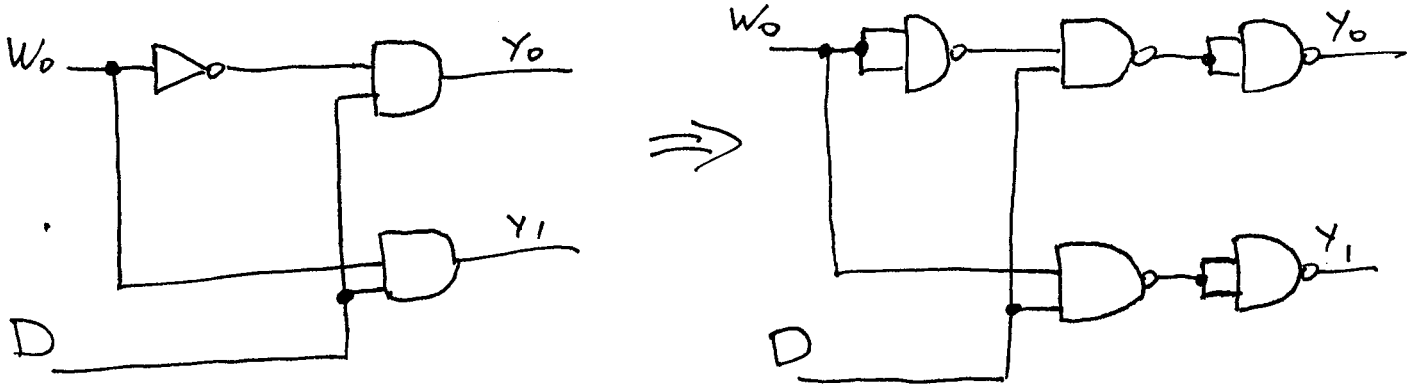




(b) Draw the complete wiring diagram for a gated D latch (with NOR gates for the latch). Clearly label all inputs and outputs.



(c) Draw the complete wiring diagram for a 1-to-2 demultiplexer using only NAND gates. Clearly label all inputs and outputs.



5. Number Conversions (3p + 4p + 4p + 4p = 15p)

(a) Convert 191_{10} to hexadecimal.

191 / 16 = 11 15

11 / 16 = 0 11

$191_{10} = BF_{16}$

11 = B 15 = F

(b) Convert -65_{10} to an <u>8-bit</u> binary number in 2's complement representation.

65/2 = 32 1
32/2 = 16 0
16/2 = 8 0
8/2 = 4 0
4/2 = 2 0
2/2 = 1 0
1/2 = 0 1

1000001 (7-bit)
0 1000001 (pad)
1 0111111 (negate)

result in 2's complement.

(c) Convert the following 32-bit float number (in IEEE 754 format) to decimal.

negative → 1 | 10000010 | 01110000000000000000000

128 + 2 = 130

$\frac{1}{4}$ $\frac{1}{8}$ $\frac{1}{16}$

$$(-1)^1 \times 2^{130-127} \times \left(1 + \frac{1}{4} + \frac{1}{8} + \frac{1}{16}\right) = -2^3 \times \left(\frac{16+4+2+1}{16}\right) = -\frac{23}{2}$$

$$= -11.5$$

(d) Write down the 32-bit floating point representation (in IEEE 754 format) for -42.0

