HW-Lecture 1 & 2

1. Convert the following unsigned binary numbers to decimal and hexadecimal.

   (a) 1010_2 (b) 11110000_2

2. Convert the following two's complement binary numbers to decimal.

   (a) 110110_2 (b) 01110000_2 (c) 10011111_2

3. What are the largest and smallest 32-bit binary number(show the decimal values) that can be represented with

   (a) unsigned numbers?

   (b) two's complement numbers?

   (c) sign/magnitude numbers?

4. Convert the following decimal numbers to 8-bit two's complement numbers or indicate that the decimal number would overflow the range.

   (a) 42_{10} (b) -63_{10} (c) 124_{10} (d) -128_{10} (e) 133_{10}

5. extend the following 4-bit two's complement numbers to 8-bit two's complement numbers.

   (a) 0101_2 (b) 1010_2

   Repeat your work if the numbers are unsigned

6. Perform the following additions of unsigned binary numbers. Indicate whether the sum overflows an 8-bit result.

   (a) $10011001_2 + 01000100_2$ (b) $11010010_2 + 10110110_2$

   Repeat your work, assuming that the binary numbers are in two's complement form.

7. Convert the following decimal and hexadecimal numbers to 8-bit two's complement binary numbers and add them. Indicate whether the sum overflows an 8-bit result.

   (a) $27_{10} + 31_{10}$ (b) $-4_{10} + 19_{10}$ (c) $-28_{10} + (-111_{10})$ (d) $8F_{16} + AD_{16}$

8. In a ***binary coded decimal*** (***BCD***) system, 4 bits are used to represent a decimal digit from 0 to 9. For example, 37_{10} is written as 00110111_{BCD}.

   (a) Write 371_{10} in BCD.

   (b) Convert 000110000111_{BCD} to decimal.

   (c) Convert 10010101_{BCD} to binary.

   (d) Explain the disadvantages of BCD when compared with binary representations of numbers.

9. Is it possible to assign logic levels so that a device with the transfer characteristics shown in Figure 1.1 would serve as a buffer? If so, what are the input and output low and high

levels (V_{IL}, V_{OL}, V_{IH}, and V_{OH}) and noise margins (NM_L and NM_H)? If not, explain why not.

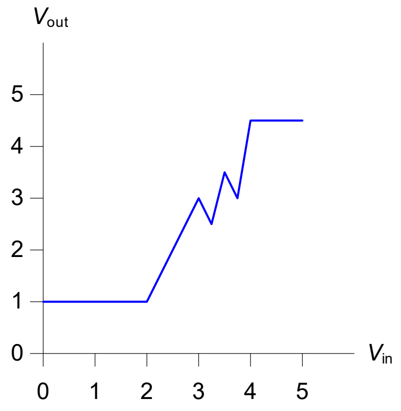


Figure 1.1 Figure 1.1 DC transfer characteristics

10. Ben Bitdiddle has invented a circuit with the transfer characteristics shown in Figure 1.2 that he would like to use as a buffer. Will it work? Why or why not? He would like to advertise that it is compatible with LVCMOS and LVTTL logic. Can Ben's buffer correctly receive inputs from those logic families? Can its output properly drive those logic families? Explain.

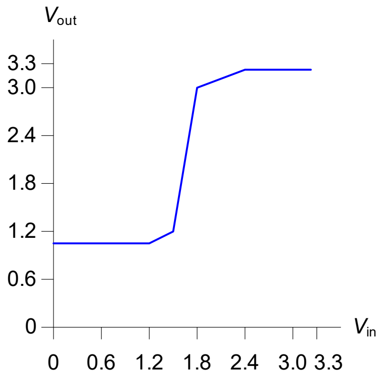


Figure 1.2 Ben's buffer DC transfer characteristics

11. The following are voltage transfer characteristics of single-input, single-output devices to be used in a new logic family:

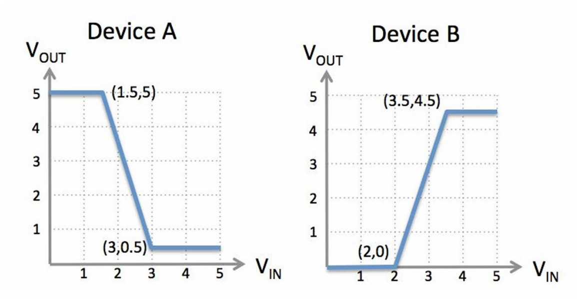


Figure 1.3 DC transfer characteristics of new logic family devices

Your job is to choose a single set of signaling thresholds V_{OL}, V_{IL}, V_{OH}, and V_{IH} to be used with both devices to give the best noise margins you can.

