Homework 02

和泳毅 PB19010450

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

The largest exponent is `11111111` --> `254` --> `254-127=127`

So, the largest positive normalized number is:

$N = (-1)^0 \times 1.11111111111111111111111 \times 2^{127} = 3.4028235 \times 10^{38}$

---

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

Because of the signed magnitude, the highest bit in a 32 bit 2's complement is 2^{31}.

So, the largest positive number is $2^{31} - 1 = 2147483647$.

---

3. a. (3.17) Draw a transistor-level diagram for a three-input AND gate and a three-input OR gate. Do this by extending the designs from following Figures 3.6a and 3.8a. (Figures can also be found in the book on pages 64 & 65 respectively).

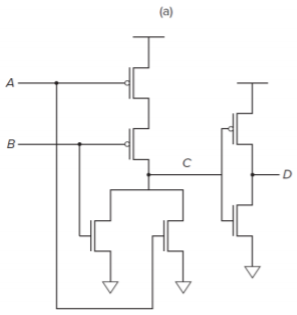


Figure 3.6a

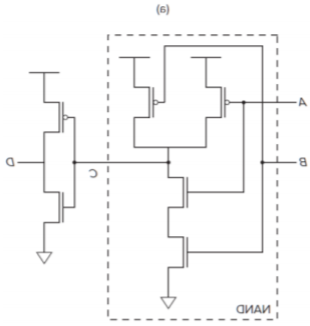
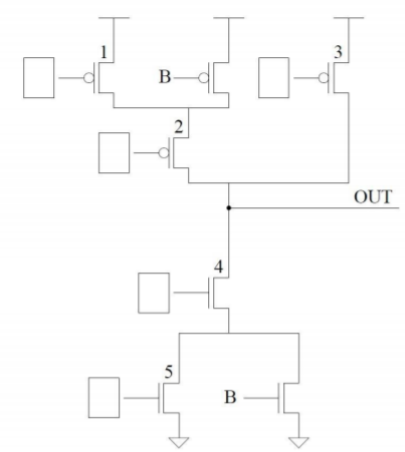


Figure 3.8a

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

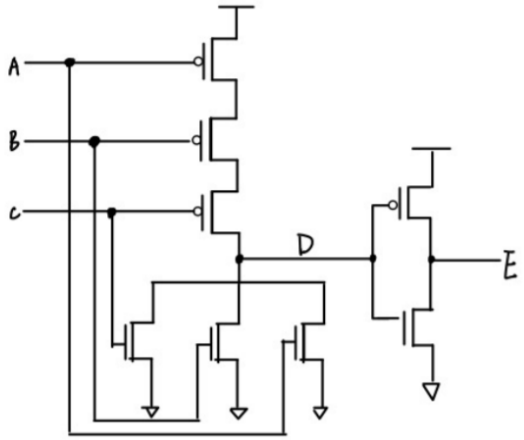
c. The transistor circuit shown below (Figure 1) produces the accompanying truth table. The inputs to some of the gates of the transistors are not specified. Also, the outputs for some of the input combinations of the truth table are not specified. Complete both specifications. i.e., all transistors 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.



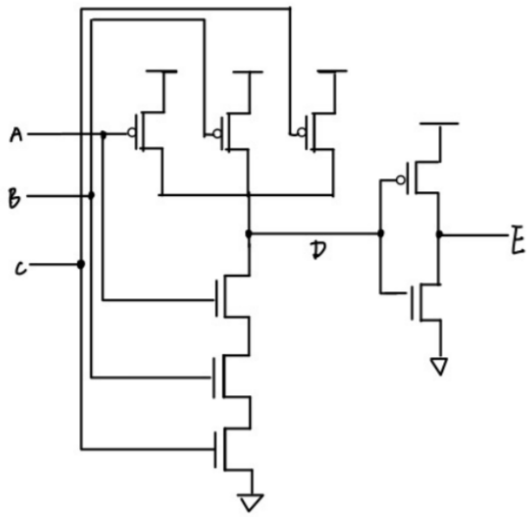
A	B	C	OUT
0	0	0	
0	0	1	
0	1	0	
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	
1	1	1	

Figure 1

a.