42 / 32 = 1.3125

32 = 2^5

132 − 127 = 5

0.3125
− 0.25
0.0625

0.5
0.25 ✓
0.125
0.0625 ✓

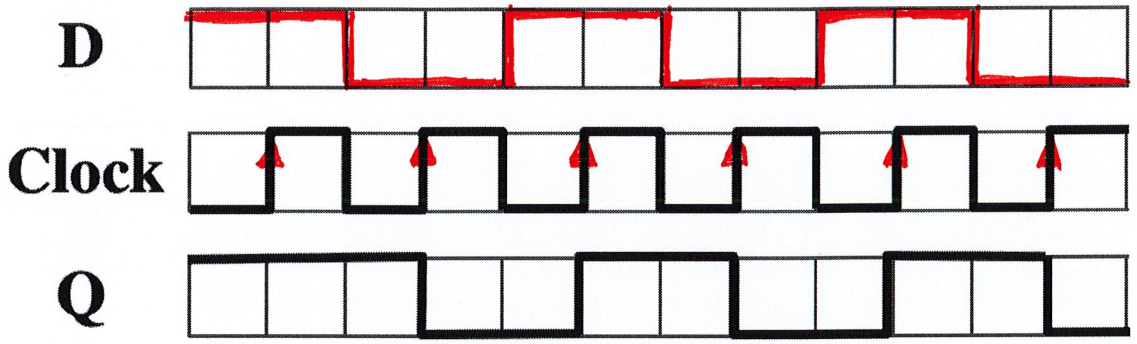
1 | 10000100 | 01010000...0

19 zeros

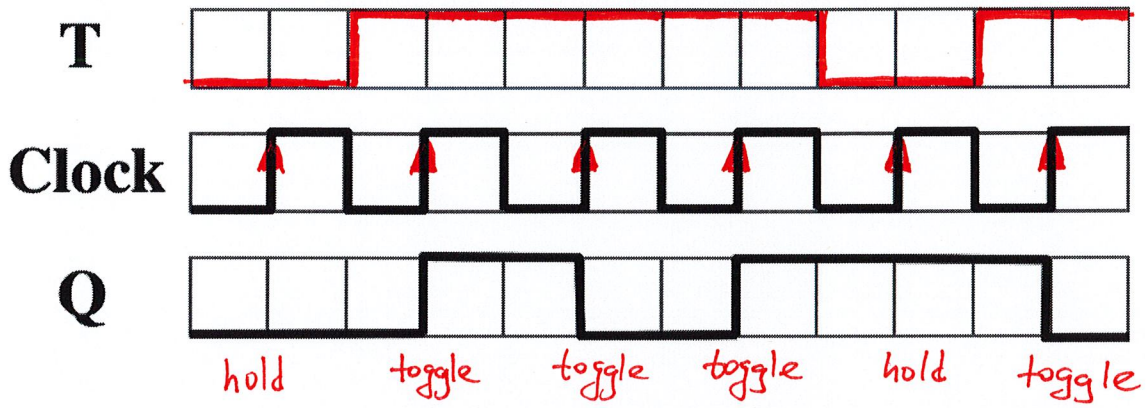
6. Flip-Flops and Timing Diagrams (3 x 5p = 15p)

Complete the timing diagram for the specified flip-flop such that the output Q will be as indicated. Assume that the input signal can change only on the vertical lines. Also, assume that the setup time t_{su} and the hold time t_h are each equal to the width of one square.

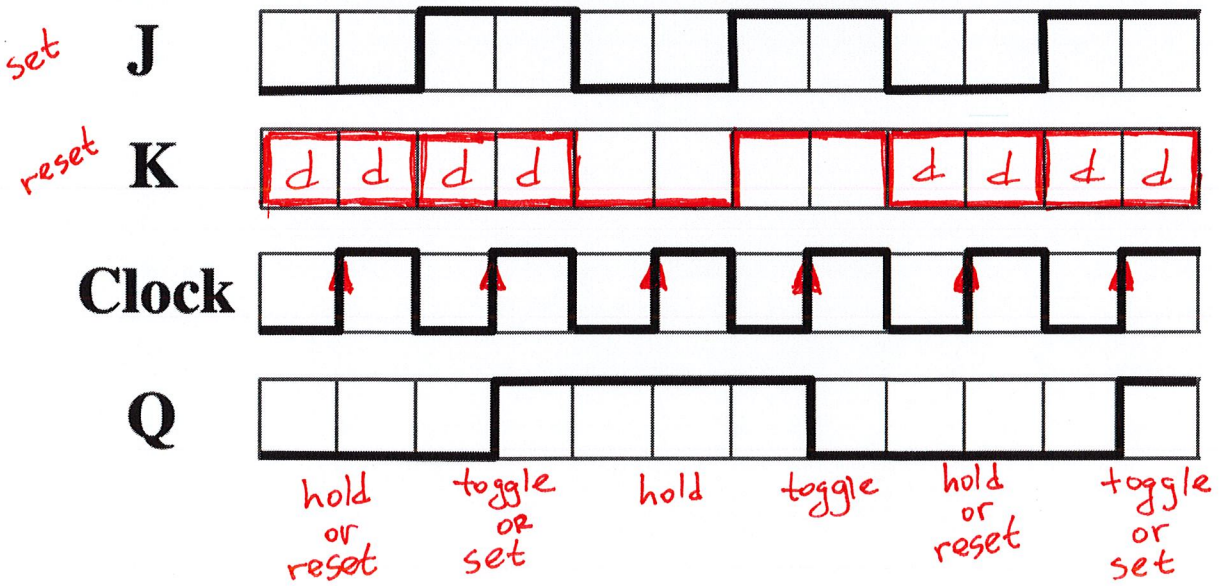
a) Complete the timing diagram for the D input to a positive-edge triggered D flip-flop.



