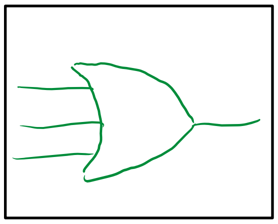
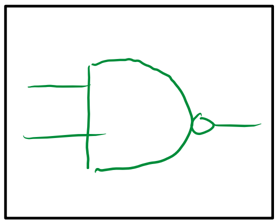
1) Draw the gate schematic symbols for…

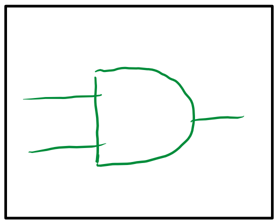
3-input OR:



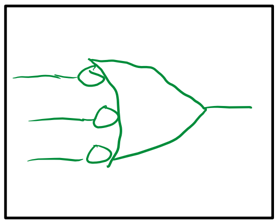
2-input NAND:



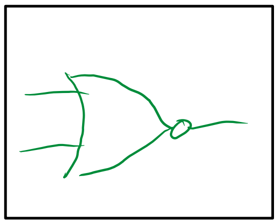
2-input AND:



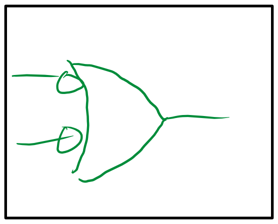
3-input BOR



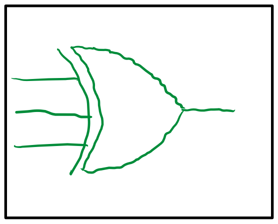
2-input NOR:



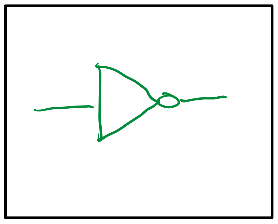
2-input BOR:



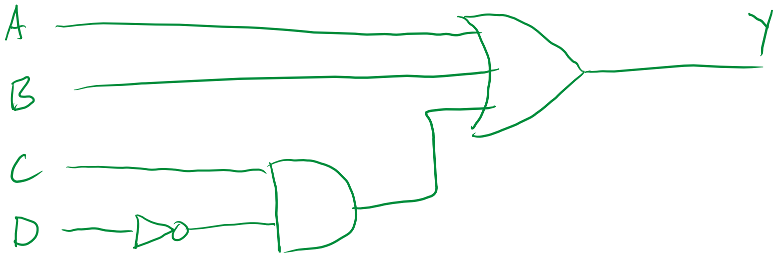
3-input XOR:



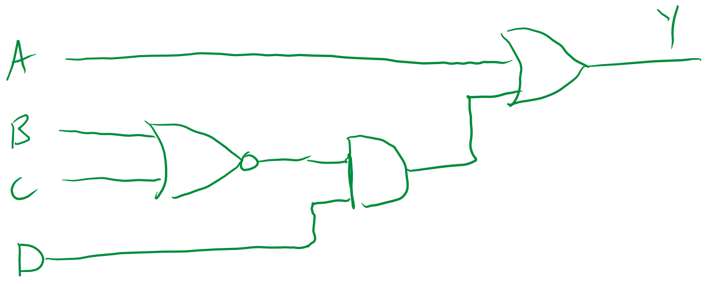
NOT:



2) Draw a gate-level schematic that implements $Y = A + B + C \cdot \overline{D}$. Use a minimal number of gates.



3) Draw a gate-level schematic that implements $Y = A + \overline{B + C} \cdot D$. Use a minimal number of gates.



