Department of Electrical and Computer Engineering

The University of Texas at Austin

EE 306, Fall 2021
Problem Set 2
Due: **September 27**, before class
Yale N. Patt, Instructor
TAs: Sabee Grewal, Ali Fakhrzadehgan, Ying-Wei Wu, Michael Chen, Jason Math, Adeel Rehman

Note: This problem set is significantly longer than the previous problem set. We encourage you to start early, to work in study groups, and come to office hours!

Instructions: You are encouraged to work on the problem set in groups and turn in one problem set for the entire group. **The problem sets are to be submitted on Gradescope**. Only one student should submit the problem set on behalf of the group, but everyone should create a gradescope account and be tagged on the homework.

1. Note: This problem is repeated from Problem Set 1 since we did not cover hexadecimal representation before it was due. This question will not be graded on the previous problem set.
   (Adapted from 2.50)
   Perform the following logical operations. Express your answers in hexadecimal notation.

   a. xABCD OR x9876 = **xBBFF**
   b. x1234 XOR x1234 = **x0000**
   c. xFEED AND (NOT(xBEEF)) = **x4000**

2. (Adapted from problem 2.55 in the textbook)

   We have represented numbers in base-2 (binary) and in base-16 (hex). We are now ready for unsigned base-4, which we will call quad numbers. A quad digit can be 0, 1, 2, or 3.

   a) What is the maximum unsigned decimal value that one can represent with 3 quad digits?

      63; (4 ^ 3) - 1

   b) What is the maximum unsigned decimal value that one can represent with n quad digits? (*Hint*: Your answer should be a function of n.)

      (4^n) -1

c) Add the two unsigned quad numbers: 023 and 221.

   310

d) What is the quad representation of the decimal number 42?

   222

3. What is the largest positive normalized number that can be represented using the IEEE 32-bit Floating Point standard?

   3.4028235 * 10^38

4. What is the largest positive number that can be represented in a 32-bit 2's complement scheme?

   2,147,483,647

5. Professor Patt is trying to decide when he should shave his beard. Your job is to design a logic circuit whose output Y is equal to 1 when Professor Patt should shave his beard, and 0 when he should not. The circuit will receive three input variables (A, B, C) that answer three different yes/no questions (1=yes, 0=no).

   A<=Has Professor Patt been uncomfortably warm this summer?
   B<=Does Professor Patt want a new, fresher look?
   C<=Are beards "cool"?

   We think that Professor Patt should shave his beard if he has been uncomfortably warm this summer. He should also shave his beard if he wants a new fresher look and beards are not "cool".
   Write the logic equation for Y in terms of A,B,C that solves this problem, and draw the gate-level diagram.

   Y = A OR (B AND (NOT C))

6. (Adapted from problem 3.17 in the textbook)

   Draw a transistor-level diagram for a three-input AND gate and a three-input OR gate. Do this by extending the designs from Figures 3.6a and 3.8a. (Figures can be found in the book on pages 64 & 65 respectively).

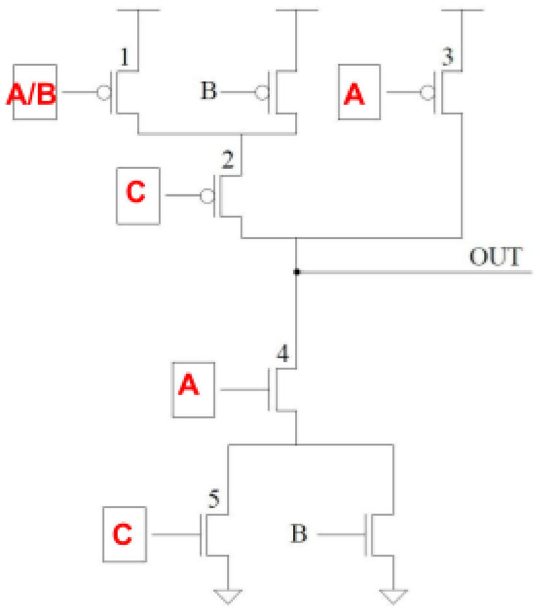
   For 3-input OR gate: Add one P-type transistor from Vdd to C, in series to the current gate, and one N-type from C to GROUND, in parallel to the current gate.
   For 3-input AND gate: Add one P-type transistor from Vdd to C, in parallel

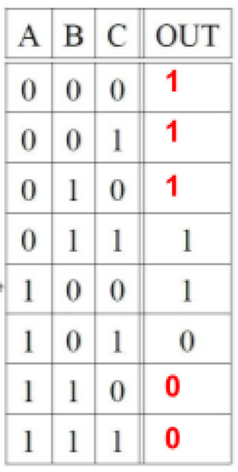
to the current gates, and one N-type from C to GROUND, in series to the current gates.

