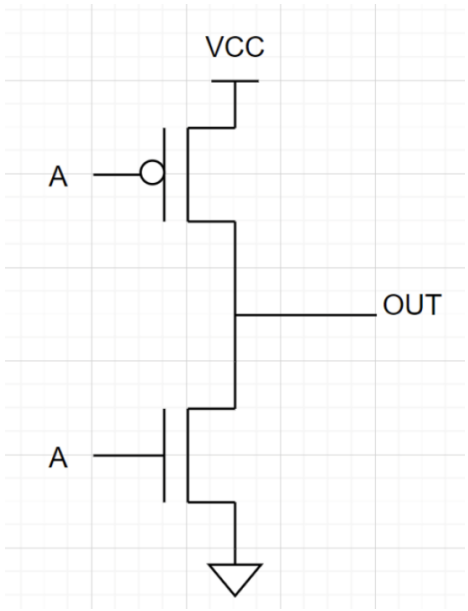
HKN ECE 120 Midterm 2 Worksheet Solutions

CMOS Logic

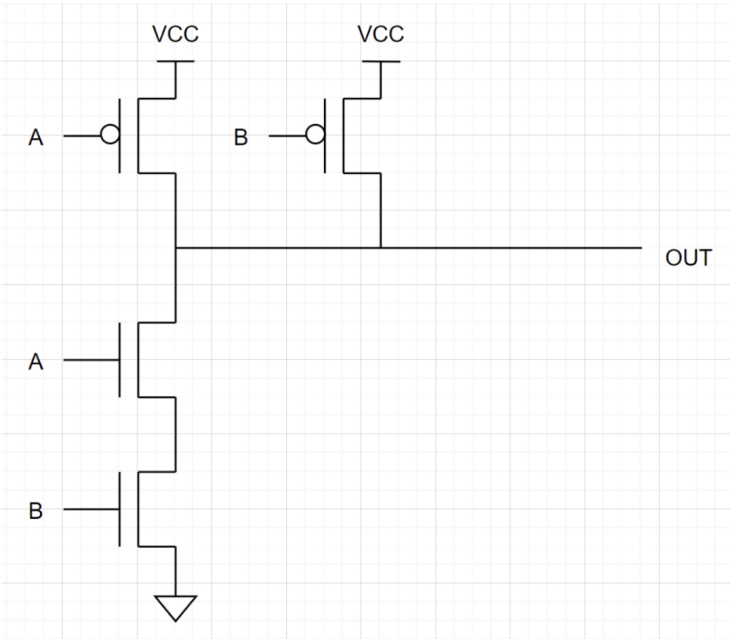
Problem 1

Draw the CMOS network for the following expressions:

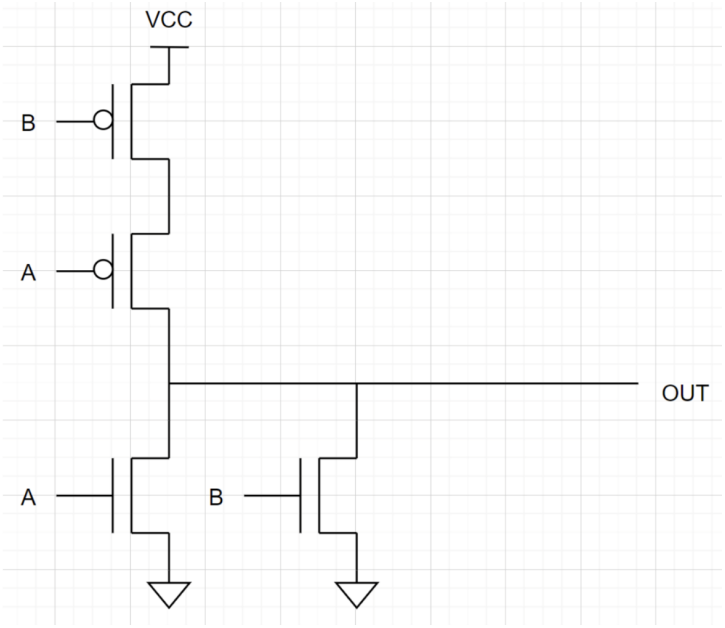
a. $Z = A'$



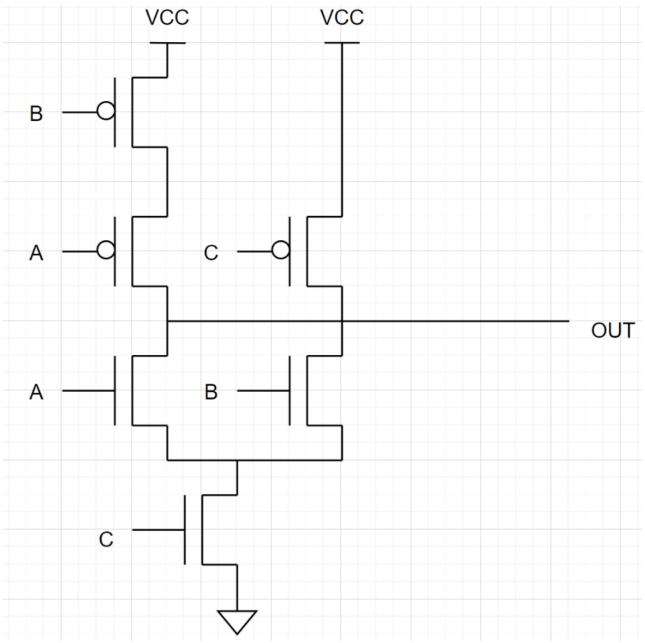
b. $Z = (A \cdot B)'$



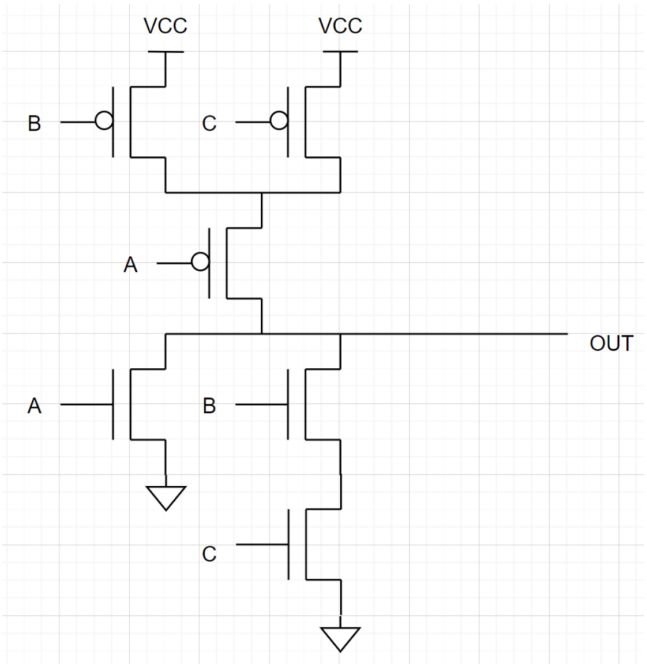
c. $Z = (A + B)'$



d. $Z = ((A + B) \cdot C)'$



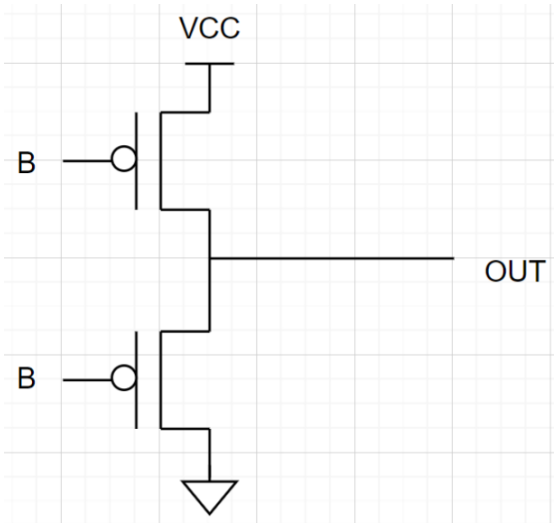
e. $Z = ((B \cdot C) + A)'$



Problem 2

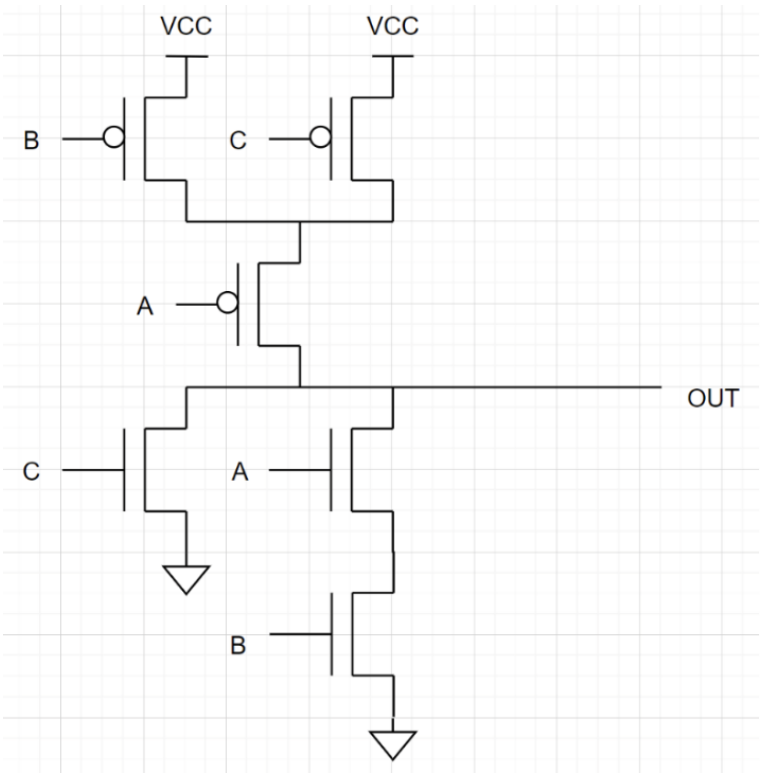
Are these valid CMOS networks? If not, explain why.

a.



No. Notice that there is a P-type MOSFET used at the bottom. The output is not connected (left floating) when $B = 0$, and power is connected (shorted) to ground when $B = 1$.

b.



No, if $A = 0$, $B = 0$, $C = 1$, power is connected (shorted) to ground, and if $A = 1$, $B = 0$, $C = 0$, the output is left floating.