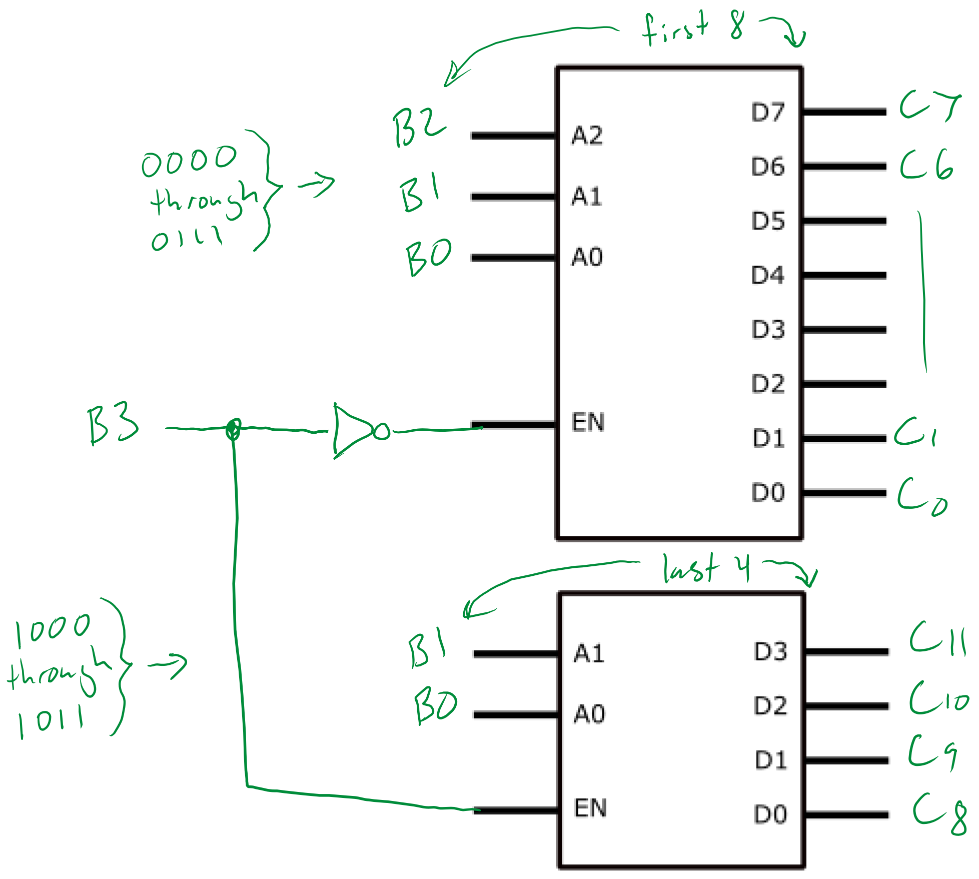
4) You have a three-input OR gate. One input is driven to ‘1’, one input is driven to’0’, and one input is not driven. What is the value at the gate’s output?

Forcing

1

5) You need a 4-to-12 line decoder and you happen to have a 2-to-4 and a 3-to-8 decoder (both with enable inputs). Since four inputs could be decoded to 16 outputs, in this 4-to-12 decoder, inputs “0000” through “1011” will be used, and “1100” (12) through “1111” (15) can be considered “don’t cares” and produce any output.

The two decoders are drawn below. Use them to create a 4-to-12 line decoder with inputs B3-0 and outputs C11-0. You can label signals as the needed inputs (B3, B2, B1, B0) or label them with 0 or 1 if they need to be constant. Be sure to label the outputs (C11, C10, […], C0). You can draw wires as needed, and you can use primitive gates (AND, OR, NOT) if needed. (You do not need very many additional gates. If you start designing a bunch of logic, you’re doing something wrong.)



If you were completing the truth table for the following expression:

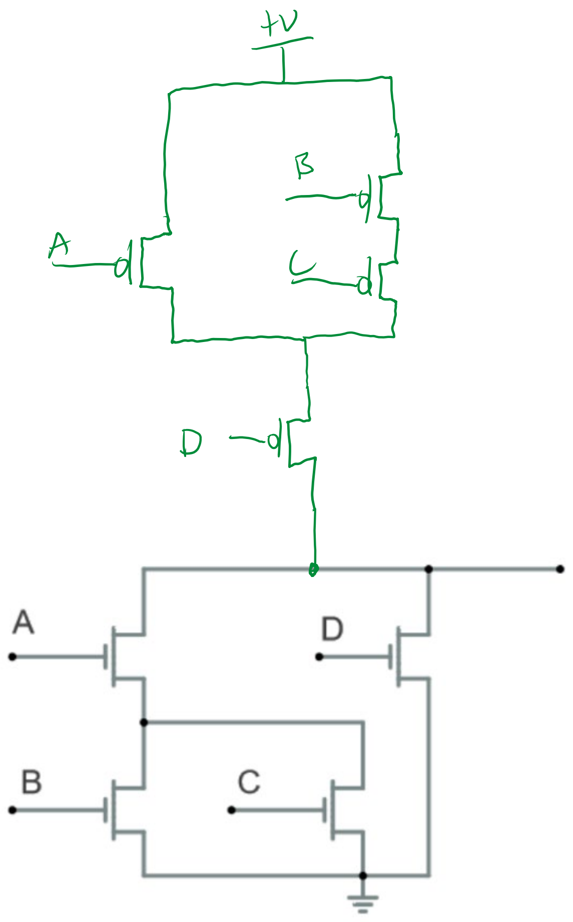
$$K = \overline{A + B \cdot \overline{C \cdot D}} + (E + F \cdot G)$$

1a) How many rows would the truth table have? $2^7 = 128$

1b) Explain how you can complete half of the truth table by setting one of the inputs to a particular value. Provide the input, the value to make it, and the resulting value for K.