V_{OL} = _____ V_{IL} = _____ V_{IH} = _____ V_{OH} = _____

Low Noise Margin = _____

High Noise Margin = _____

12. Draw the symbol, Boolean equation, and truth table for a three-input OR gate.

13. Sketch a transistor-level circuit for a three-input AND gate, use a minimum number of transistors.

14. A three-input OR-AND-INVERT (OAI) gate shown in Figure 1.4 produces a FALSE output if C is TRUE and A or B is TRUE. Otherwise, it produces a TRUE output. Complete a truth table for the gate.

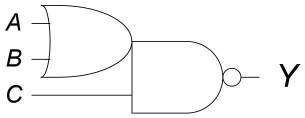


Figure 1.4 OAI

Sketch a transistor-level circuit for this CMOS gates. Use a minimum number of transistors.

15. Write a truth table for the function performed by the gate in Figure 1.5. The truth table should have two inputs, A and B. What is the name of this function?

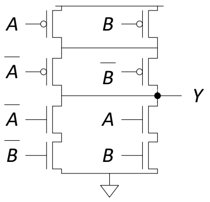


Figure 1.5

HW-Lecture 3& 4

Name. ______________.   Student No. _______________   2024/3/20

1. （1）Write Boolean equations in (a) sum-of-products and (b) product-of-sums canonical form for the truth table in Figure 1.

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Figure 1

（2）Minimize the Boolean equations from (1)

(3) Sketch a reasonably simple combinational circuit implementing the function from (2).

2. Simplify the following Boolean equations using Boolean theorems.

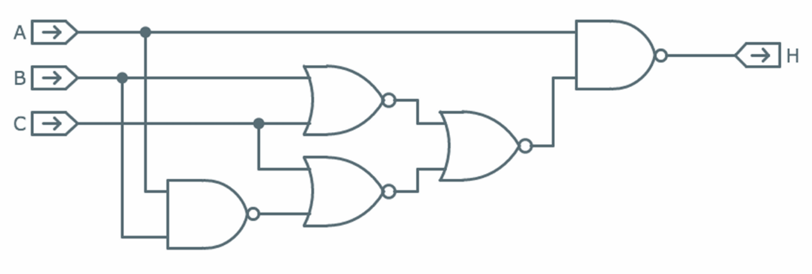
(1) $Y = A\overline{B} + \overline{A}B + AB + \overline{A+B}$　　(2) $Y = \overline{A + \overline{A}B + \overline{A}\overline{B}} + \overline{A + \overline{B}}$

(3) $Y = AC + \overline{A}D + \overline{C}D$　　(4) $Y = \overline{A}\overline{B} + \overline{A}B\overline{C} + (\overline{A + \overline{C}})$

(5) $Y = \overline{A}B\overline{C} + \overline{A}BC + ABC$　　(6) $Y = AC + \overline{B}C + B\overline{D} + C\overline{D} + A(B+\overline{C}) + \overline{A}BC\overline{D} + A\overline{B}DE$

3.

Consider the Boolean function $H(A,B,C)=\overline{A}+\overline{B}\cdot\overline{C}+A\cdot B\cdot\overline{C}$. Its truth table is shown to the right and a possible implementation is shown in the schematic below.



A	B	C	H
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	1
1	1	1	0

(1) Give a minimal sum-of-products expression for H. A couple of scratch 3-input Karnaugh map templates are provided for your convenience.

minimal sum-of-products expression for H: ________________________

	00	01	11	10
0				
1				

	00	01	11	10
0				
1				

(A) What is the largest number of product terms possible in a minimal sum-of-products expression for a 3-input, 1-output Boolean function?