Replace the transistors in your diagrams from part (*a*) with either a wire or no wire to reflect the circuit's operation when the following inputs are applied:
A = 1, B = 0, C = 0

Output of 3-input OR will be 1, Output of 3-input AND will be 0.

The transistor circuit shown below produces the accompanying truth table. The inputs to some of the gates of the transistors are not specified. Also, the outputs for some of the input combinations of the truth table are not specified. Complete both specifications. i.e., all transistors will have their gates properly labeled with either A, B, or C, and all rows of the truth table will have a 0 or 1 specified as the output.





A	B	C	OUT
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Figure 1

7. Shown below are several logical identities with one item missing in each. X represents the case where it can be replaced by either a 0 or a 1 and the identity will still hold. Your job: Fill in the blanks with either a 0, 1, or X.
For example, in part a, the missing item is X. That is 0 OR 0 = 0 and 0 OR 1 = 1

    a) 0 OR X = _X__

    b) 1 OR X = _1__

c) 0 AND X = _**0**__

d) 1 AND X = _**X**__

e) _**0**_ XOR X = X

8. (Adapted from problem 3.25 in the textbook)
Logic circuit 1 in Figure 3.39 (page 102 of the book) has inputs A, B, C. Logic circuit 2 in Figure 3.40 (page 102 of the book) has inputs A and B. Both logic circuits have an output D. There is a fundamental difference between the behavioral characteristics of these two circuits. What is it? *Hint*: What happens when the voltage at input A goes from 0 to 1 in both circuits?

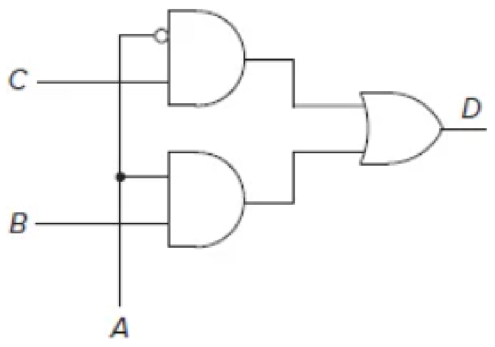


Figure 3.39 **Logic circuit 1 for Exercise 3.25.**

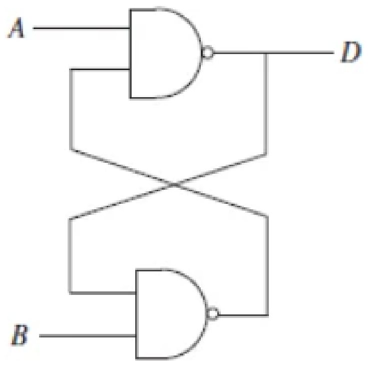
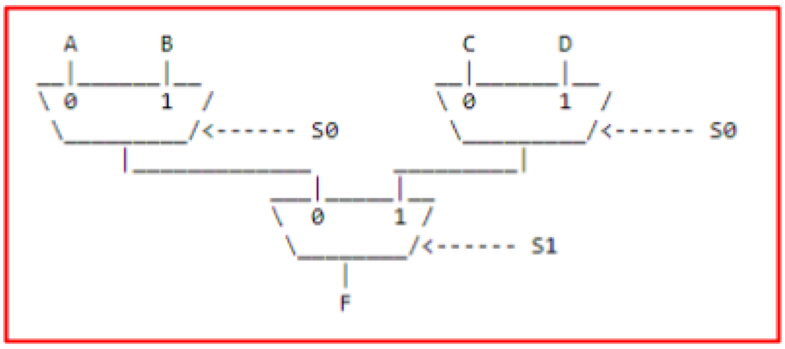


Figure 3.40 **Logic circuit 2 for Exercise 3.25.**

Figure 3.39 is a 2-input mux, which combinational logic i.e., D is the output of the circuit.
Figure 3.40 is a storage element, which stores the data value previously stored in the latch.

9. (Adapted from 3.28 in the textbook)

a) Implement a 4-to-1 mux using only 2-to-1 muxes making sure to properly connect all of the terminals. Remember that you will have 4 inputs (A, B, C, and D), 2 control signals (S1 and S0), and 1 output (OUT). After implementing the 4-1 mux, fill in the truth table below.



You require 3 muxes. First, the inputs are A and B and the select line is S0. Second, inputs are C and D and the select line is also S0. Third, is a mux where both its inputs are the outputs of the first two muxes and the select line is S1.