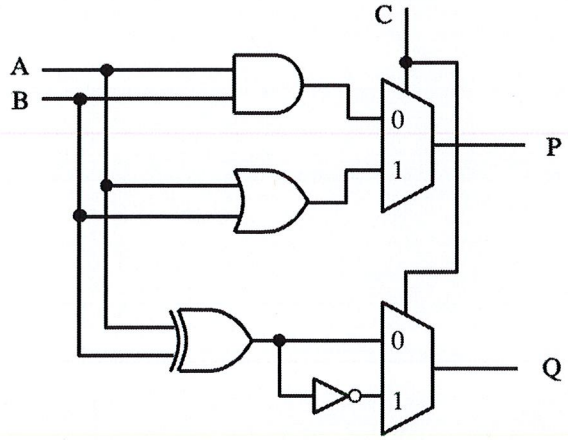
b) Complete the timing diagram for the T input to a positive-edge triggered T flip-flop.



c) Complete the timing diagram for the K input to a positive-edge triggered JK flip-flop. If more than one value is possible for J at any time, indicate that with a don't care (d).



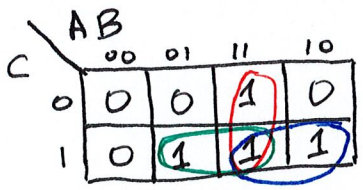
7. Mystery Circuit (3 x 5p = 15p)

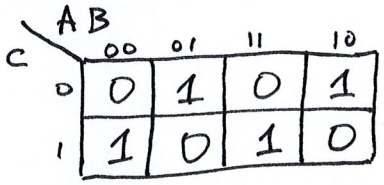


a) Draw the truth table for the outputs P and Q as functions of A, B, and C. **(5p)**

A	B	C	P	Q
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) Use K-maps to find the minimum-cost SOP expressions for P and Q. **(5p)**





$P = AB + BC + AC$

$Q = A \oplus B \oplus C$

c) What type of familiar circuit is this equivalent to? Explain. **(5p)**

This is a full-adder with Q = Sum and P = Carry. The inputs A, B, and C are the inputs to the full-adder.

8. Implement a JK flip-flop using a T flip-flop. (3p +4p +4p = 10p)

You need to implement a JK flip-flop, but you only have a T flip-flop and some extra logic gates (ANDs, ORs, NOTs, and XORs). You also know the effects of the J and K inputs on the output Q. Hopefully, that is all you need to get the job done.

a) Draw the truth table for a positive-edge-triggered JK flip-flop.
Hint: the inputs are J, K, and Q(t); the output is Q(t+1). (3p)

b) Add a column T to the truth table and infer the value of T that will cause the transition from Q(t) to Q(t+1). Use a K-map to derive the minimum-cost SOP expression for T. (4p)

c) Draw the circuit for the JK flip-flop, using the graphical symbol for a T flip-flop and any other necessary logic gates. Clearly label all inputs, outputs, and pins. (4p)

a)