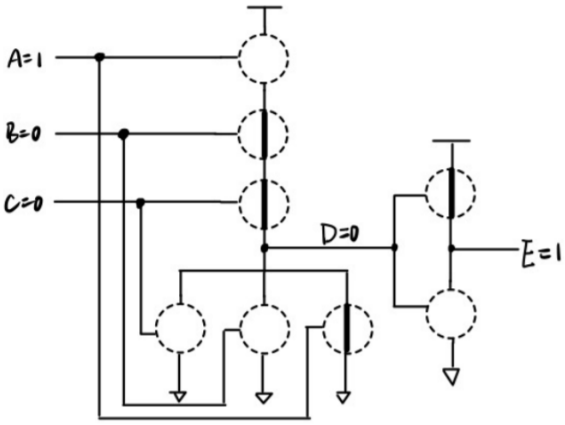


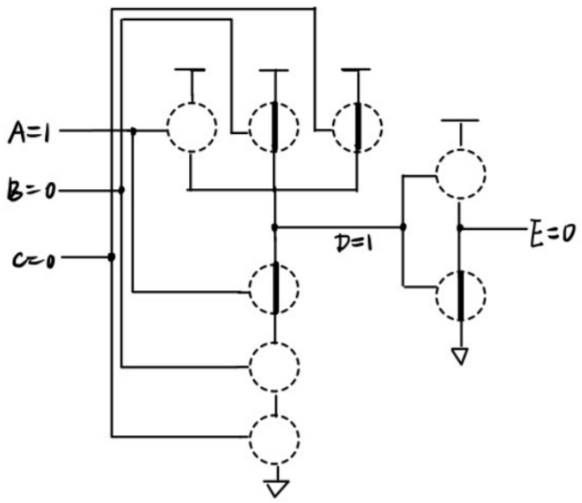
Three-input OR gate



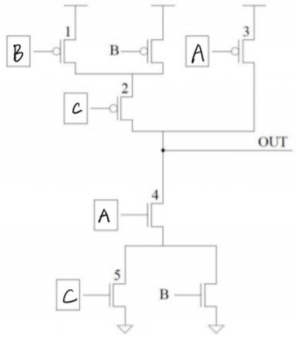
Three-input AND gate

b.





c.



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

---

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) __0__ XOR X = X

---

5. (3.25)

   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?

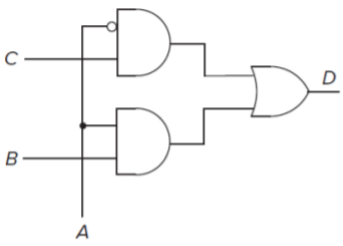


Figure 3.39 Logic circuit 1 for Exercise 3.25.

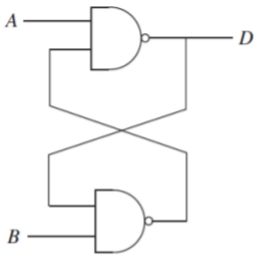


Figure 3.40 Logic circuit 2 for Exercise 3.25.

Logic circuit 1 is a simple combinational circuit. The output value only depends on the input values as they currently exist. Logic circuit 2 is an R-S Latch. It is a logic circuit which can store information. That is, if A, B are both 1, the value of D depends on which of the two (A or B) was 0 most recently.