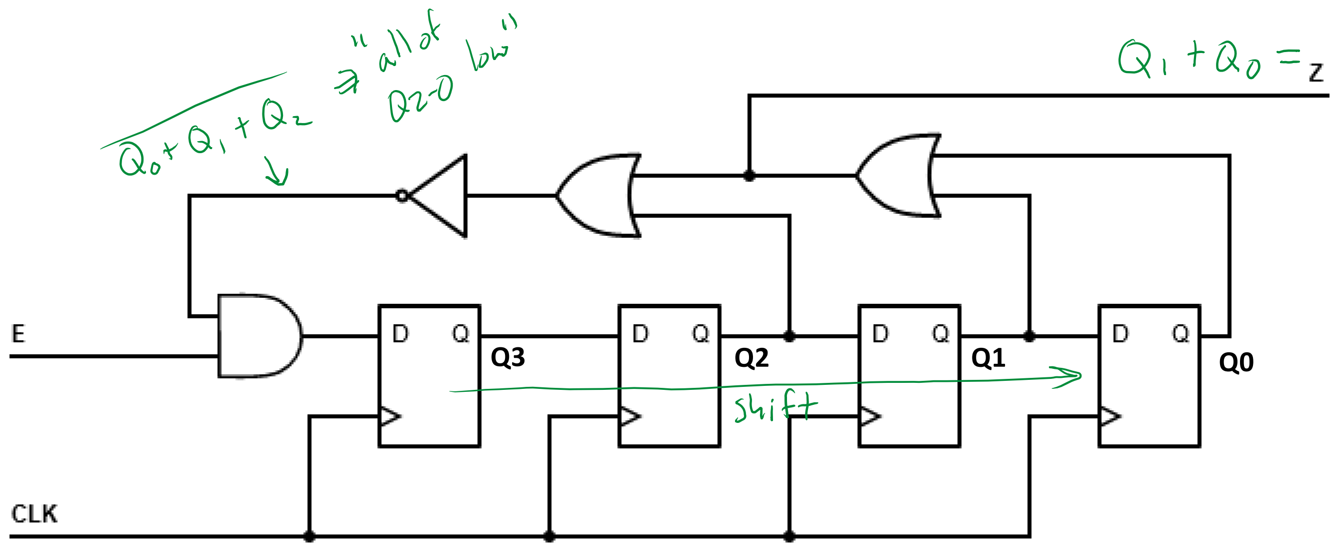
$E=1$ makes $K=1$

2) Attach an appropriate pull-up network to this pull-down network to create a valid CMOS gate:



3) What is the result if a barrel-shift right by 3 is performed on the following:

1001|101 1011001



4a) Complete the blank entries in the following transition table based on the state machine above. The state "names" are the same as the state encodings Q_3-Q_0.

Current State	Input E	Next State	Output Z
0000	1	1000	0
0000	0	0000	0
1000	1	1100	0
0001	1	0000	1
1101	0	0110	1
1001	1	0100	1

4b) What is the maximum frequency at which this state machine can be safely clocked? Express your answer as a mathematical expression in terms of the parameters T_{pOR}, T_{pAND}, T_{pNOT}, T_{pDFF}, T_{SU}, and T_H.

$$1/(T_{pDFF} + 2T_{pOR} + T_{pNOT} + T_{pAND} + T_{su})$$

4c) After a rising clock edge, how long do you have to wait before the output (Z) is guaranteed to be correct? Express your answer in terms of the same parameters.

$$T_{pDFF} + T_{pOR}$$

4d) If a rising clock edge occurs at time t=0, during what period of time should input E not change? Express your answer as a range in terms of the timing parameters; e.g. $-(T_H)$ to $+(T_H+T_{SU}+T_{PNOT})$.

