1. What is the smallest positive normalized number that can be represented using the IEEE Floating Point standard?

   2^(-126)

   0 00000001 000000…000

   What about the smallest positive integer that can NOT be represented using IEEE Floating Point standard?

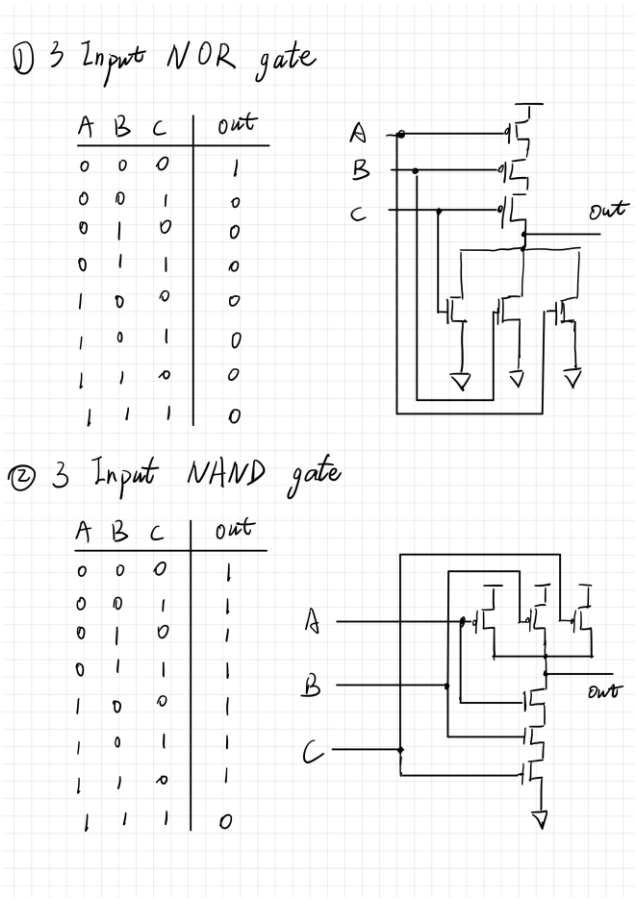
2. What is the largest positive number that can be represented in a 32 bit 2's complement scheme?

   2,147,483,647

   01111…111

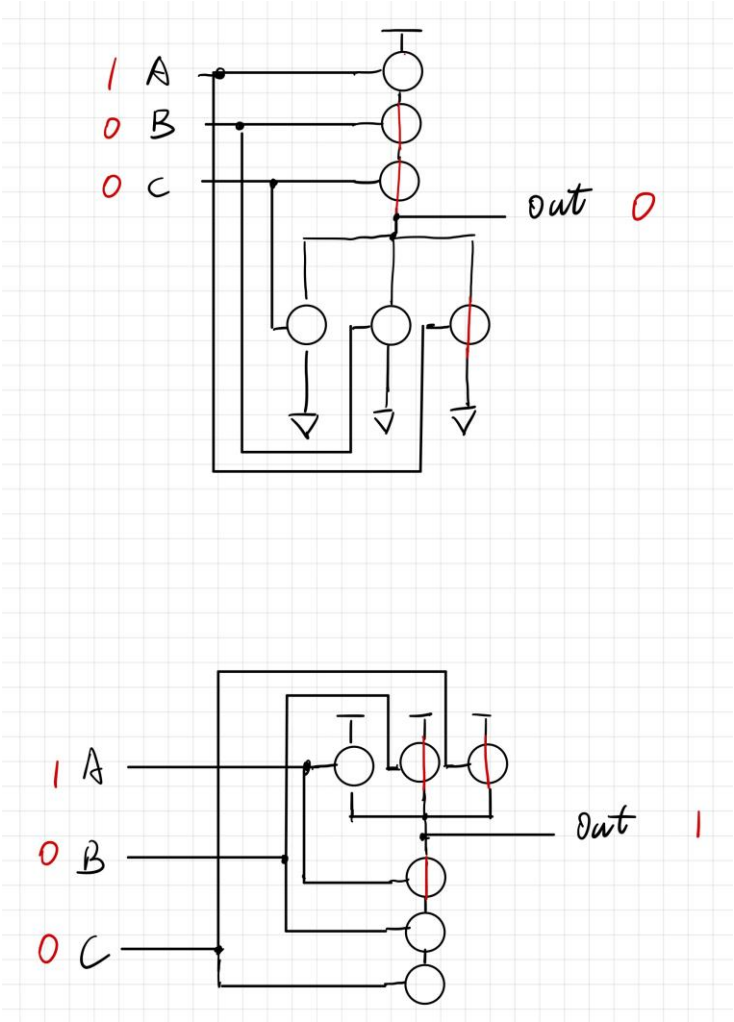
3.

   a. (Adapted from 3.17) Draw a transistor-level diagram for a three-input NAND gate and a three-input NOR gate. Do this by extending the designs from following Figures 3.5a and 3.8a(NAND). (Figures can also be found in the book on pages 63 & 65 respectively).