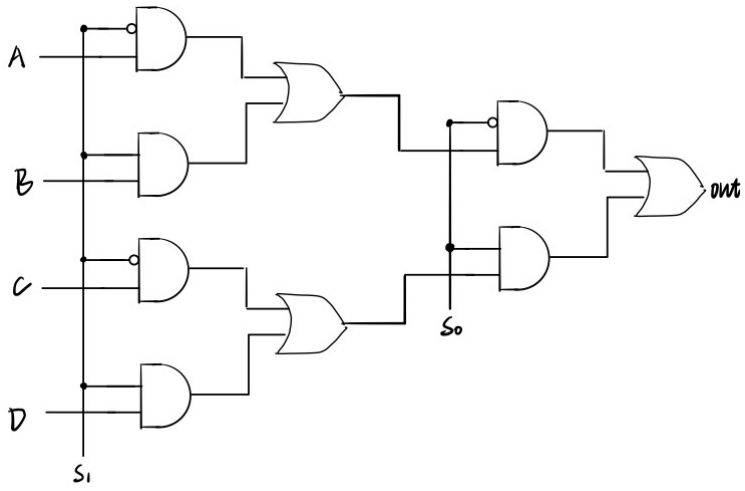
---

6. (Adapted from 3.28)

(1) 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.

(2) 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).

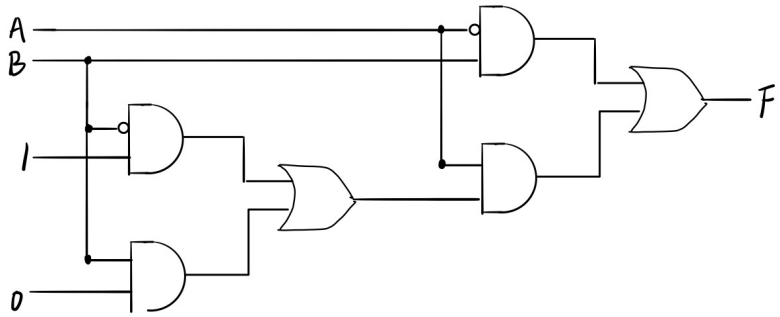
(1)



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	1
0	1	0	0	1	1	1
0	1	0	1	0	0	0
0	1	0	1	0	1	0
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	1
0	1	1	0	1	1	1
0	1	1	1	0	0	0
0	1	1	1	0	1	0
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	0
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	0	0	1	0	1	1
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	0
1	0	1	0	1	1	0
1	0	1	1	0	0	1
1	0	1	1	0	1	1
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)



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.

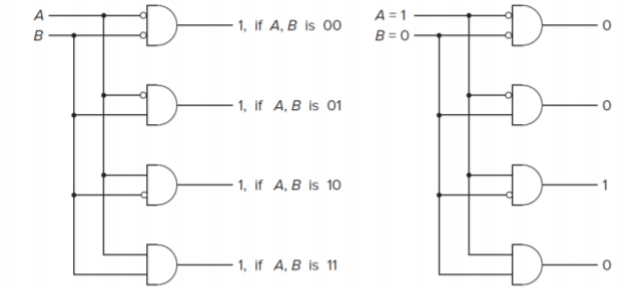


Figure 3.11 A two-input decoder.

a) What is the propagation delay for the two-input mux shown in Figure 3.12 (page 68)?

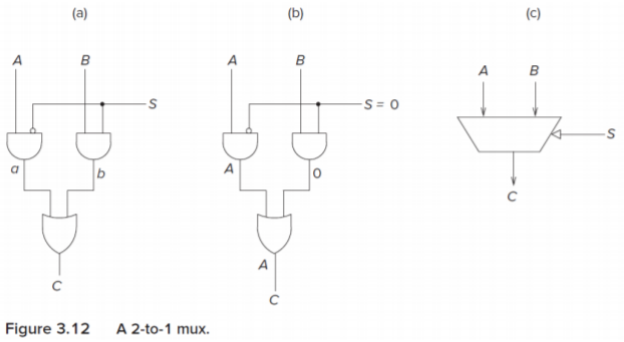


Figure 3.12 A 2-to-1 mux.

b) What is the propagation delay for the 4-bit adder shown in Figure 3.16 (page 71)?