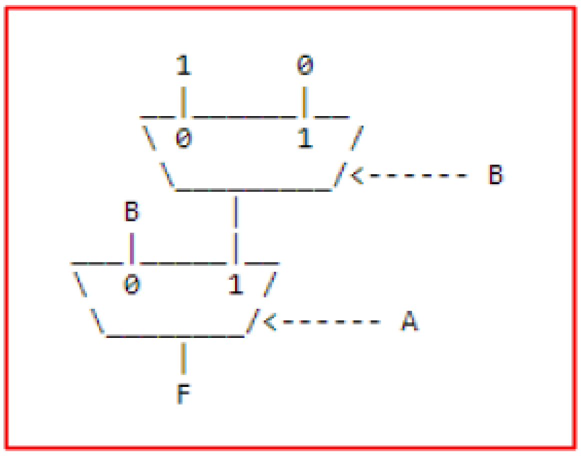
Every time S1 and S0 are both 0, OUT is the same as A.

Every time S1 and S0 are 0 and 1 respectively, OUT is the same as B.

Every time S1 and S0 are 1 and 0 respectively, OUT is the same as C.

Every time S1 and S0 are 1 and 1 respectively, OUT is the same as D.

b) Implement F = A xor B using ONLY two 2-to-1 muxes. You are not allowed to use a not gate (A' and B' are not available).



S1	S0	A	B	C	D	OUT
0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	1	0	0
0	0	0	0	1	1	0
0	0	0	1	0	0	0
0	0	0	1	0	1	0
0	0	0	1	1	0	0
0	0	0	1	1	1	0
0	0	1	0	0	0	1
0	0	1	0	0	1	1
0	0	1	0	1	0	1
0	0	1	0	1	1	1
0	0	1	1	0	0	1
0	0	1	1	0	1	1
0	0	1	1	1	0	1
0	0	1	1	1	1	1
0	1	0	0	0	0	0
0	1	0	0	0	1	0
0	1	0	0	1	0	0
0	1	0	0	1	1	0
0	1	0	1	0	0	1
0	1	0	1	0	1	1
0	1	0	1	1	0	1
0	1	0	1	1	1	1
0	1	1	0	0	0	0
0	1	1	0	0	1	0
0	1	1	0	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
0	1	1	1	0	1	1
0	1	1	1	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	0
1	0	0	0	0	1	0
1	0	0	0	1	0	1
1	0	0	0	1	1	1
1	0	0	1	0	0	0
1	0	0	1	0	1	0
1	0	0	1	1	0	1
1	0	0	1	1	1	1
1	0	1	0	0	0	0
1	0	1	0	0	1	0
1	0	1	0	1	0	1
1	0	1	0	1	1	1
1	0	1	1	0	0	0
1	0	1	1	0	1	0
1	0	1	1	1	0	1
1	0	1	1	1	1	1
1	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	0
1	1	0	0	1	1	1
1	1	0	1	0	0	0
1	1	0	1	0	1	1
1	1	0	1	1	0	0
1	1	0	1	1	1	1
1	1	1	0	0	0	0
1	1	1	0	0	1	1
1	1	1	0	1	0	0
1	1	1	0	1	1	1
1	1	1	1	0	0	0
1	1	1	1	0	1	1
1	1	1	1	1	0	0
1	1	1	1	1	1	1

10. (Adapted from 3.31 in the textbook)
Say the speed of a logic structure depends on the largest number of logic gates through which any of the inputs must propagate to reach an output. Assume that a NOT, an AND, and an OR gate all count as one gate delay. For example, the propagation delay for a two-input decoder shown in Figure 3.11 is 2 because some inputs propagate through two gates.

   a) What is the propagation delay for the two-input mux shown in Figure 3.12 (page 68)?
   3
   b) What is the propagation delay for the 4-bit adder shown in Figure 3.16 (page 71)?
   12
   c) Can you reduce the propagation delay for the circuit shown in Figure 2 by implementing the equation in a different way? If so, how?

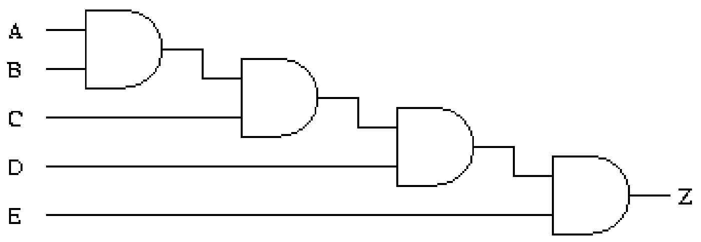


Figure 2

You can construct a tree-like structure.
E = ((A AND B) AND (C AND D)) AND E)

11. (Adapted from problem 3.32 in the textbook)
Recall that the adder was built with individual "slices" that produced a sum bit and carryout bit based on the two operand bits A and B and the carryin bit. We called such an element a full-adder. Suppose we have a 3-to-8 decoder and two six-input OR gates, as shown in Figure 3 below. Can we connect them so that we have a full-adder? If so, please do. (*Hint*: If an input to an OR gate is not needed, we can simply put an input 0 on it and it will have no effect on anything. For example, see the figure below.)