	J	K	Q_t	Q_{t+1}
hold	0	0	0	0
	0	0	1	1
reset	0	1	0	0
	0	1	1	0
set	1	0	0	1
	1	0	1	1
toggle	1	1	0	1
	1	1	1	0

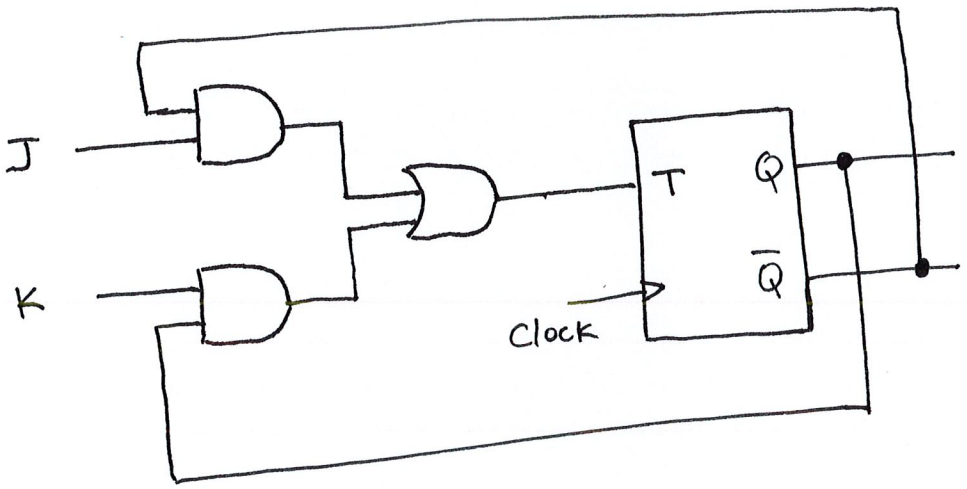
b)

T
0
0
0
1
1
0
1
1

Q_t \ JK	00	01	11	10
0	0	0	1	1
1	0	1	1	0

$T = J\overline{Q_t} + KQ_t$

c)



9. Alternative Implementation (3 x 5p each = 15p)

a) Draw the truth table for the function $f(a, b, c) = \overline{a + b} + \bar{c}\,(\overline{\bar{a} + \bar{b}})$.

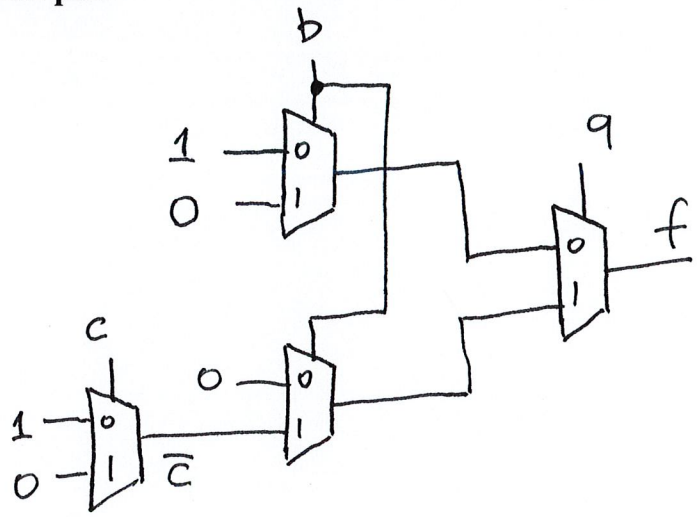
$= \overline{a+b} + \bar{c}(\bar{\bar{a}} \cdot \bar{\bar{b}})$

$= \overline{a+b} + ab\bar{c}$

a	b	c	$\overline{a+b}$	$+ ab\bar{c}$	f
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	1	1
1	1	1	0	0	0