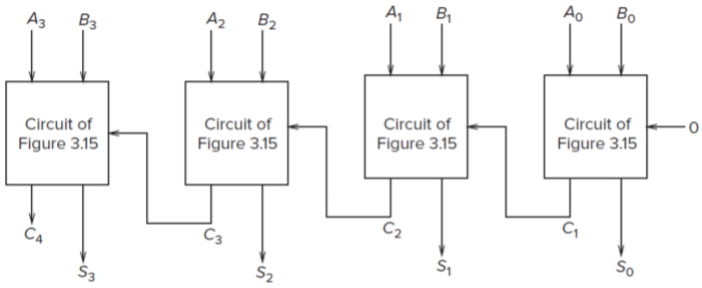


Figure 3.16 A circuit for adding two 4-bit binary numbers.

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?

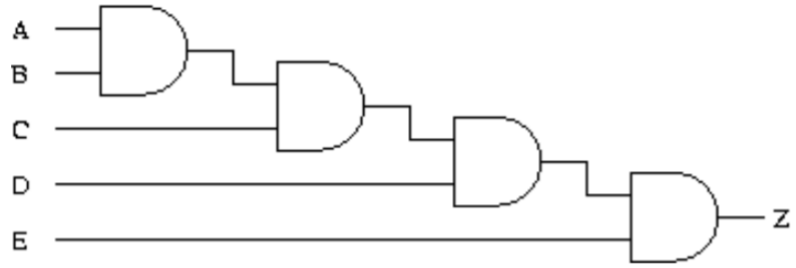


Figure 2

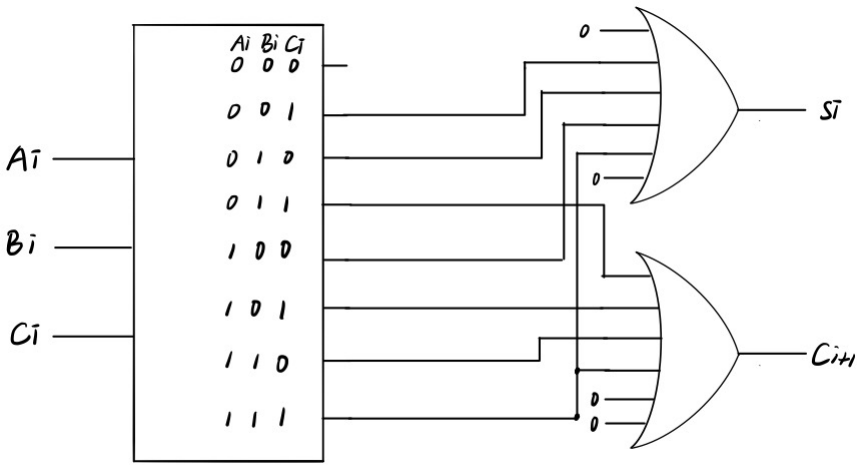
(a) The propagation delay is 3.

(b) `3+3+3+3=12` The propagation delay is 12.

(c) Yes, we can turn the circuit shown in Figure 2 into a five-input AND gate.

---

8. 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.)



---

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: 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.

L_i is the switch for a desired floor, S_i is the input from memory elements for a current floor, S_i' is the output to memory elements.

L1	L2	L3	L4	S1	S2	S3	S4	S1'	S2'	S3'	S4'	OUT
0	0	0	0	0	0	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0	1
0	1	0	0	0	1	0	0	0	1	0	0	1
0	0	1	0	0	0	1	0	0	0	1	0	1
0	0	0	1	0	0	0	1	0	0	0	1	1
1	0	0	0	0	1	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0	1	0	0	0
0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	1	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	1	0	0	1	0	0
0	0	0	1	0	0	1	0	0	0	0	1	0
1	0	0	0	0	0	1	0	1	0	0	0	0
1	0	0	0	0	0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1	0	1	0	0	0
0	0	1	0	1	0	0	0	0	0	1	0	0
0	0	0	1	1	0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0	0	0	0	1	0