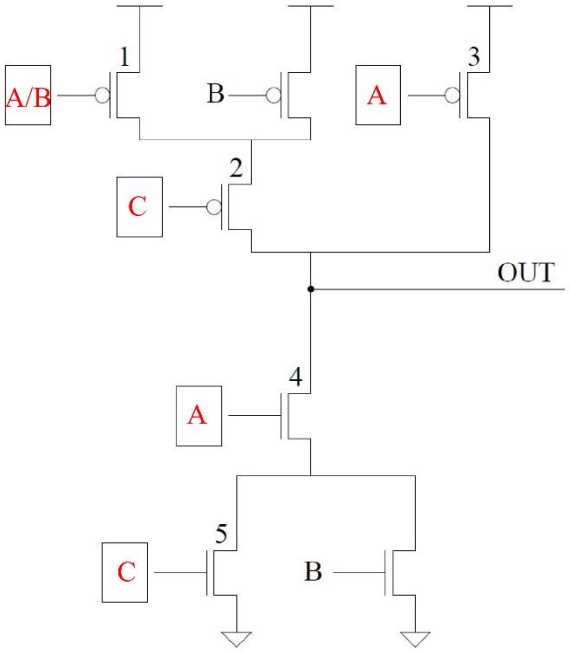


b. Replace the transistors in your diagrams from part (*a*) with either a wire or no wire to reflect the circuit's operation when the following inputs are applied:

A = 1, B = 0, C = 0



c. 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

4. 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. __ XOR X = X
      0
   f. X XOR X = ___
      0

5. (3.25)
   Logic circuit 1 in Figure 3.36 (page 87 of the book) has inputs A, B, C. Logic circuit 2 in Figure 3.37 (page 87 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 3.36 is a 2-input mux, which combinational logic i.e., D is the output of the circuit.
   Figure 3.37 is a storage element, which stores the data value previously stored in the latch.

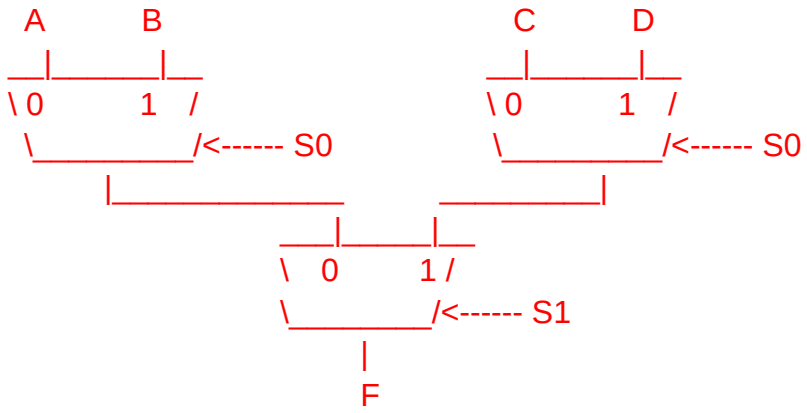
6. (Adapted from 3.28)

(1) Fill in the truth table of 4-to-1 mux:

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

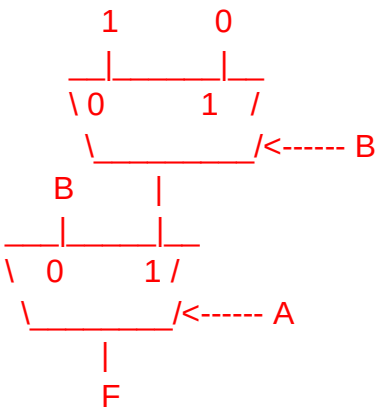
S1	S0	A	B	C	D	OUT
1	0	0	0	0	0	0
1	0	0	0	0	1	0
1	0	0	0	1	0	1
1	0	0	0	1	1	1
1	0	0	1	0	0	0
1	0	0	1	0	1	0
1	0	0	1	1	0	1
1	0	0	1	1	1	1
1	0	1	0	0	0	0
1	0	1	0	0	1	0
1	0	1	0	1	0	1
1	0	1	0	1	1	1
1	0	1	1	0	0	0
1	0	1	1	0	1	0
1	0	1	1	1	0	1
1	0	1	1	1	1	1
1	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	0
1	1	0	0	1	1	1
1	1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

(2) Implement the 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).