Largest number of product terms possible: __________

	00	01	11	10
0				
1				

4. Find a minimal Boolean equation for the function in Figure blow. Remember to take advantage of the don't care entries.

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

5. A gate or set of gates is universal if it can be used to construct any Boolean function. For example, the set {AND, OR, NOT} is universal.

(1) Is a AND gate by itself universal? Why or why not?

(2) Is a NAND gate by itself universal? Why or why not?

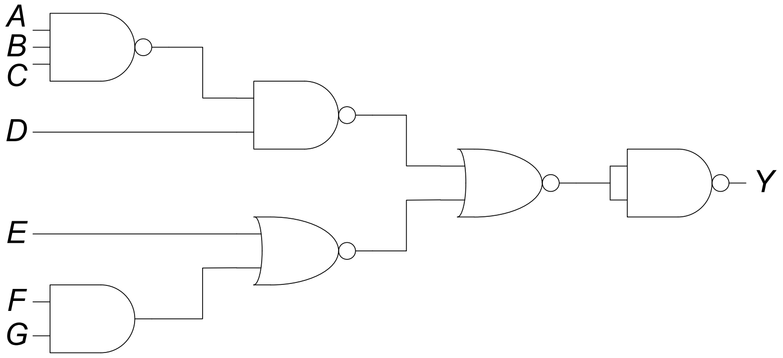
6. A minority gate has three inputs (call them A, B, C) and one output (call it Y). The output will be 0 if two or more of the inputs are 1, and 1 if two or more of the inputs are 0.

(1). Give a *minimal* **sum-of-products** Boolean expression for the minority gates output Y, in terms of its three inputs A, B, and C.

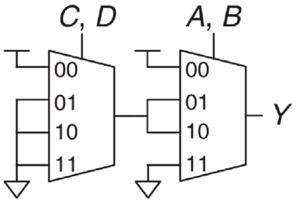
Minimal SOP expression: Y = ________________________

(2). Is a minority gate **universal**, in the sense that using only minority gates (along with constants 0 and 1) its possible to implement arbitrary combinational logic functions?

7. Using De Morgan equivalent gates and bubble pushing methods, redraw the circuit in Figure below so that you can find the Boolean equation by inspection. Write the Boolean equation.



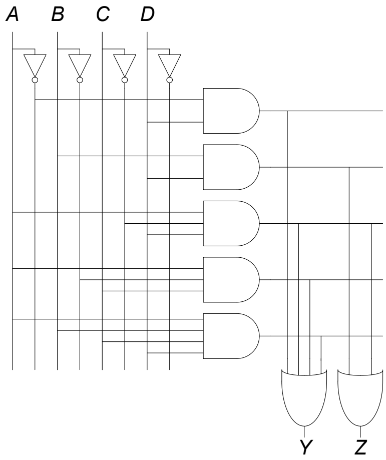
8. Write a minimized Boolean equation for the function performed by the circuit in the Figure blow.



9. **Two level logic**

(1) Write Boolean equations for the circuit in Figure blow. You need NOT minimize the equations.

(2) Minimize the Boolean equations you get from (1) and sketch an improved circuit with the same function.