Boolean Expressions, Algebra, & Optimization

Problem 1

Simplify the following expressions:

a. $F(A, B, C) = (A + B) \cdot (B' + C) \cdot (A + C)$
Consensus: $F = (A + B) \cdot (B' + C)$

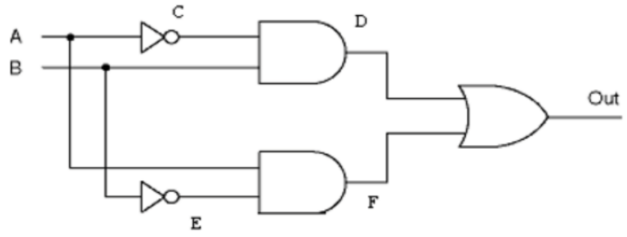
b. $F(A, B, C) = A'BC + ABC + ABC' + AB'C$
Distributive: $F = A'BC + A(BC + BC' + B'C)$
Distributive: $F = A'BC + A(B(C + C') + B'C)$
Complementarity: $F = A'BC + A(B + B'C)$
Distributive: $F = A'BC + A(B + B')(B + C)$
Identity: $F = A'BC + A(B + C)$
Distributive: $F = A'BC + AB + AC$
Distributive: $F = B(A'C + A) + AC$
Distributive: $F = B(A' + A)(C + A) + AC$
Complementarity + Identity + Distributive: $F = BC + BA + AC$

c. $F(A, B, C) = A' \cdot (A + B) + (A + B) \cdot (A + B')$
Distributive: $F = A'A + A'B + AA + AB' + AB + BB'$
Idempotence/Complementarity: $F = A'B + A + AB' + AB$
Distributive: $F = A \cdot (1 + B) + B \cdot (A + A')$
Idempotence/Identity: $F = A + B$

Problem 2

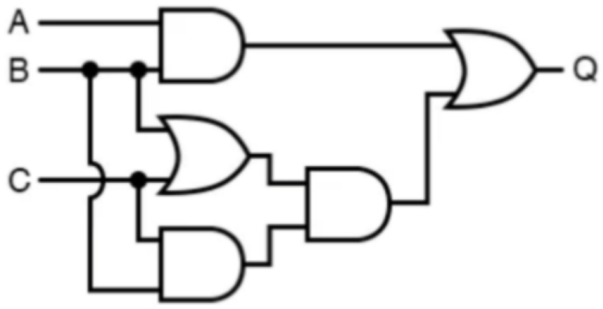
True or False:

a. The delay for the following expression is 2.



True. The maximum delay to get from an input to an output is through 2 gates, not including inverters. (we do not consider NOT gates as part of area or delay for ECE 120 area/delay heuristic)

b. The area for the following expression is 7.



False. The area heuristic is calculated as the number of literals in the base expression plus the number of gates. There are 6 literals that feed directly into gates, and 5 gates in total, so the area is 11.

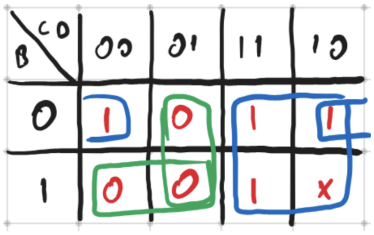
K-maps, SOP & POS Expressions

Problem 1

Use K-maps or simplification to find the minimal SOP and POS expressions for the following, then draw them in NAND/NOR form:

a.

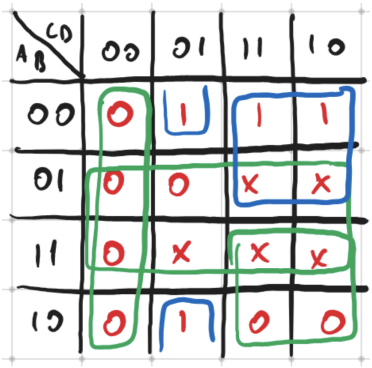
B	C	D	Z
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	1



SOP: $Z = C + B'D'$
POS: $(B' + C) \cdot (C + D')$

b.

A	B	C	D	Z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	X
0	1	1	1	X
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



SOP: $Z = A'C + B'C'D$
POS: $B' \cdot (C + D) \cdot (A' + C')$

Adders, ALUs, and Bit-Slicing

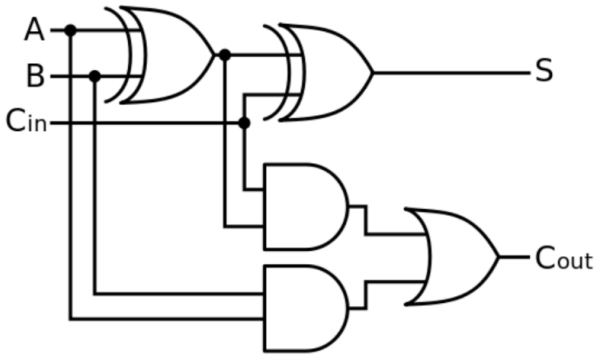
Problem 1

Remember that the logical output for one slice of a full adder is:

A	B	Cin	Cout	S
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

a. Find an expression for Cout and S using AND and OR gates.
Multiple possible solutions. Use K-maps or simplify the boolean expressions.
$S = (AB')' \cdot Cin + (AB') \cdot Cin' + (A'B)' \cdot Cin + (A'B) \cdot Cin'$
$Cout = AB + AC + BC$

b. Consider the digital circuit:

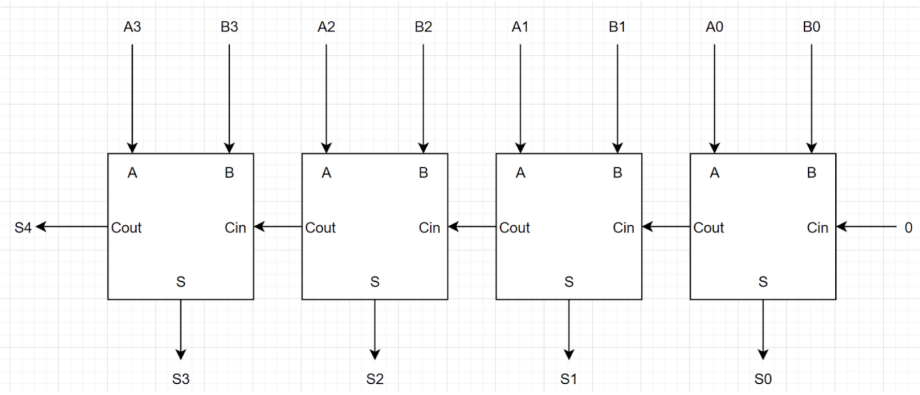


Verify that it is equivalent to your expression.
Trace out the inputs from the truth table, you should get the same results for S.

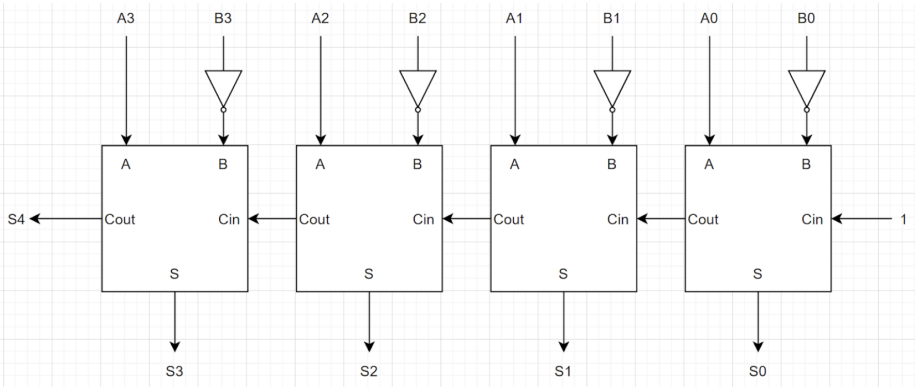
Problem 2

Build each of these circuits using only adders, inverters, and fixed inputs (1 or 0). Do not account for overflows unless otherwise noted.

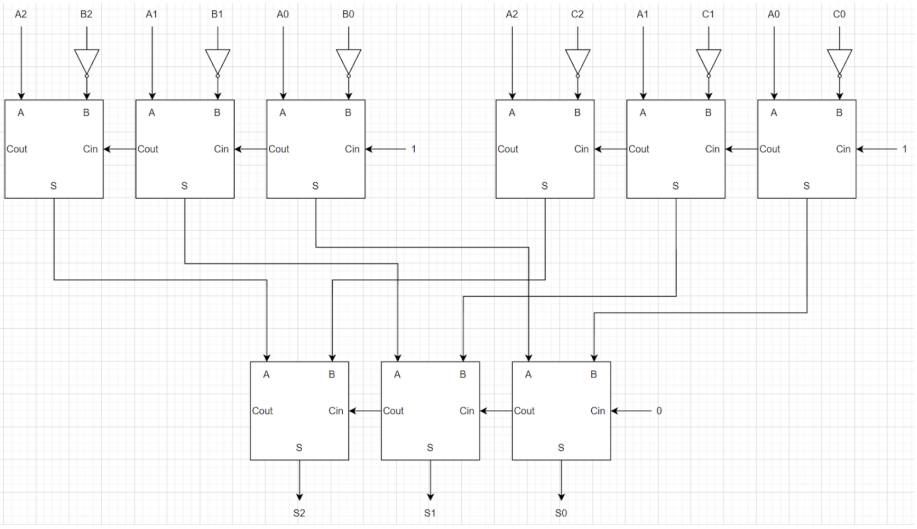
a. For two 4-bit unsigned integers $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$, calculate the 5-bit unsigned integer $S_4S_3S_2S_1S_0 = A + B$.



b. For two 4-bit two's complement integers $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$, calculate the 4-bit two's complement integer $S_3S_2S_1S_0$ where $S = A - B$.



c. For three 3-bit two's complement integers $A_2A_1A_0$, $B_2B_1B_0$, $C_2C_1C_0$, calculate $S_2S_1S_0 = A + A - B - C$.
$S_2S_1S_0 = (A - B) + (A - C)$



Problem 3

Assume an ALU that takes two 8-bit integers A and B as input and can calculate AB (AND),

a. How many bits is the output S?
8 bits.

b. How many bits must the control signal be?
At least 2 bits. There are 3 operations, which is less than 2^2.

c. Is the ALU logically complete? Assume you have 0 and 1 available.
(multiple possible ways to show this) We know that the set AND, OR, NOT is logically complete (Lumetta 1.4.5), so we want to show that we can produce these operations using the operations we are given. We already have AND, so we want to make OR and NOT. For NOT, we know that any $A \bigoplus 1 = A'$, and we are given logical 1. For OR, one way to do this is by adding $(A \bigoplus B)+(AB)$.