You require 3 muxes. First, the input 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 select line is S1.

(3) 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).



7. (Adapted from 3.31)
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 3 by implementing the equation in a different way? If so, how?

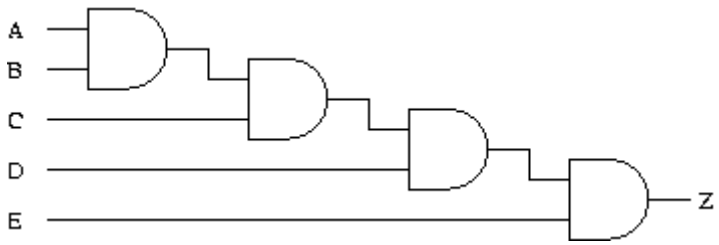


Figure 3

You can construct a tree-like structure.

E = ((A AND B) AND (C AND D)) AND E)

8. (3.32)
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.)

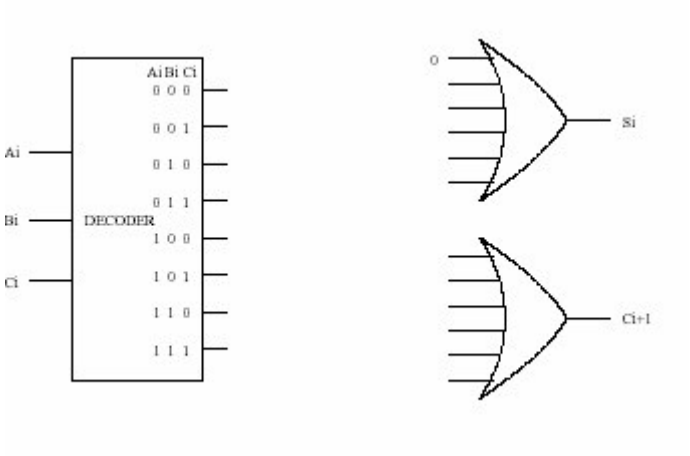
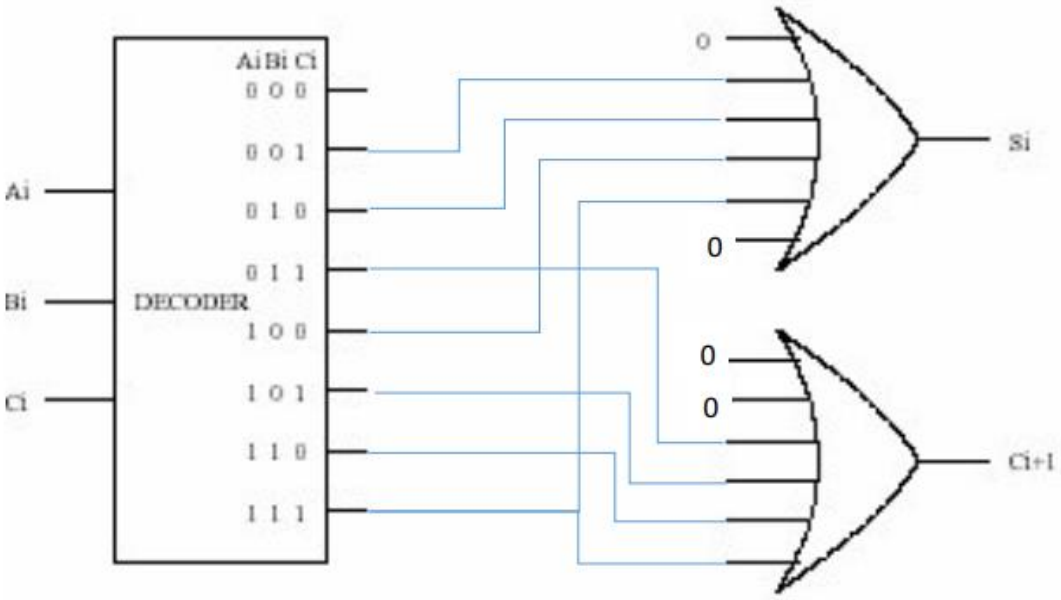