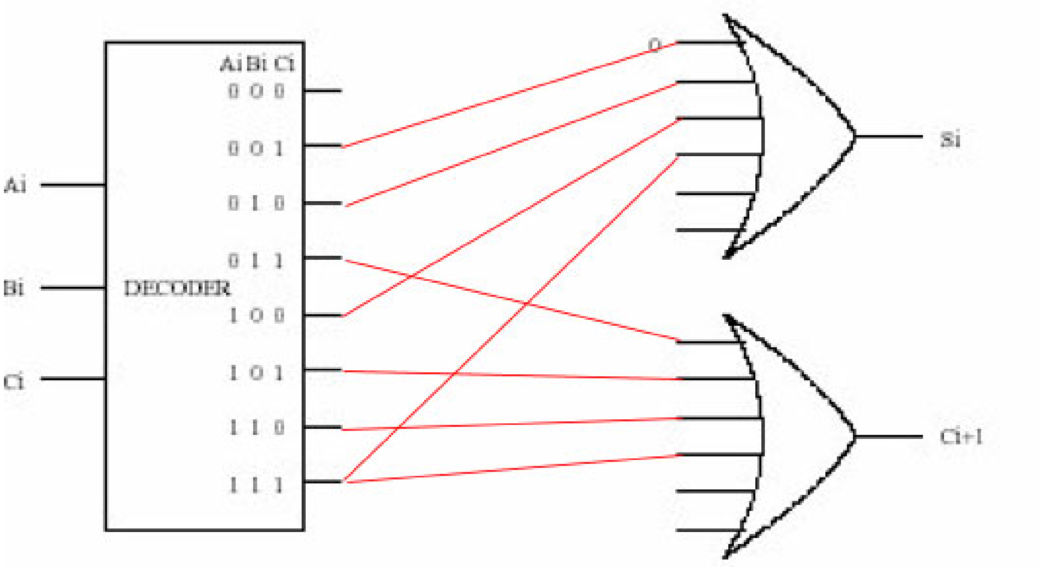


Figure 3

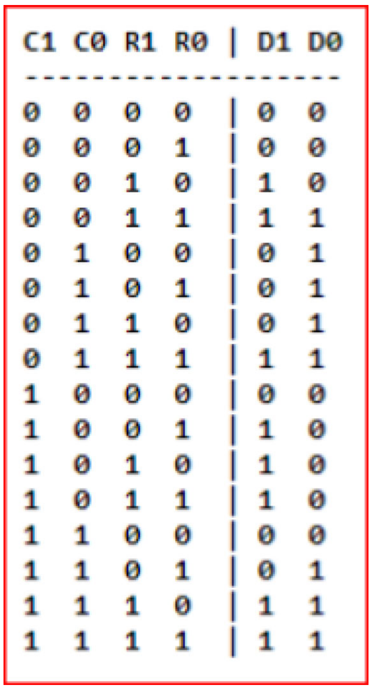
For the OR gate generating the Si, connect the 4 outputs: 001, 010, 100, and 111 from the decoder to the OR gate. Connect the remaining 2 inputs of the OR gate to 0.
For the OR gate generating the Ci+1, connect the 4 outputs: 011, 101, 110, and 111 from the decoder to the OR gate. Connect the remaining 2 inputs of the OR gate to 0.

12. We wish to design a controller for an elevator such that if you push a button for a desired floor, the controller will output the floor number that the elevator should go to. However, to deter lazy people from going up or down one floor, if you push the button for the next floor (up or down), the elevator will stay on its current floor. If you push the button for the same floor that you're currently on, the controller will output the current floor number. There are four floors in the building.

    Your job: construct a complete truth table for the elevator controller. It is not necessary to draw the logic here; the truth table is sufficient.

    Hint: What information does the controller need in order to output the floor to go to?
    Hint: How many input bits will that require.
    Hint: How many output bits will the controller have to supply.

Since there are four floors, you will need 2 bits to represent a floor. Let the logic variable C[1:0] represent the current floor, R[1:0] represent the requested floor, and D[1:0] represent the floor the elevator should go to given a current floor and a requested floor. Shown below is the truth table for this combinational logic circuit.