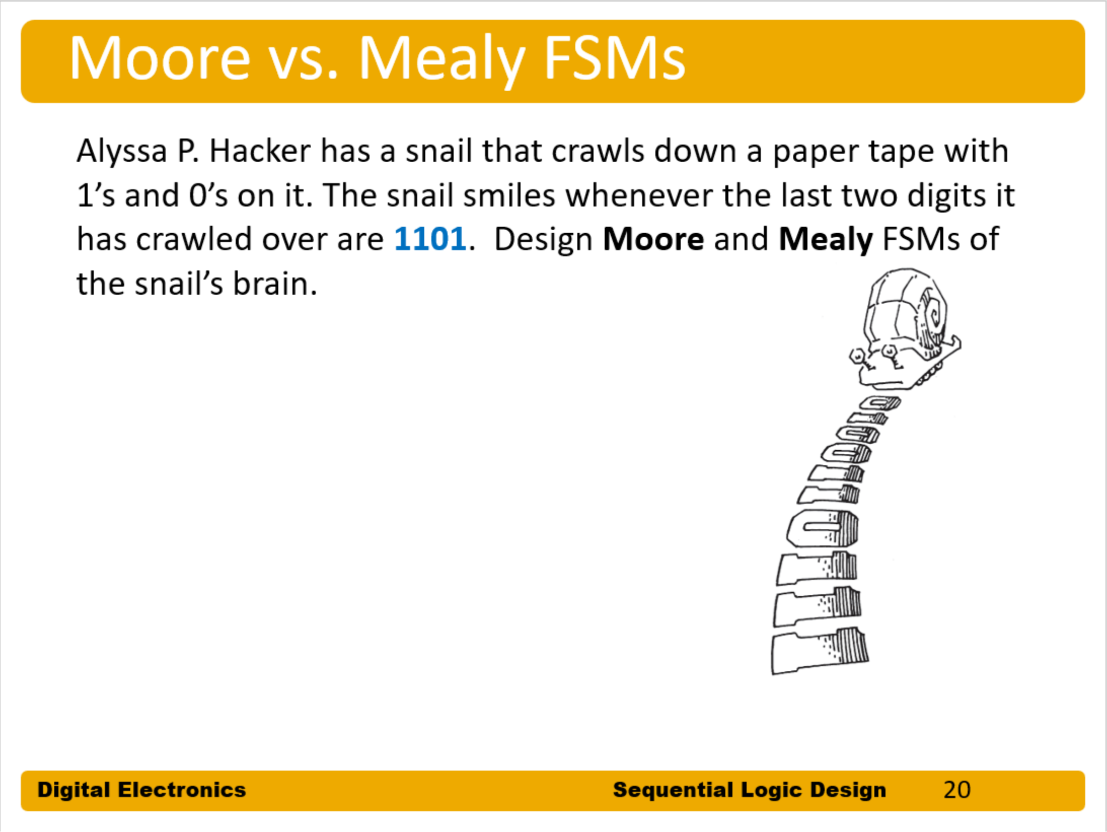
10. Boolean equation simplification

(1) Simplify the Boolean equation: $Y = BC + \overline{A}\overline{B}\overline{C} + B\overline{C}$

(2) Implement the function using an 8:1 multiplexer

(3) a 4:1 multiplexer and no other gates

(4) a 2:1 multiplexer, one OR gate, and an inverter



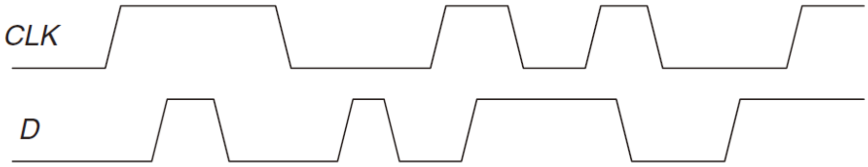
Moore vs. Mealy FSMs

Alyssa P. Hacker has a snail that crawls down a paper tape with 1’s and 0’s on it. The snail smiles whenever the last two digits it has crawled over are 1101. Design **Moore** and **Mealy** FSMs of the snail’s brain.

What to turn in：

1. A completed State Transition Diagram for the FSM.
2. Tables listing (1) next state in terms of current state and inputs, and (2) output in terms of current state and inputs.
4. The binary encoding for each state.
5. The revised copy of your tables, using your binary encoding.
6. Boolean logic equations for the outputs and each bit of the next state in terms of the state and inputs.
7. schematic of the FSM.
8. Verilog code for the FSM design and testbench, snapshot of the simulation waveform

1.Given the input waveforms shown in Figure blow, sketch the output *Q_latch* of a D latch, and output Q_ff of a D flip-flop



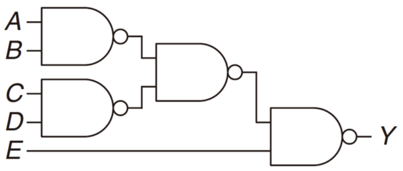
2.The *toggle* (*T*) *flip-flop* has one input, *CLK*, and one output, *Q*. On each rising edge of *CLK*, *Q* toggles to the complement of its previous value.Draw a schematic for a T flip-flop using a D flip-flop and an inverter.