$$-(T_{su} + T_{pAND}) \text{ to } +(T_H - T_{pAND})$$

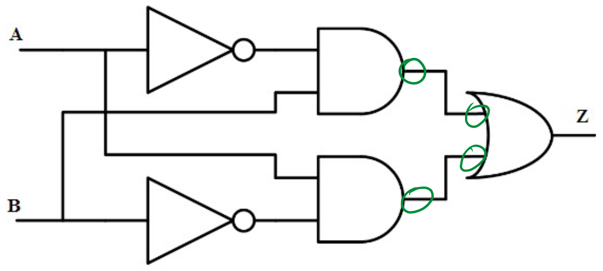
1) Provide an example of why a "don't care" would exist in the output column of a truth table.

That combo of inputs should never happen, or are outside the specified functionality, or cause something else to happen such that this output doesn't matter.

2) Assuming that the following is a transition table for a Moore state machine, fill in the empty box.

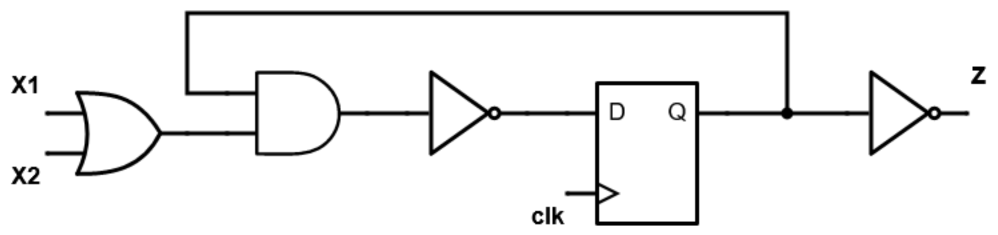
Q1	Q0	X	Q1+	Q0+	Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	1	1
1	1	1	0	0	1

3) Write a Boolean expression that is logically equivalent to the circuit below but that only involves NANDs and inverters. In other words, if I took your expression and directly drew a gate schematic from it, I would only draw NAND gates and inverters.



$$Z = \overline{\overline{\bar{A} \cdot B} \cdot \overline{A \cdot \bar{B}}}$$

Consider the following circuit:



4a) Does this circuit implement a Mealy or a [✓ Moore] state machine?

4b) Complete the transition table for the state machine:

Q	X1	X2	Q+	Z
0	0	0	1	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

next state 1 if current state 0 or both inputs 0.

4c) Using the parameters T_{pOR}, T_{pAND}, T_{pNOT}, T_{pDFF}, T_{SU}, and T_H, what is the maximum safe clock frequency for this state machine?

$1/(T_{pDFF} + T_{pAND} + T_{pNOT} + T_{SU})$

1a) Using the truth table below, create a K-map and solve for a minimal sum-of-products expression.

A	B	C	D	Y
0	0	0	0	X
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	X
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	X
1	1	0	1	X
1	1	1	0	0
1	1	1	1	1

Label the rows and columns of the K-map appropriately.

AB \ CD	00	01	11	10
00	X	1	1	1
01	X	1	1	0
11	X	X	1	0
10	0	0	1	1

All prime implicants

← essential

could be $\bar{A} \cdot D$

1b)

Y = $\bar{B} \cdot C + BD + \bar{A}\bar{B}$

1c) Regardless of whether or not you circled them or used them in the SoP expression, are there any **non-essential prime implicants** in the K-map above? If so, what are the expressions that represent them?

All of them except $\bar{B} \cdot C$.

$\bar{A}\bar{C}$ $\bar{A}\bar{B}$

$\bar{A}D$ CD

$B\bar{C}$

BD

2) Create a state diagram (in the ASM format we used in this class) for a Moore state machine that implements the following behavior. Assume that it is destined to be implemented in digital hardware, so e.g. it has a clock. For all signals, “high”, “active” and “1” are synonymous.

The state machine has two inputs, “one” and “two”, and one output, “ok”. The inputs indicate how many items came out of a machine that clock cycle, which can be none, one, or two (never three).