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

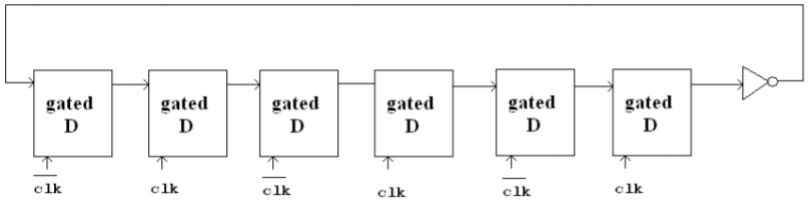


Figure 5

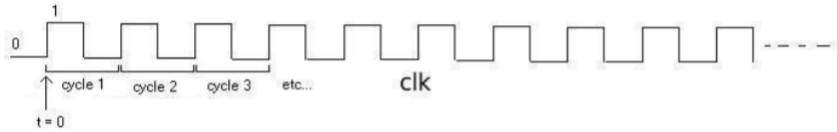


Figure 6

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

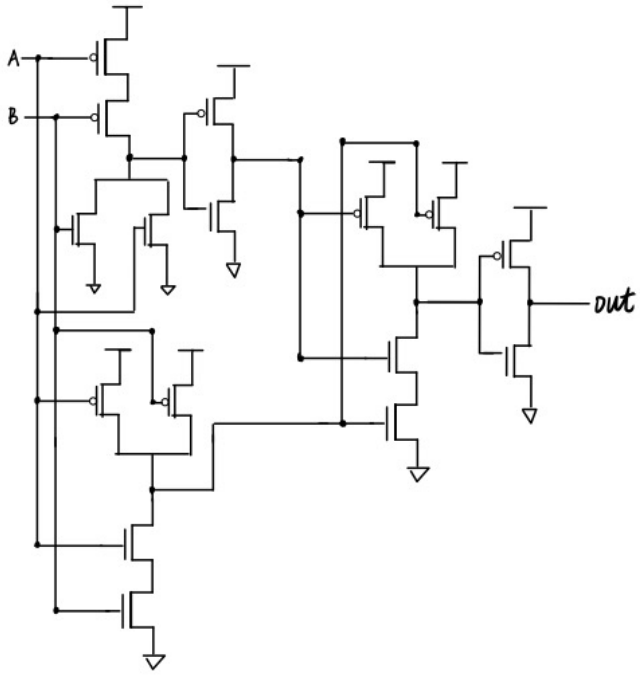
```
cycle1: 000000 100000
cycle2: 110000 111000
cycle3: 111100 111110
cycle4: 111111 011111
cycle5: 001111 000111
cycle6: 000011 000001
cycle7: 000000 100000
```

`50 % 6 = 2`

So the state after 50 cycles is 111000; 6 cycles take for a specific state to show up again.

---

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.

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

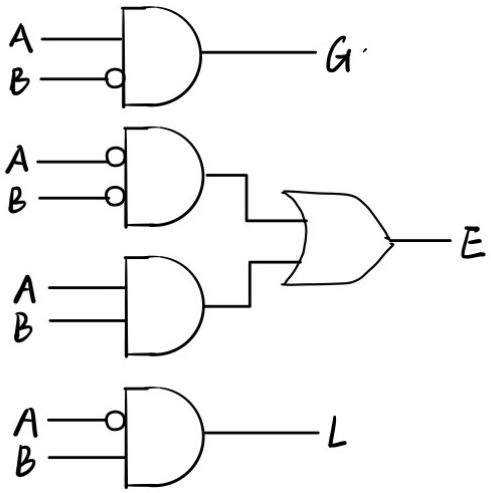
    c. Figure 3.44 performs one-bit comparisons of the corresponding bits of two unsigned integer A[3:0] and B[3:0]. Using the 12 one-bit results of these 4 one-bit comparators, construct a logic circuit to output a 1 if unsigned integer A is larger than unsigned integer B (the logic circuit should output 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.)

a.