3.A JK flip-flop receives a clock and two inputs, J and K. On the rising edge of the clock, it updates the output, Q. If J and K are both 0, Q retains its old value. If only J is 1, Q becomes 1. If only K is 1, Q becomes 0. If both J and K are 1, Q becomes the opposite of its present state.

(a) Construct a JK flip-flop, using a D flip-flop and some combinational logic.

(b) Construct a D flip-flop, using a JK flip-flop and some combinational logic.

Determine the propagation delay and contamination delay of the circuit in Figure below. Use the gate delays given in the Table.



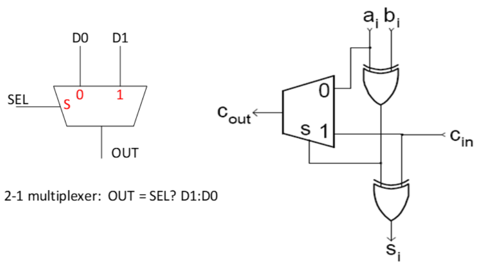
Gate	t_{pd} (ps)	t_{cd} (ps)
NOT	15	10
2-input NAND	20	15
3-input NAND	30	25
2-input NOR	30	25

7. Figure. 7 is the schematic of a 1-bit full adder cell.

(1) please give a brief analyzation of how the carry out signal (C_{out}) is generated

component	Delay time (ns)
XOR	3ns
2-1 multiplexer: from D0/D1 to OUT	1ns
2-1 multiplexer: from SEL to OUT	1.5ns

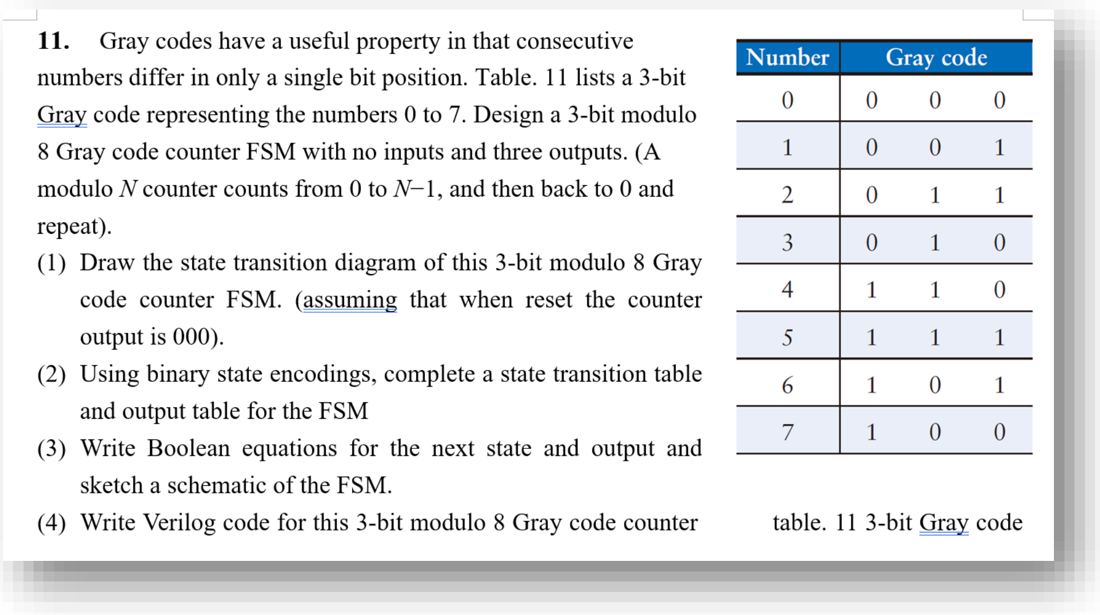
table. 7



Figuire.7

(2) build a 4-bit ripple carry adder based on this 1 bit full adder cell. Draw the block diagram.

(3) The delay time of the 2-1 multiplexer and XOR gate is given in table.7. For a 4-bit ripple carry adder based on this adder cell, calculate the delay time of the critical path.



11. Gray codes have a useful property in that consecutive numbers differ in only a single bit position. Table. 11 lists a 3-bit Gray code representing the numbers 0 to 7. Design a 3-bit modulo 8 Gray code counter FSM with no inputs and three outputs. (A modulo *N* counter counts from 0 to *N*−1, and then back to 0 and repeat).