A	B	$A \bigoplus B$	AB	$(A \bigoplus B) + (AB)$
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	1

We can see by the truth table that this is equivalent to OR. We also don't have to worry about the carry as at most one of its inputs are 1. Thus, the set AND, XOR, ADD is logically complete.

d. (Optional) Draw out the full circuit diagram for this ALU, using basic logic gates, MUXs, decoders, and full adders.

Problem 4

In 50 or fewer words, explain the advantages and disadvantages of bit-slicing over optimizing for many variables.
Bit-slicing allows for per-bit design, which may be simpler to implement, and usually results in much smaller area in exchange for a larger delay. Optimizing for many variables can yield results in as little as a delay of 2 (two-level logic), but uses many more gates and much larger area.

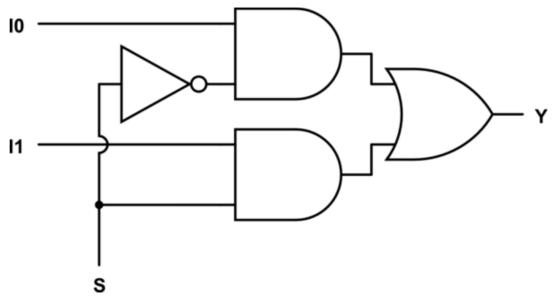
Multiplexers

Problem 1

Write down the truth table for a 2-to-1 multiplexer. Write an expression for the output, and implement the 2:1 MUX using AND gates, OR gates, and inverters.

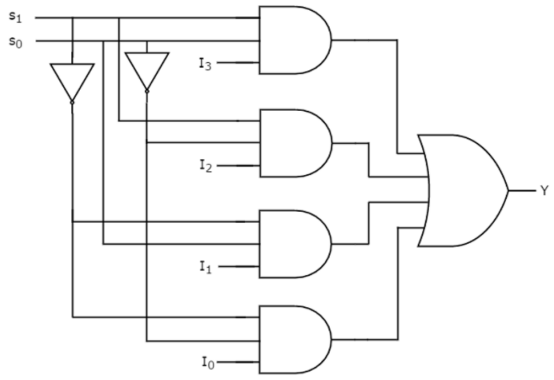
I0	**I1**	**S**	**Y**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$\mathbf{Y} = (\mathbf{I_0 S'} + \mathbf{I_1 S})$$



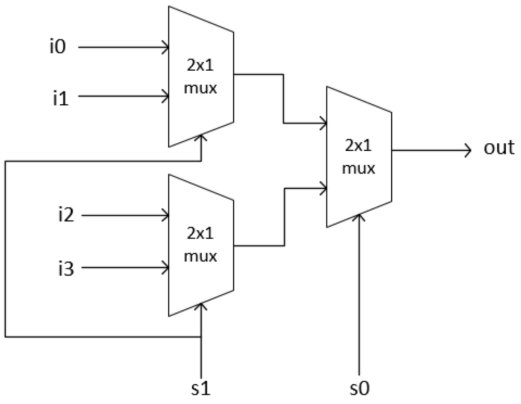
Problem 2

Using the same process (you may not need the truth table), implement a 4:1 MUX using AND gates, OR gates, and inverters.



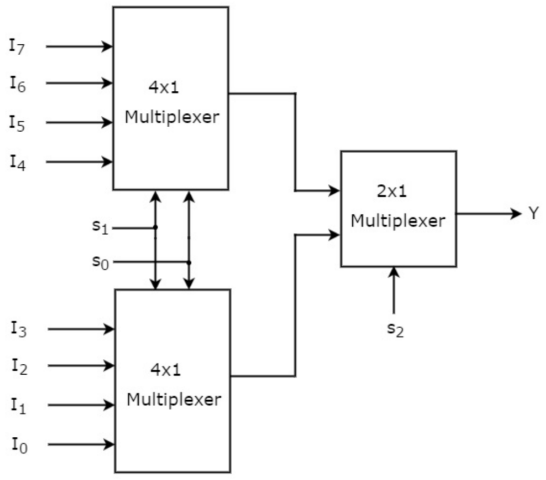
Problem 3

Implement a 4:1 MUX using three 2:1 MUXs.



Problem 4

Implement a 8:1 MUX using two 4:1 MUXs and a 2:1 MUX.



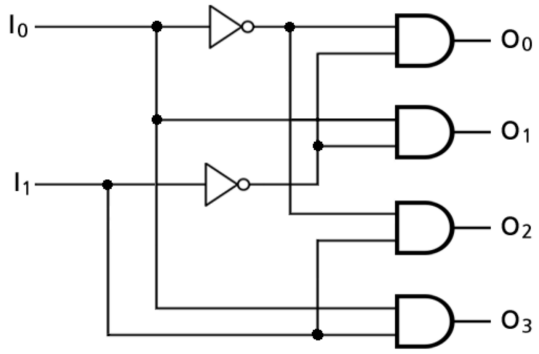
Decoders

Problem 1

In 20 words or less, what does a decoder do?
Take an N-bit integer I and output 1 on D_i and 0 on all other $2^N - 1$ outputs.