C1	C0	R1	R0	D1	D0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	1
1	1	1	1	1	1

13. A logic circuit consisting of 6 gated D latches and 1 inverter is shown below:

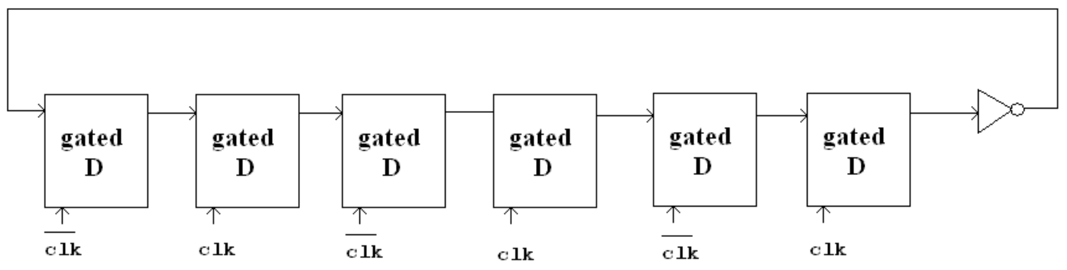


Figure 5

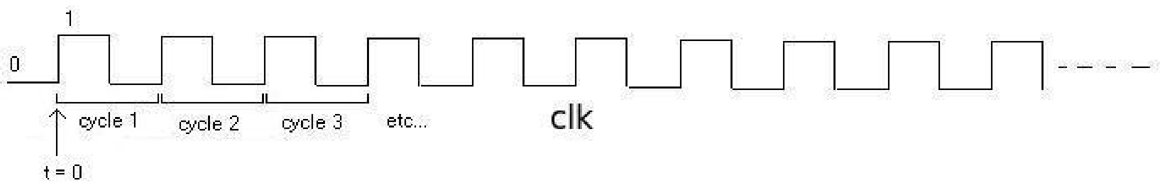
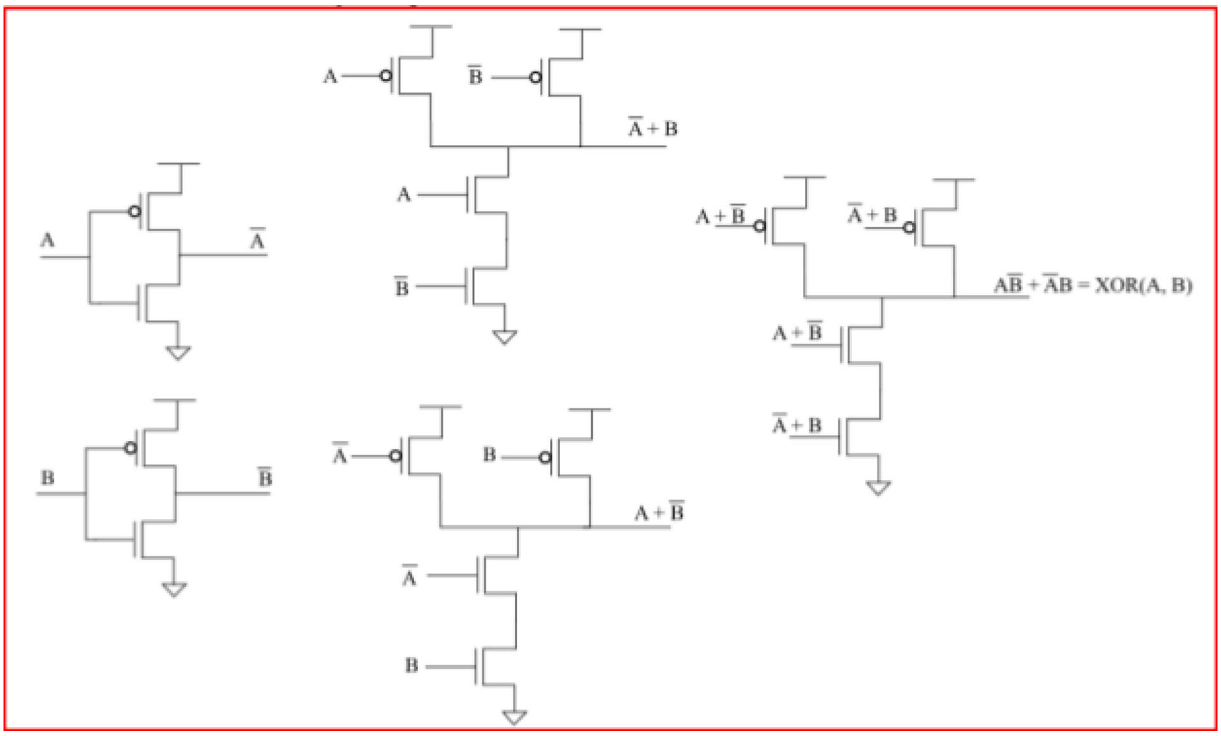