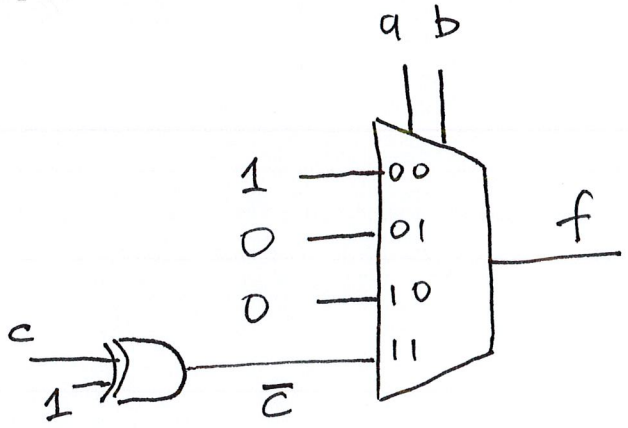
b) Implement this function using a <u>minimal</u> number of 2-to-1 multiplexers. You must use <u>only</u> 2-to-1 multiplexers and <u>no other logic gates</u>. Assume that the signals a, b, and c are available <u>only</u> in their non-inverted form. You can also use the constants 0 and 1. Clearly label all inputs, outputs, and pins.

a	b	c	f	
0	0	0	1	} 1
0	0	1	1	
0	1	0	0	} 0
0	1	1	0	
1	0	0	0	} 0
1	0	1	0	
1	1	0	1	} $\bar{c}$
1	1	1	0	



c) Implement this function using one 4-to-1 multiplexer and one XOR gate. Clearly label all inputs, outputs, and pins.

a	b	c	f	
0	0	0	1	} 1
0	0	1	1	
0	1	0	0	} 0
0	1	1	0	
1	0	0	0	} 0
1	0	1	0	
1	1	0	1	} $\bar{c}$
1	1	1	0	



10. Error-Correcting Code (7p + 8p = 15p) [Use the space on the next page if needed.]

(a) The Hamming (7, 4) code is a popular error-correcting code that is used to store or transfer data that can be corrupted by noise. Given four data bits (d4, d3, d2, and d1) this code computes three parity bits (p3, p2, and p1) and interleaves them to construct a 7-bit message m7, m6, m5, m4, m3, m2, m1 from d4, d3, d2, p3, d1, p2, p1. Parity bit p1 is computed from d1, d2, and d4. Bit p2 from d1, d3, and d4. And bit p3 from d2, d3, and d4. In all cases, what is computed is even parity. That is, the parity bit is set to 0 if the number of 1's in the three corresponding data bits is even. Otherwise, it is set to 1. Draw a circuit that encodes a 7-bit message given 4-bit input data. Explain your solution.

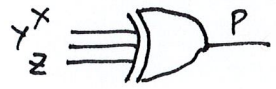
(b) When the message is received the code has the ability to detect and correct 1-bit errors. Three new parity bits are computed: P3, P2, and P1 (note the capital letter). P1 is 0 if the number of 1's in m7, m5, m3, and m1 is even. Otherwise, it is 1. Similarly for P2, which depends on m7, m6, m3, and m2. And P3, which is computed from m7, m6, m5, and m4. If P3=P2=P1=0, then the message was not corrupted. When P3, P2, P1 is interpreted as a binary number it points to the 1-based index of the bit in m7, m6, m5, m4, m3, m2, m1 that is wrong. This bit can be corrected by simply flipping its value (from 1 to 0 or from 0 to 1). Draw a circuit that uses this method to detect and correct 1-bit errors. Explain.

a)

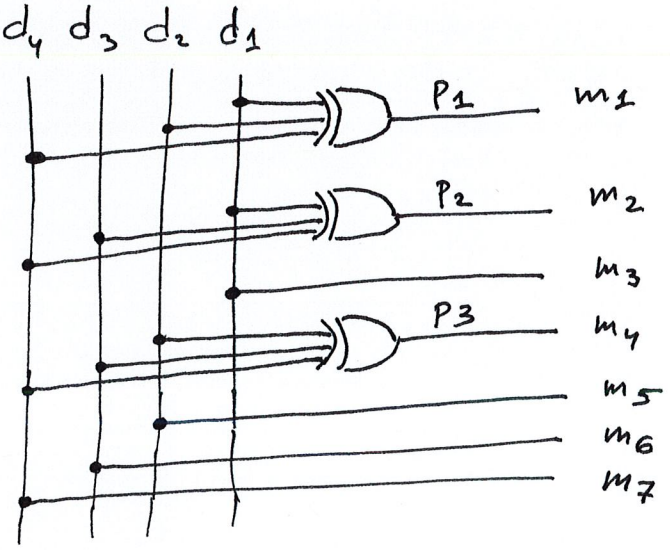
X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Z \ XY	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$P = X \oplus Y \oplus Z$