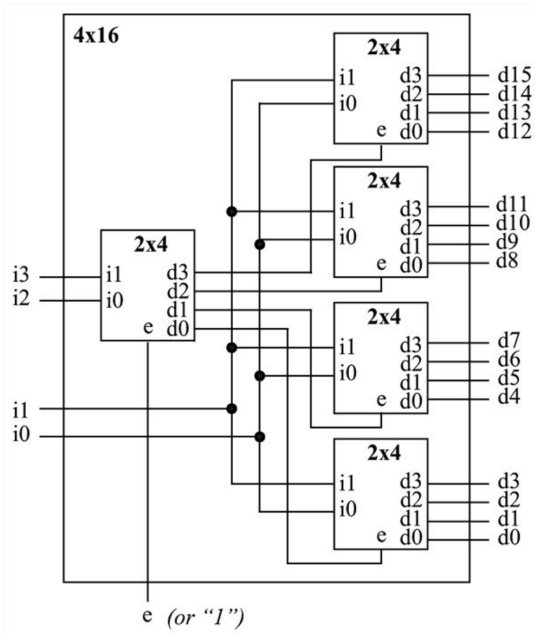
Problem 2

Implement a 2x4 decoder using AND gates, OR gates, and inverters.



Problem 3

Implement a 4x16 decoder using five 2x4 decoders. (Hint: you should use the ENABLE pin)



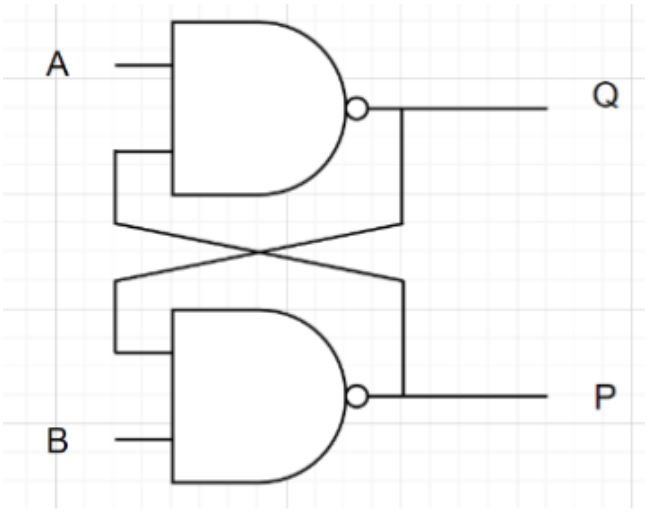
Note: you can also do this without the ENABLE pin by adding an AND gate for every output.

Latches & Flip-Flops

Problem 1

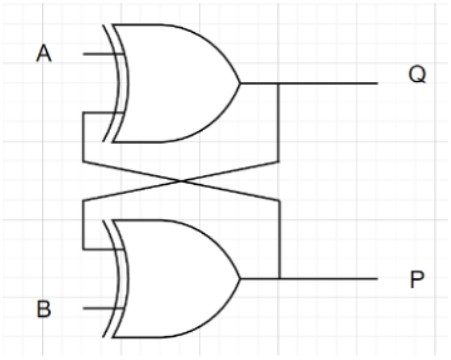
Find the stable states of the following latches:

a.



A	B	Q	P
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0	1
1	1	1	0

b.



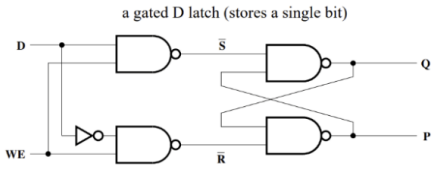
A	B	Q	P
0	0	0	0
0	0	1	1
1	1	1	0
1	1	0	1

c. Which latch is more viable for data storage?

Try to store some data on the XOR latch: Falstad Simulation (click the inputs)
When we want to store data, we can only toggle one input at a time. (it's impossible to synchronize two inputs changing at exactly the same time)
When we enter any unstable state, since the circuit flickers, it's impossible to predict what Q and P end up storing. Thus, it's not actually able to reliably store anything. Thus, the NAND latch is more viable.

Problem 2

Explain the process of storing a bit of data in a D latch.



When we want to store data, we set WE (write-enable) to high, which allows the two NAND gates on the left to toggle. Then, we

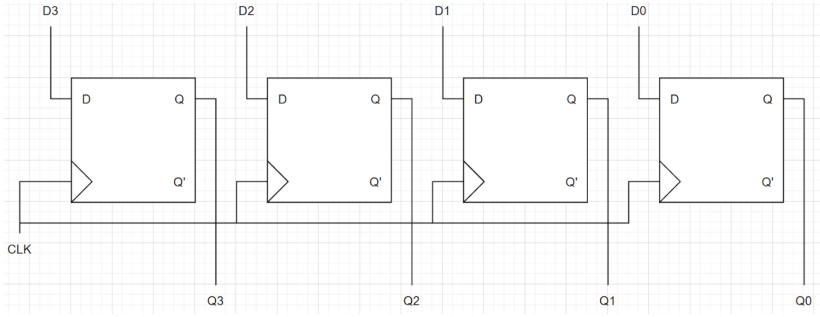
toggle D, and Q mirrors the signal on D. Then to stop writing, we set WE to low.