Figure 6

Let the state of the circuit be defined by the state of the 6 D latches. Assume initially the state is 000000 and clk starts at the point labeled t0.
Question: What is the state after 50 cycles? How many cycles does it take for a specific state to show up again?
Every 6 clock cycles a pattern repeats. Because 50 = 6*8+2, after 50 cycles the state will be the same as after 2 cycles. It will be in state 111000 after 50 cycles

14. Draw the transistor level circuit of a 2 input XOR gate.



15. (Adapted from 3.36 in the textbook)
A comparator circuit has two 1-bit inputs, A and B, and three 1-bit outputs, G (greater), E (equal), and L (less than). Refer to figures 3.43 and 3.44 on page 106 in the book for this problem.

i. Draw the truth table for a 1-bit comparator.

A	B	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

ii. Implement G, E and L for a 1-bit comparator using AND, OR, and NOT gates.
G = AB' , L = A'B , E = A'B' + AB

iii. Figure 3.44 performs one-bit comparisons of the corresponding bits of two unsigned integer A[3:0] and B[3:0]. Using the 12 one-bit results of these 4 one-bit comparators, construct a logic circuit to output a 1 if unsigned integer A is larger than unsigned integer B (the logic circuit should output 0 otherwise). The inputs to your logic circuit are the outputs of the 4 one-bit comparators and should be labeled G[3], E[3], L[3], G[2], E[2], L[2], ... L[0]. (*Hint*: You may not need to use all 12 inputs.)
Y = G[3] + E[3]G[2] + E[3]E[2]G[1] + E[3]E[2]E[1]G[0]

16. One of Professor Patt's students is always late to meetings, so Professor Patt wants you to design an alarm clock to help his student be on time. Your job is to design a logic circuit whose output Z is equal to 1 when the alarm clock should go off. The circuit will receive four input variables (A, B, C, D) that answer four different yes/no question (1=yes, 0=no):
A <= Is it going to be sunny today?
B <= Is it the weekend?
C <= Is it 7:00am?
D <= Is it 9:00am?

Professor Patt wants the alarm clock to go off if it's sunny and it's either 7:00am or 9:00am. The alarm clock should go off if it's the weekend and it's 9:00am. The alarm clock should also go off if it's not the weekend and it's 7:00am. Write the truth table and draw a gate-level diagram that performs this logic.

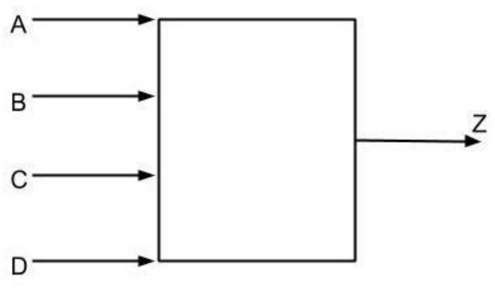
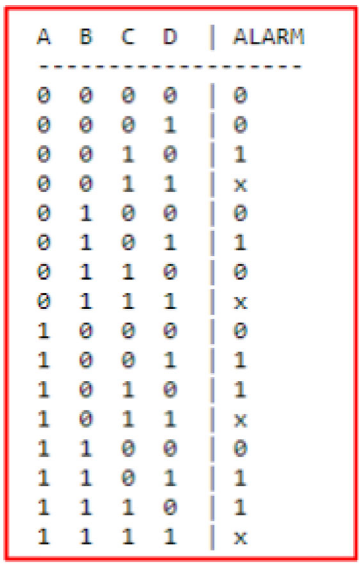
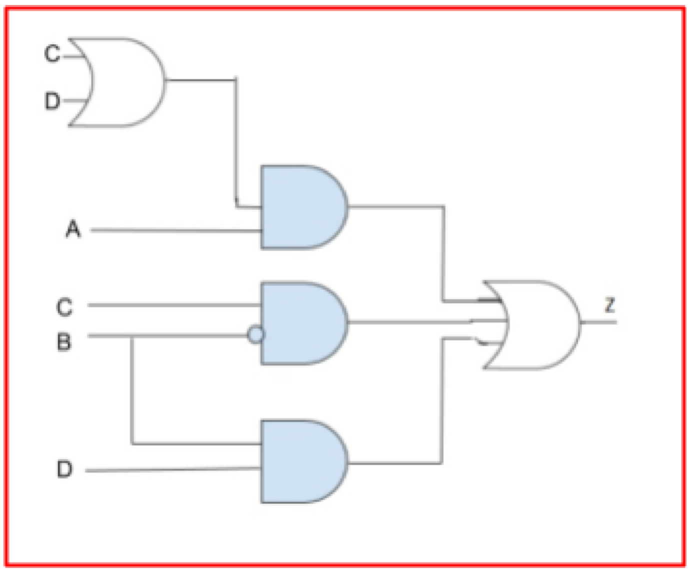