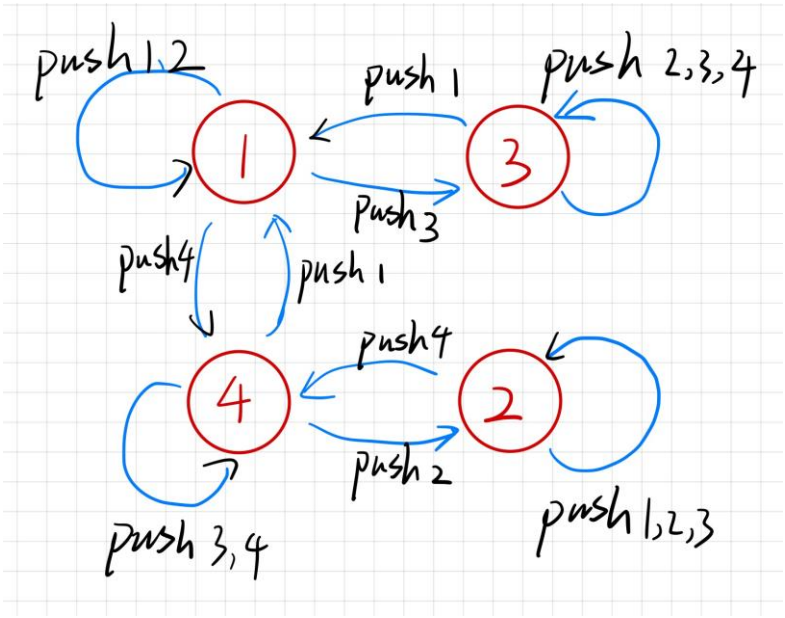
Figure 3



9. 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:

   a. Draw the state diagram of the elevator scheduling.



b. Construct a complete truth table for the elevator controller. It is not necessary to draw the logic here; the truth table is sufficient.

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

10.

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

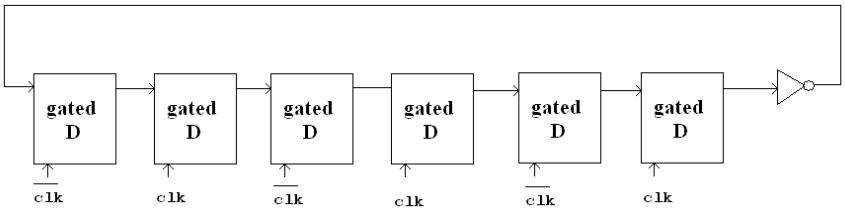


Figure 5

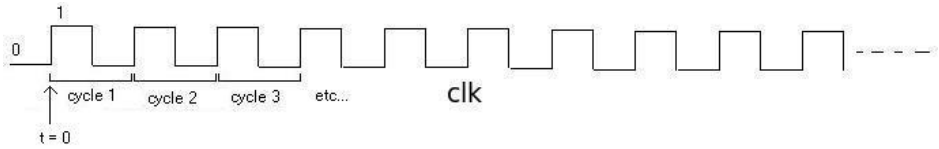


Figure 6

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 cyles. How many cycles does it take for a specific state to show up again?

Every 6 clock cycles a pattern repeats. A and B represent the first half and the second half of each clock cycle respectively.

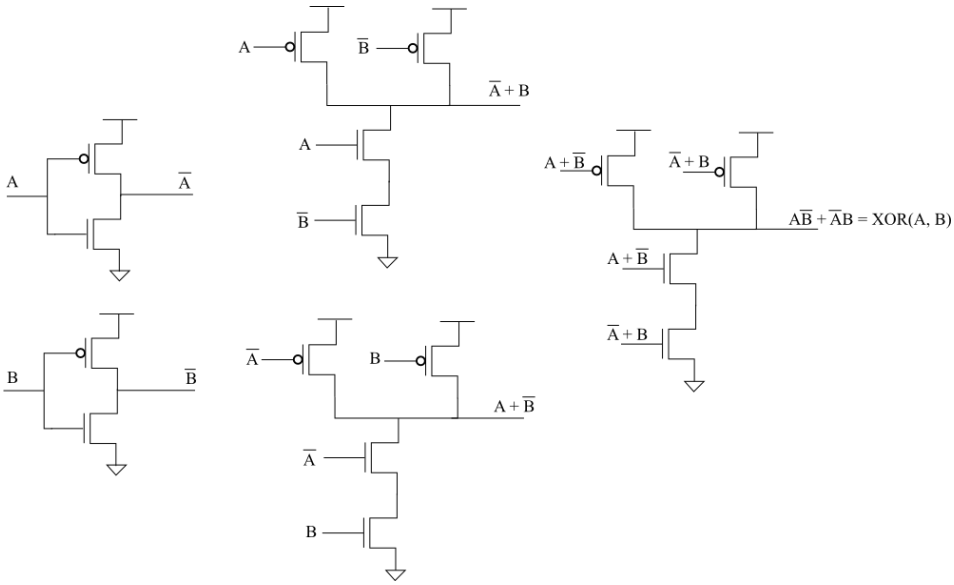
Cycle1 A: 000000
Cycle1 B: 100000
Cycle2 A: 110000
Cycle2 B: 111000
Cycle3 A: 111100
Cycle3 B: 111110
Cycle4 A: 111111
Cycle4 B: 011111
Cycle5 A: 001111
Cycle5 B: 000111
Cycle6 A: 000011
Cycle6 B: 000001