Problem 3

Explain the difference between a D latch and a D flip-flop.
The output of a D latch updates as long as the WE (write-enable) signal is high. On the other hand, the output of a D flip flop only updates upon the rising/falling edge of the clock. (rising edge for a positive-edge triggered D flip flop and falling edge for a negative-edge triggered D flip flop).

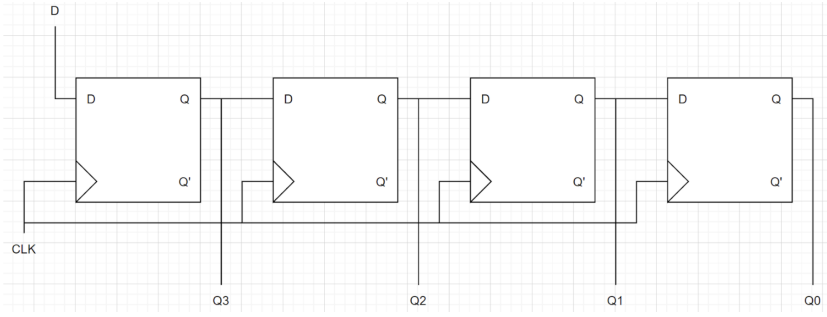
Problem 4

Implement a 4-bit register using D flip-flops



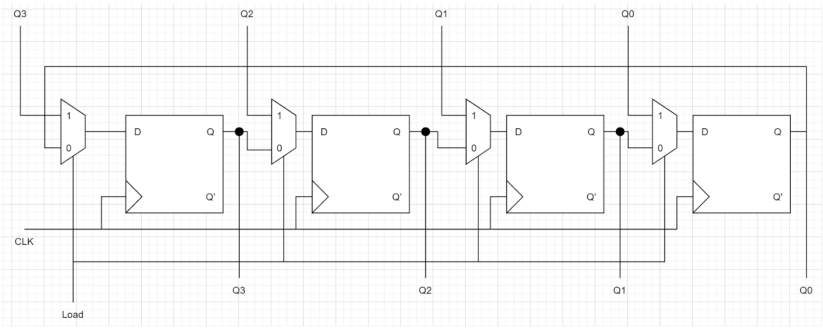
Problem 5

Implement a 4-bit shift register with MSB load using D flip-flops.



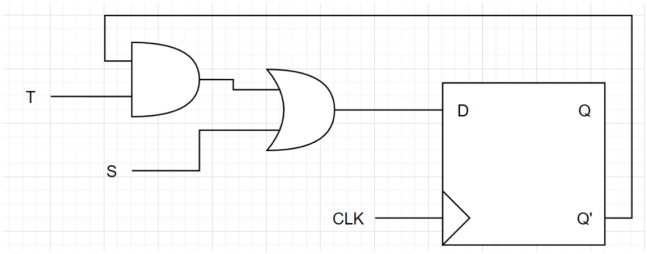
Problem 6

Implement a 4-bit circular shift register with 4-bit load using D flip-flops.



Problem 7

Consider this digital circuit below.



Assume at the start, $Q = 0$. Trace and fill out the table below. T and S signals are given in the table.

Clock Cycle #	Q	T	S
0	0	1	0
1	1	1	0
2	0	0	0
3	0	0	0
4	0	0	1

Problem 8

What are timing hazards? What are a few ways we can design around them? (Lumetta 2.6.3-2.6.5)

When we implement a circuit, since gates take time to switch, there may be glitches in our output temporarily (static hazards) or may bounce between outputs until settling to the correct, stable state (dynamic hazards).

We can design around them by adding additional gates for stability or assign the outputs of unused states to ensure smooth switching.

Serialized Design

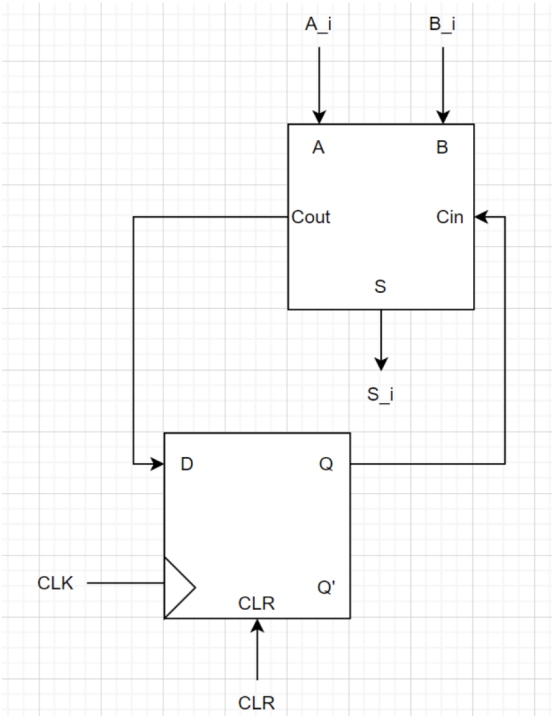
Problem 1

Consider one bit-slice of a full adder.

a. How many bits need to be passed between each bit slice? What are they?
1 bit. We need to pass the carry bit between each bit slice.

b. If we wanted to serialize this design and add 1 bit at a time, how many D flip-flops would we need to store these signals?
1. We need 1 DFF for each signal we need to store.