(1) Draw the state transition diagram of this 3-bit modulo 8 Gray code counter FSM. (assuming that when reset the counter output is 000).

(2) Using binary state encodings, complete a state transition table and output table for the FSM

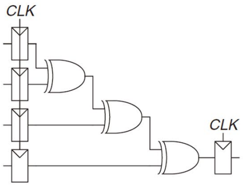
(3) Write Boolean equations for the next state and output and sketch a schematic of the FSM.

(4) Write Verilog code for this 3-bit modulo 8 Gray code counter

Number	Gray code		
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

table. 11 3-bit Gray code

Exercise 3.33 Ben Bitdiddle has designed the circuit in Figure 3.74 to compute a registered four-input XOR function. Each two-input XOR gate has a propagation delay of 100 ps and a contamination delay of 55 ps. Each flip-flop has a setup time of 60 ps, a hold time of 20 ps, a clock-to-*Q* maximum delay of 70 ps, and a clock-to-*Q* minimum delay of 50 ps.



(a) If there is no clock skew, what is the maximum operating frequency of the circuit?

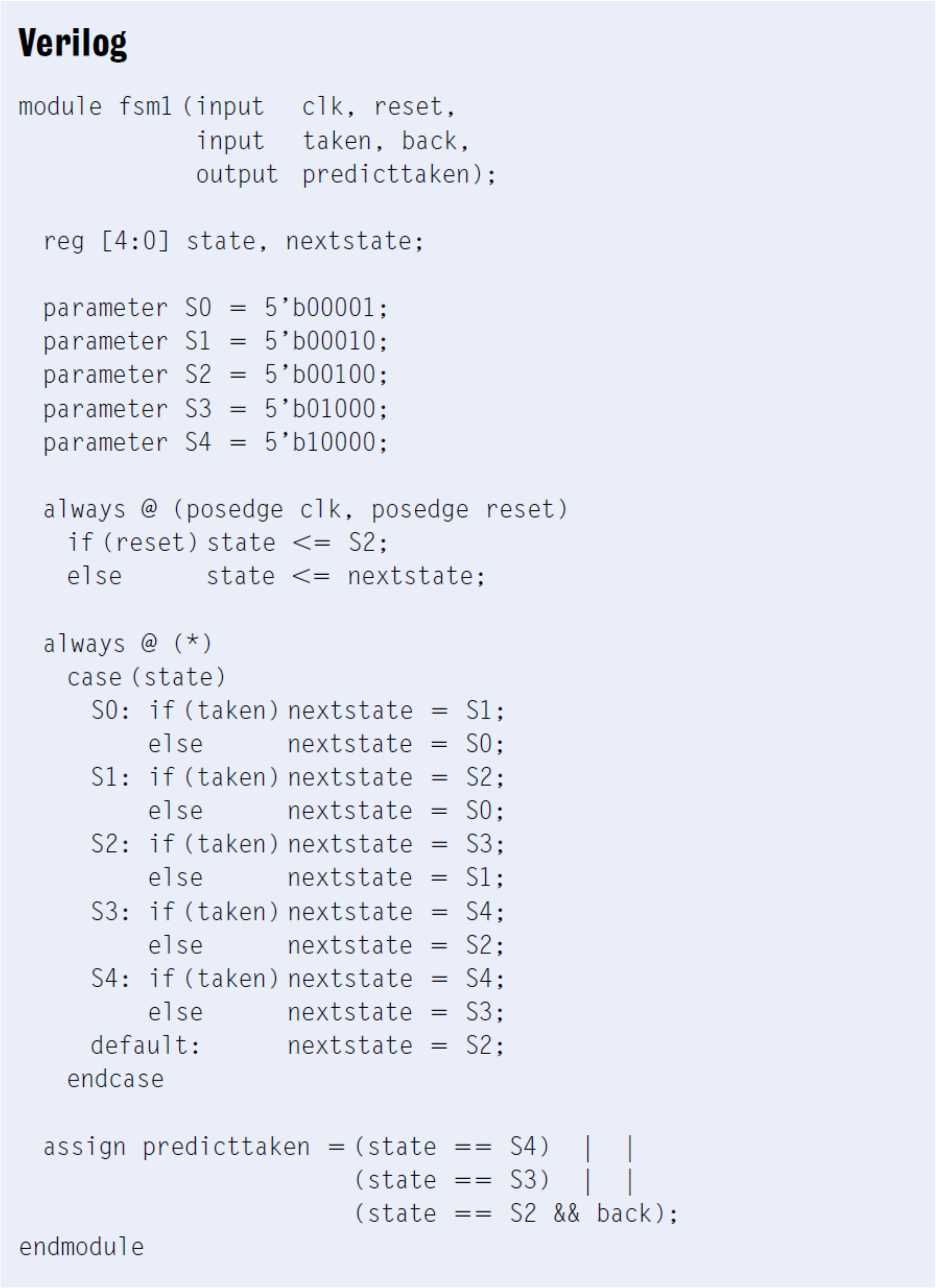
(b) How much clock skew can the circuit tolerate if it must operate at 2 GHz?