Cycle7 A: 000000

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

11.

Draw the transistor level circuit of a 2 input XOR gate



12. (Adapted from 3.36)

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..

a. 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

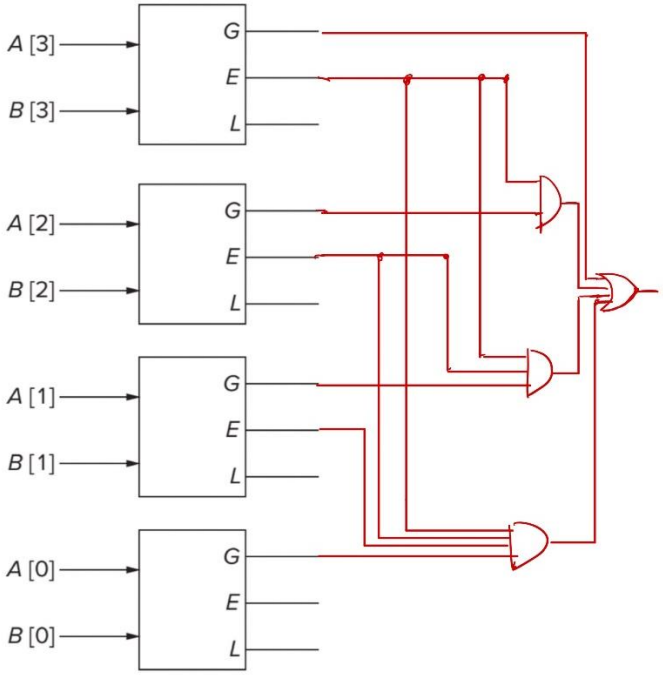
b. Implement G, E and L for a 1-bit comparator using AND, OR, and NOT gates.

G = AB' , L = A'B , E = A'B' + AB

c. 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

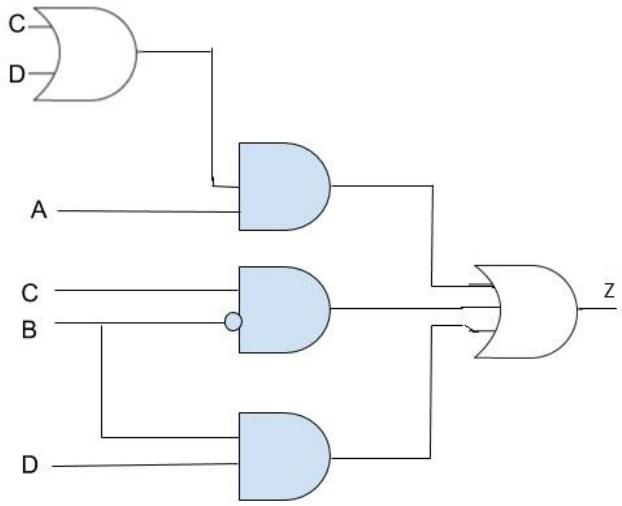
circuit to output a 1 if unsigned integer A is larger than unsigned integer B (the logic circuit should output 0 otherwise). The inputs to your logic circuit are the outputs of the 4 one-bit comparators and should be labeled G[3], E[3], L[3], G[2], E[2], L[2], ... L[0]. (Hint: You may not need to use all 12 inputs.)

Y = G[3] + E[3]G[2] + E[3]E[2]G[1] + E[3]E[2]E[1]G[0]



13. One of Zhang San's students is always late to meetings, so Professor Zhang San 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?

    Zhang San 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.



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

14. Prove that NAND is logically complete.

NOT: A NAND A => NOT A

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

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