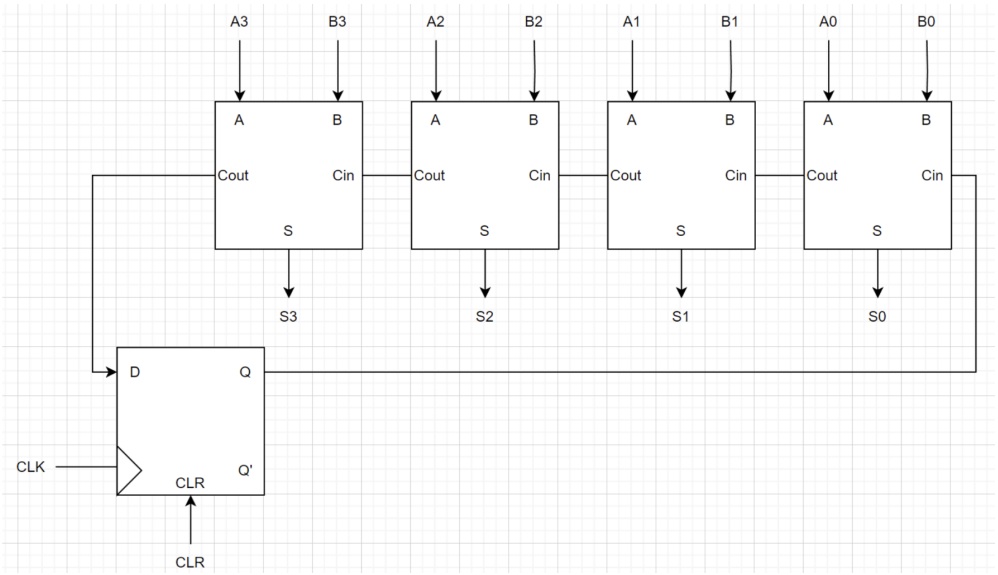
c. Implement a serialized binary adder, adding 1 bit at a time.



Note: the CLR (clear) signal is used when we want to clear the carry bit between adding different sets of numbers.

d. Assume we instead wanted to add 4 bits (1 hexadecimal) at a time. Would any signals be different?
No, we still only need to pass one carry bit, the carry bit of the MSB, to the next clock cycle's LSB.

e. Implement a serialized binary adder, adding 4 bits at a time.

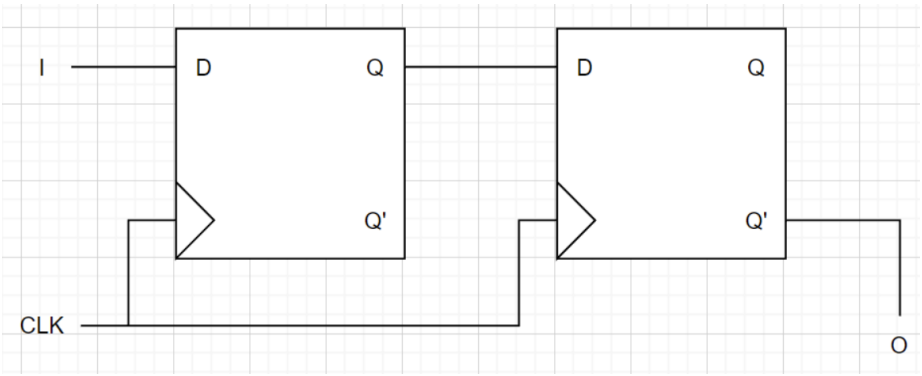


Problem 2

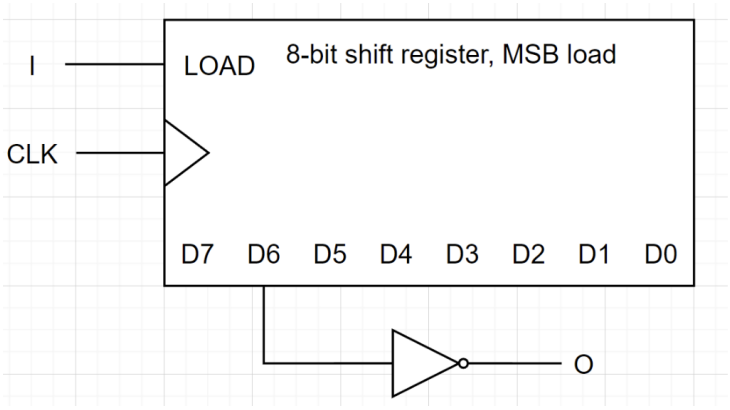
Consider a serialized circuit that takes a sequence of bits, 1 bit at a time, and outputs the same sequence of bits but is delayed by 2 bits and inverted. For example, assuming inputs and outputs are taken at the start of each clock cycle, if the input is 100101101100, then the output is XX0110100100.

a. How many bits do we need to store between each clock cycle?
2 bits.

b. Implement this circuit using D flip-flops.



c. Can we also implement this circuit using a shift register?
Yes:



Problem 3

In 50 words or less, what are the advantages and disadvantages of serialization over bit-slicing?

For larger sequences, serialized designs take much less area than bit-sliced designs. However, serialized designs are much slower, as they depend on the speed of a clock signal, while bit-sliced design is only dependent on gate delay.