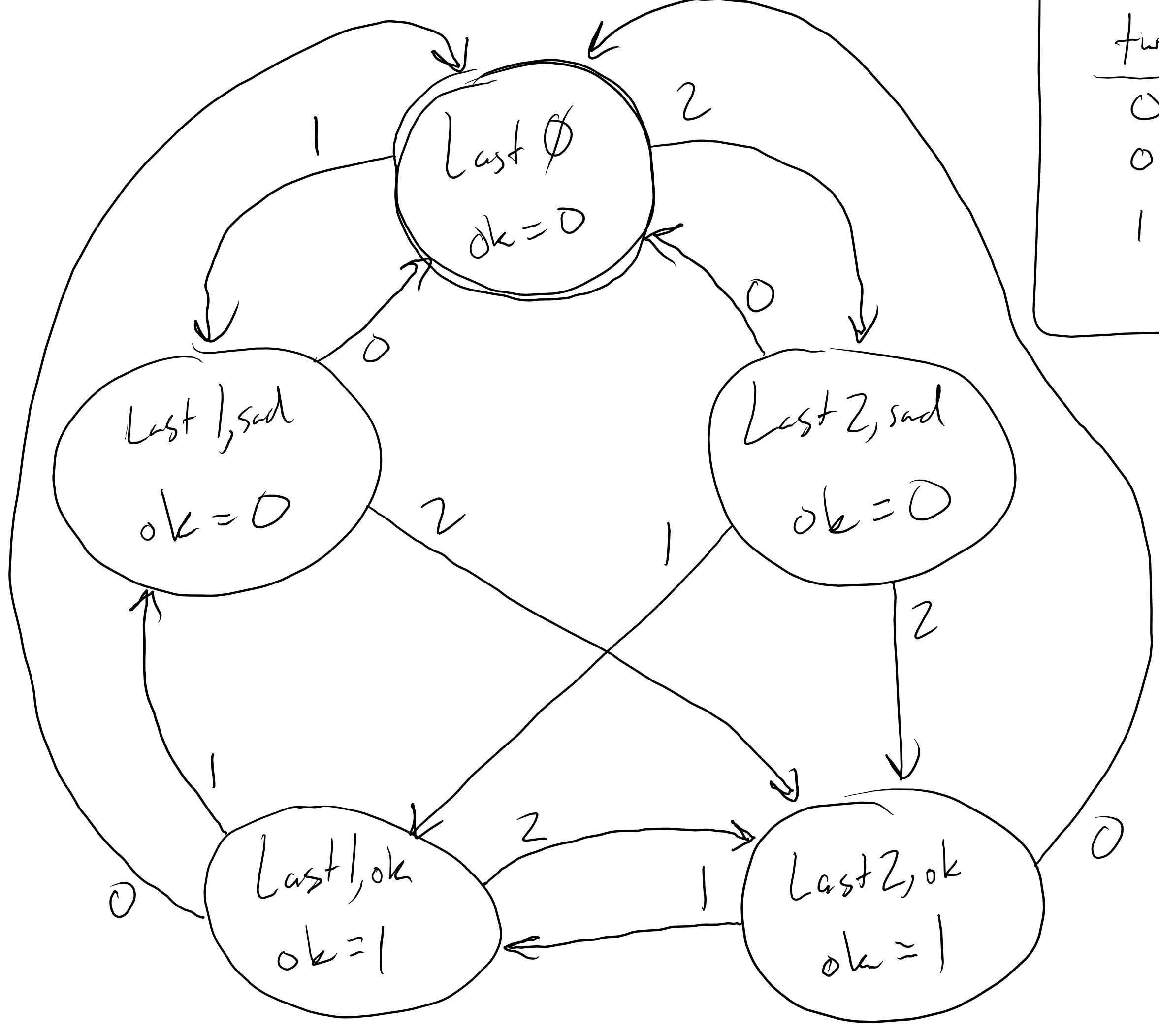
As long as the sum of items from two cycles (the “current” one and the previous one) is three or more, “ok” should be active. If not enough items are coming out, “ok” should be inactive.

As an example, if input “two” is always active (indicating that two items are coming out each cycle), the output should be active since that’s always four items in two cycles. If the inputs oscillate each cycle between “one” being active and “two” being active, the output would be high (since that’s always three items in two cycles). If the inputs oscillate each cycle between “one” being active and neither being active, the output would be low (since that’s only one item in two cycles).

Assume that at reset, there have not been any items, and thus the output should be inactive.

Legend:

two	one	arc label
0	0	0
0	1	1
1	0	2



1) Provide the **decimal value** if these binary numbers are **interpreted as** unsigned, sign-magnitude, 2's complement, and fixed-point numbers in the specified format.

	unsigned	sign-magnitude	2's complement	unsigned fixed-point (NNNN.NN)
010001	17	17	17	4.25
100111	39	-7	-25	9.75
100100	36	-4	-28	9.0

2) Perform the following conversions to hexadecimal.