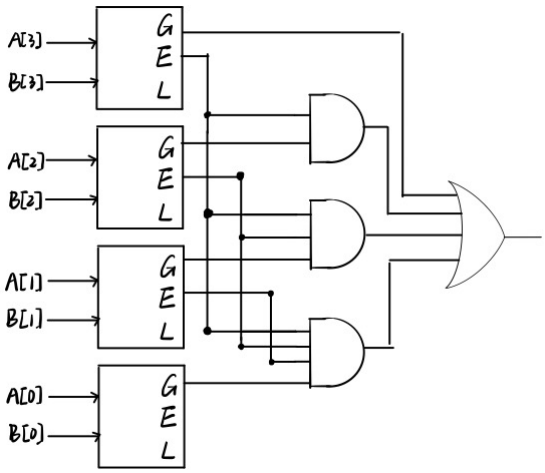
A	B	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

b.



G = AB' ; L = A'B ; E = A'B' + AB.

c.



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 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.

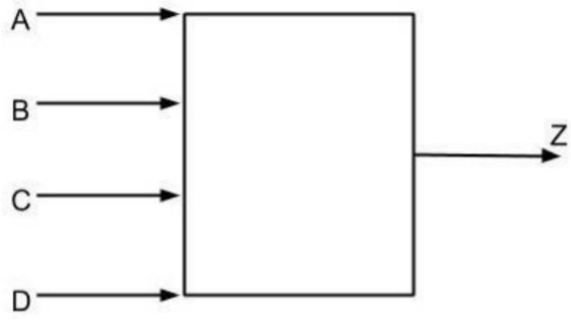
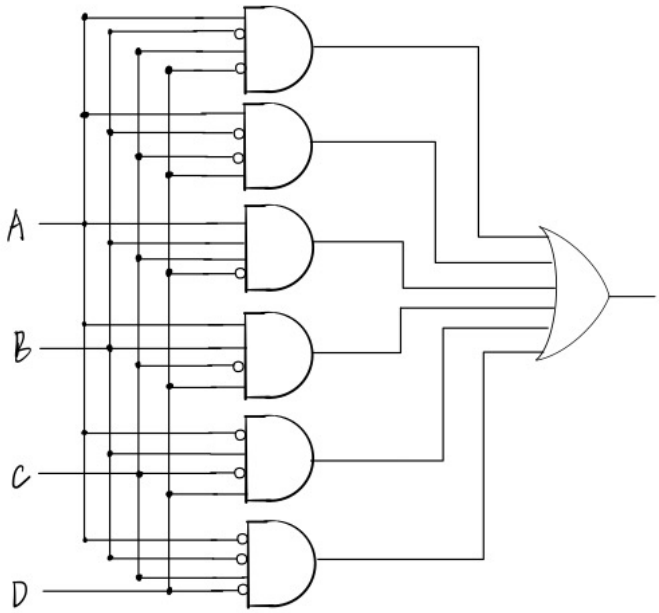
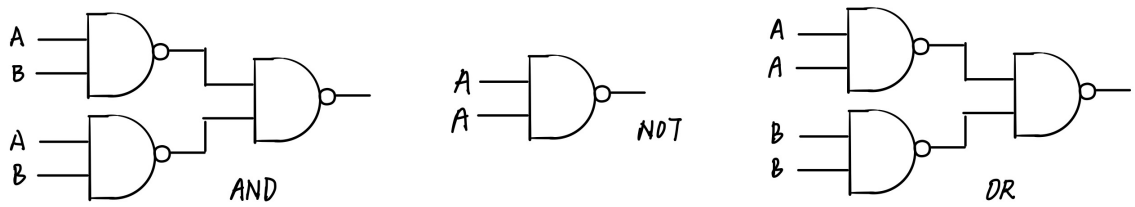


Figure 7

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



14. Prove that NAND is logically complete.



NOT A = NAND A;

A AND B = (A NAND B) NAND (A NAND B) = NOT (A NAND B);

A OR B = (NAND A) NAND (NAND B) = (NOT A) NAND (NOT B) .