Figure 7



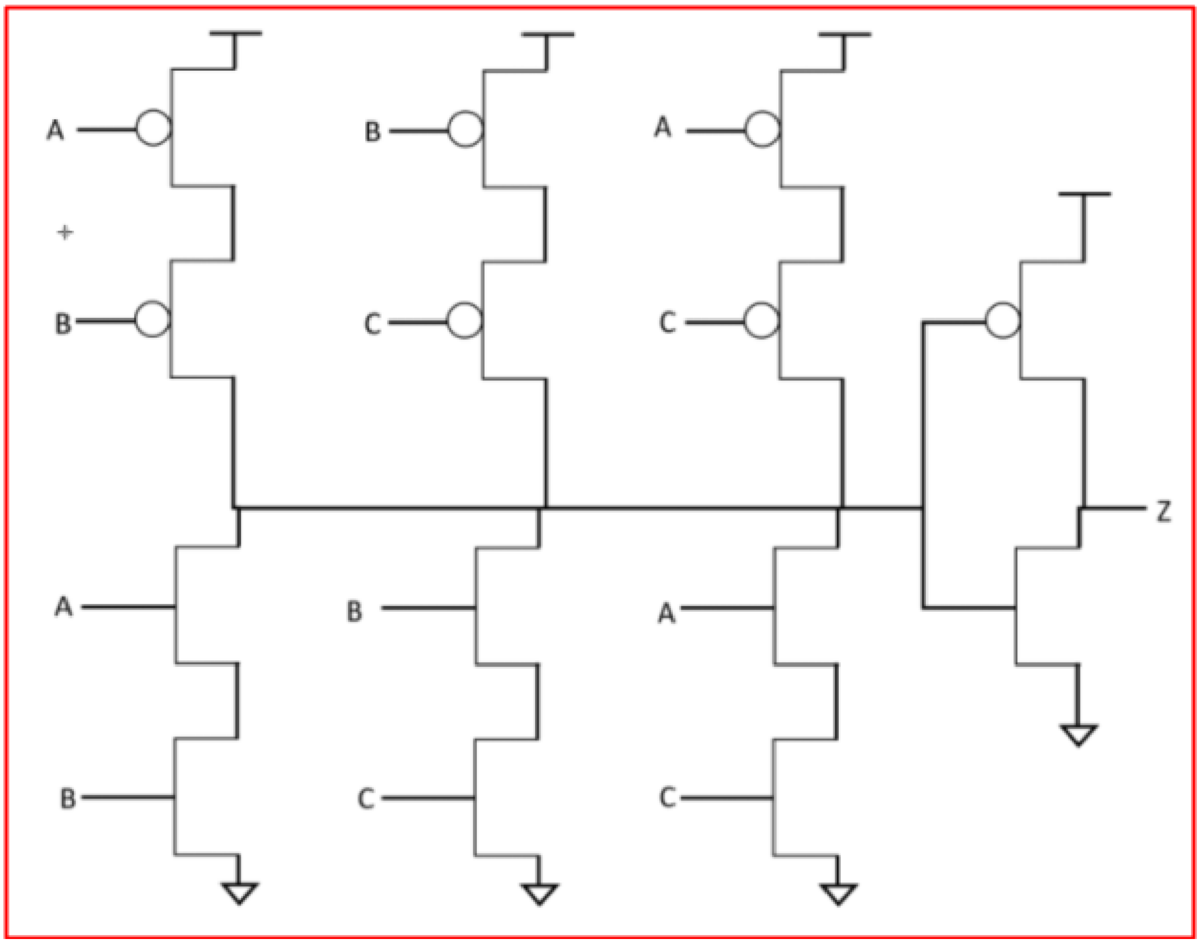
A	B	C	D	ALARM
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	x
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	x
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	x
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	x



17. In class, we showed that NOT((NOT A) AND (NOT B)) == A OR B using their truth tables. This equivalence is known as DeMorgan's law. Similarly, show that NOT((NOT A) OR (NOT B)) == A AND B.