$01110010_2 = \underline{\quad 72 \quad}_{16}$

$101110010_2 = \underline{\quad 172 \quad}_{16}$

$0000101101110010_2 = \underline{\quad 0B72 \quad}_{16}$

3) Perform the following unsigned operations. Restrict the result to 5 bits, and indicate if overflow occurred.

```
 01001
+10001
------
 11010
```

overflow?
y / [✓ n]

```
 01100
+10101
------
 00001
```

overflow?
[✓ y] / n

4) Perform the following 2's complement operations. Restrict the result to 5 bits, and indicate if overflow occurred.

```
 10101
+01101
------
 00010
```

overflow?
y / [✓ n]

```
 01100
-11101
------
 01111
```

overflow?
y / [✓ n]

5) Perform the following sign-magnitude operation. Restrict the result to 5 bits, and indicate if overflow occurred.

```
 01001
+10001
------
 01000
```

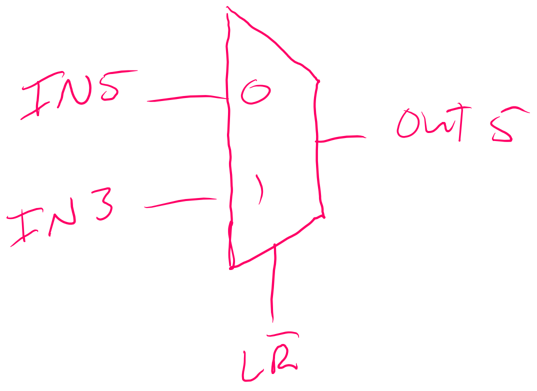
overflow?
y / [✓ n]

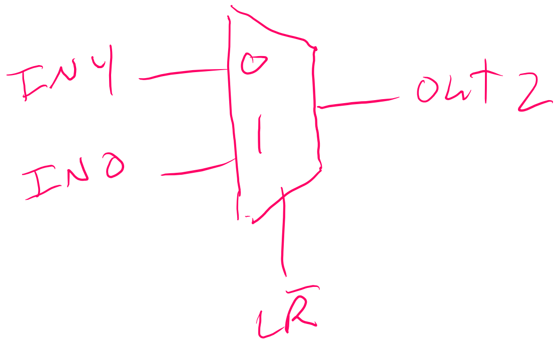
6) Assume that there is a 6-signal bus named IN_{5-0}. You need to create logic for OUT_{5-0}, where OUT is the result of arithmetically-shifting IN by two places. However, whether the shift is left or right is selected by another signal, $L/\overline{R}$.

For this problem, you only need to create logic for OUT_5 and OUT_2.

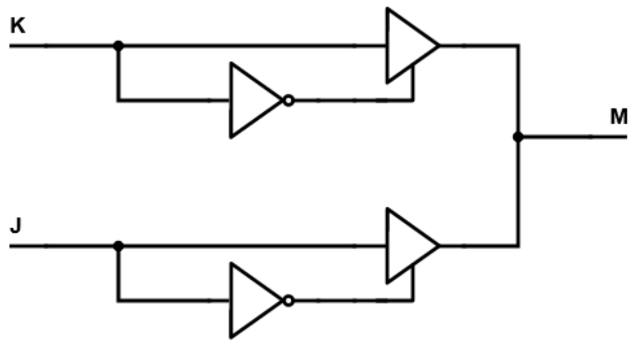
To summarize, you have access to signals IN_{5-0} and $L/\overline{R}$ and you need to create logic for OUT_5 and OUT_2 as specified above. You can use primitive gates and any of [encoders, decoders, multiplexers, demultiplexers] if desired.

$L\overline{R}$	OUT5	OUT2
0	IN5	IN4
1	IN3	IN0



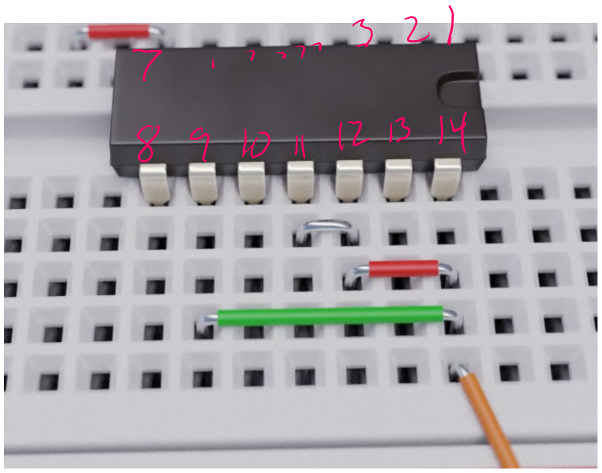


1) Complete the truth table for this circuit.