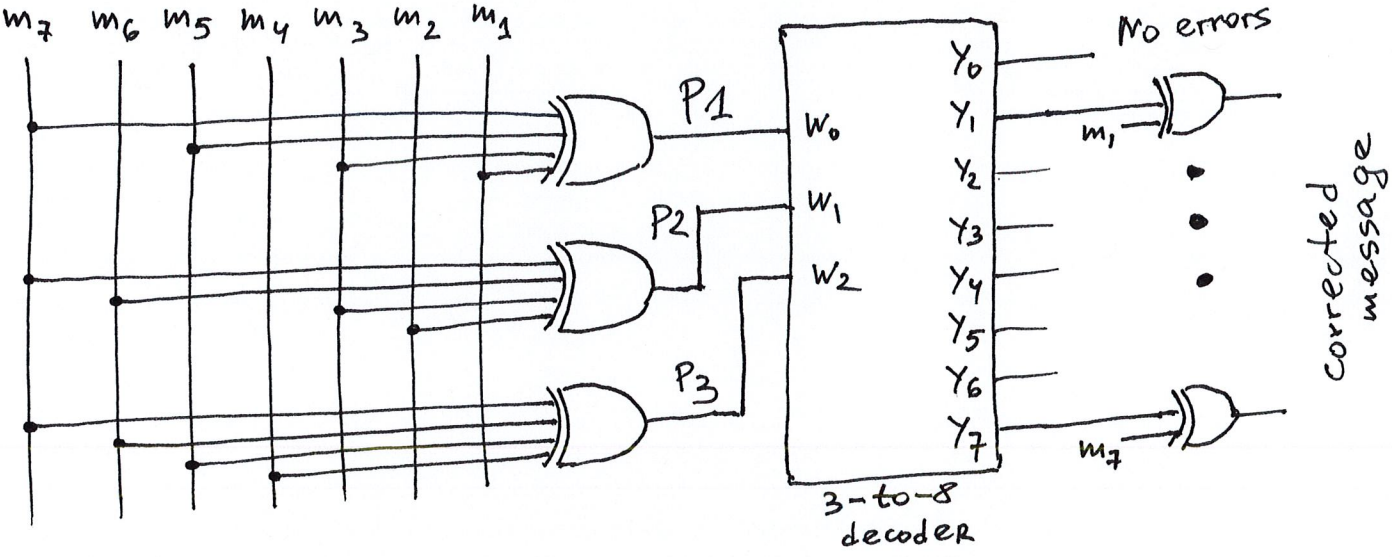
The XOR is used to compute the parity in all 3 cases. Just its three inputs are different.



b)

d_4 d_3 d_2 P_3 d_1 P_2 P_1

m_7 m_6 m_5 m_4 m_3 m_2 m_1



The parity bits are now computed with 4-input XORs. They go as inputs to the 3-to-8 decoder. If $P_1 = P_2 = P_3 = 1$, then only Y_0 is set to 1 to indicate that the message is OK. All other Y's will be 0's and the XORs will output the m's. In all other cases the decoder will have a 1 on the index of the wrong/corrupted bit. That one will make the XOR work like NOT which will correct the wrong bit.

Question	Max	Score
1. True/False	10	
2. Decoder Expressions	5	
3. Computations with Adders	15	
4. Basic Circuits	15	
5. Number Conversions	15	
6. Flip-Flops	15	
7. Mystery Circuit	15	
8. JK flip-flop with T flip-flop	10	
9. Alternative Implementation	15	
10. Error-Correcting Code	15	
TOTAL:	130	