A B	NOT A	NOT B	(NOT A) OR (NOT B)	NOT ((NOT A) OR (NOT B))	A AND B
0 0	1	1	1	0	0
0 1	1	0	1	0	0
1 0	0	1	1	0	0
1 1	0	0	0	1	1

18. Draw the transistor level circuit of a 3 input majority gate.



19. Prove that NAND is logically complete.

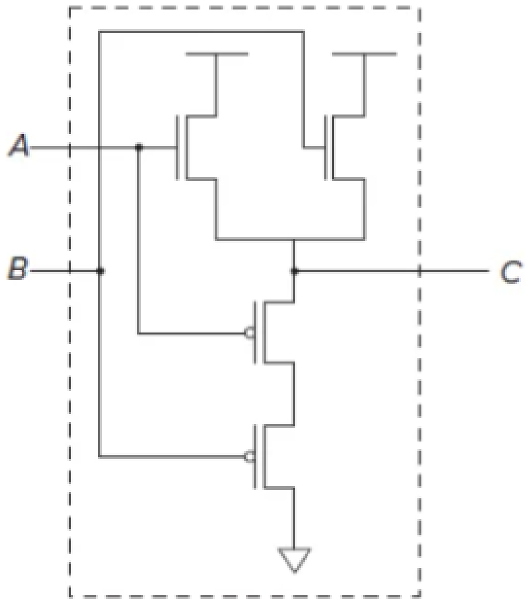
Hint: Since we know that NOT and AND are logically complete, you just need to implement NOT and AND using NAND.

NOT: A NAND A => NOT A

AND: (A NAND B) NAND (A NAND B) = NOT (A NAND B) => A AND B

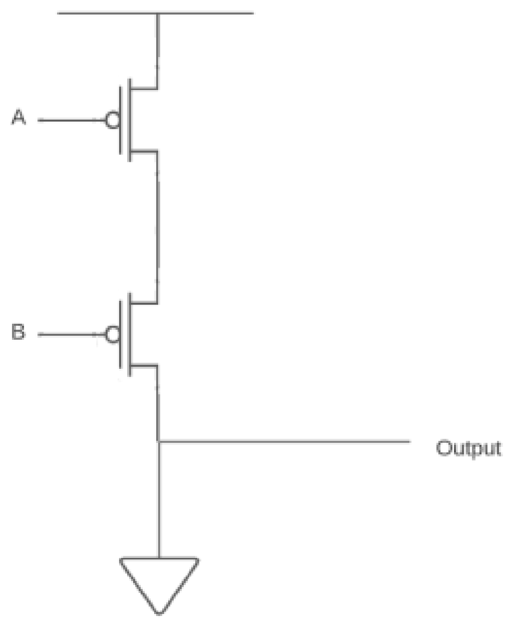
20. A clever student believes they have designed the OR gate and NOR gate using fewer transistors than Professor Patt (see Figure 3.5 and Figure 3.6 in the textbook). The transistor diagrams are shown below. For each diagram, state whether or not the diagram is correct and justify your answer. Answer in 15 words or fewer.

a) The clever student's OR gate:



P type must go on top, N type must go on bottom.

b) The clever student's NOR gate:



When A=0 and B=0, there is a short circuit from our voltage source to ground

21. (Reworded on 9/22 to improve clarity)
Suppose we have two 4-bit binary strings: A and B. Assume that A is a 4-bit one-hot bit string (only a single bit in A is set and the rest are zero). For example, A could be 0100 because only the 2nd bit is set. (As a shorthand, we would say that A[2] = 1, while A[3] = A[1] = A[0] = 0.) However, A could not be 0110 since we assume only a single bit of A is set.

a. Given A and B as the inputs, let *p* be the position of the (only) bit in A that is set, i.e., A[*p*] = 1. We want to design a circuit that uses *p* and produces the single-bit output C = B[*p*]. Here are a few examples:

Example 1: A = 1000, B = 1010 → p=3 (since A[3] = 1) → C = B[3] = 1

Example 2: A = 0100, B = 1010 → p=2 → C = B[2] = 0

Example 3: A = 1100, B = 1111 → A is invalid → C can be anything.

Your job: Draw a gate-level logic structure that implements this circuit.

[Note: when an output can be any value, we call that a don’t-care term. In the truth table, we denote a don’t-care term with a lowercase ‘x’.]

C = A[0]B[0] + A[1]B[1] + A[2]B[2] + A[3]B[3]

b. We want to add a new feature to the logic structure designed in part (a). Specifically, we want to add a second output bit -- call it E -- that is 1 whenever A is invalid and 0 otherwise.

   Example 1: A = 1000, B = 1000; C = 1, E = 0.

   Example 2: A = 1000, B = 0100; C = 0, E = 0.

   Example 3: A = 1100, B = 1111; C can be anything, E = 1.

   Your job: Draw a gate-level logic structure that implements this modified circuit.

   E = A[0]A[1] + A[0]A[2] + A[0]A[3] + A[1]A[2] + A[1]A[3] + A[2]A[3] + NOT(A[0]) NOT(A[1]) NOT(A[2]) NOT(A[3])