K	J	M
0	0	0
0	1	0
1	0	0
1	1	Z

2) In the following image, assume that the wire going off the bottom connects to Vcc. Which pins on the chip are connected to Vcc? Provide the pin numbers as you would find them in the chip's datasheet.



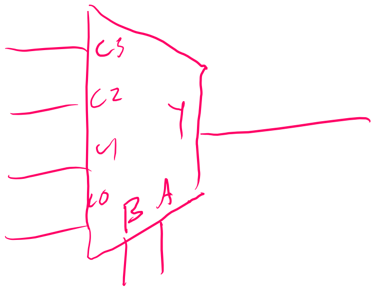
14, 12, 11, 9

3) The following table is from a chip's datasheet. Ignore the strobe signal. What logic device does this chip contain?

Function Table

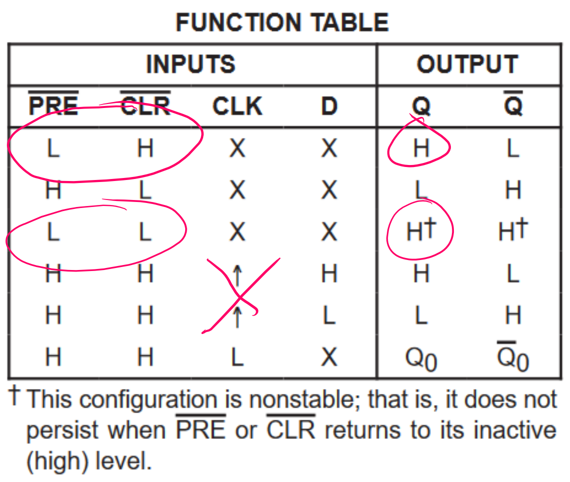
Select Inputs		Data Inputs				Strobe	Output
B	A	C0	C1	C2	C3	G	Y
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

0
1
2
3



multiplexer

5) This is table is from the datasheet from the type of flip-flop that you used in the second lab. If you can't cause any clock edges, how can you force the Q output to be high?



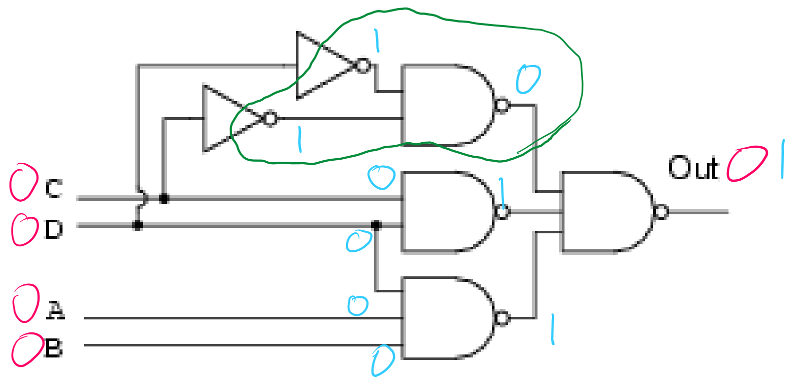
FUNCTION TABLE

INPUTS				OUTPUT	
$\overline{PRE}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H†	H†
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$

† This configuration is nonstable; that is, it does not persist when $\overline{PRE}$ or $\overline{CLR}$ returns to its inactive (high) level.

Make pre low, preferably with clr high.

6) Assume you have built a circuit implementing the following logic and are currently testing it.



Known in pink
Expected in blue

You drive the inputs all low, and your test equipment (e.g. a myDAQ) indicates that the output is low. You do some debugging and determine that the problem must be in the circuit somewhere (i.e. it's not the myDAQ, connections to the myDAQ, power supply, etc.) so you start testing nodes in the circuit. Which node in the circuit is a good node to probe first, and why?

The output of the top NAND or either output of the NOTs, because any of those being wrong could be directly responsible for the incorrect output.