(c) How much clock skew can the circuit tolerate before it might experience a hold time violation?

(d) Alyssa P. Hacker points out that she can redesign the combinational logic between the registers to be faster *and* tolerate more clock skew. Her improved circuit also uses three two-input XORs, but they are arranged differently. What is her circuit? What is its maximum frequency if there is no clock skew? How much clock skew can the circuit tolerate before it might experience a hold time violation?

FSM-1

Sketch the state transition diagram for the FSM described by the following HDL code. An FSM of this nature is used in a branch predictor on some microprocessors.



Verilog

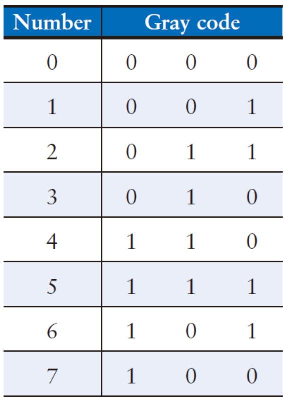
```
module fsm1(input   clk, reset,
            input   taken, back,
            output  predicttaken);

  reg [4:0] state, nextstate;

  parameter S0 = 5'b00001;
  parameter S1 = 5'b00010;
  parameter S2 = 5'b00100;
  parameter S3 = 5'b01000;
  parameter S4 = 5'b10000;

  always @ (posedge clk, posedge reset)
    if(reset) state <= S2;
    else      state <= nextstate;

  always @ (*)
    case(state)
      S0: if(taken) nextstate = S1;
          else      nextstate = S0;
      S1: if(taken) nextstate = S2;
          else      nextstate = S0;
      S2: if(taken) nextstate = S3;
          else      nextstate = S1;
      S3: if(taken) nextstate = S4;
          else      nextstate = S2;
      S4: if(taken) nextstate = S4;
          else      nextstate = S3;
      default:      nextstate = S2;
    endcase

  assign predicttaken = (state == S4)  |  |
                        (state == S3)  |  |
                        (state == S2 && back);
endmodule
```

FSM-2

Alyssa P. Hacker's snail has a daughter with a Mealy machine FSM brain. The daughter snail smiles whenever she slides over the pattern 1101 or the pattern 1110. Sketch the state transition diagram for this happy snail using as few states as possible. Write Verilog code(including testbench) for this FSM, run simulation to verify you design.

FSM-3

(1). Gray codes have a useful property in that consecutive numbers differ in only a single bit position. Table blow lists a 3-bit Gray code representing the numbers 0 to 7. Design a 3-bit modulo 8 Gray code counter FSM with no data inputs and three outputs. (A modulo N counter counts from 0 to $N - 1$, then repeats. For example, a watch uses a modulo 60 counter for the minutes and seconds that counts from 0 to 59.) When reset, the output should be 000. On each clock edge, the output should advance to the next Gray code. After reaching 100, it should repeat with 000. Draw state transition diagram for this FSM.



Number	Gray code		
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

(2). Extend your modulo 8 Gray code counter to be an UP/DOWN counter by adding an UP input. If UP = 1, the counter advances to the next number. If UP = 0, the counter retreats to the previous number. write Verilog code and testbench for this FSM, compile and simulate your design.

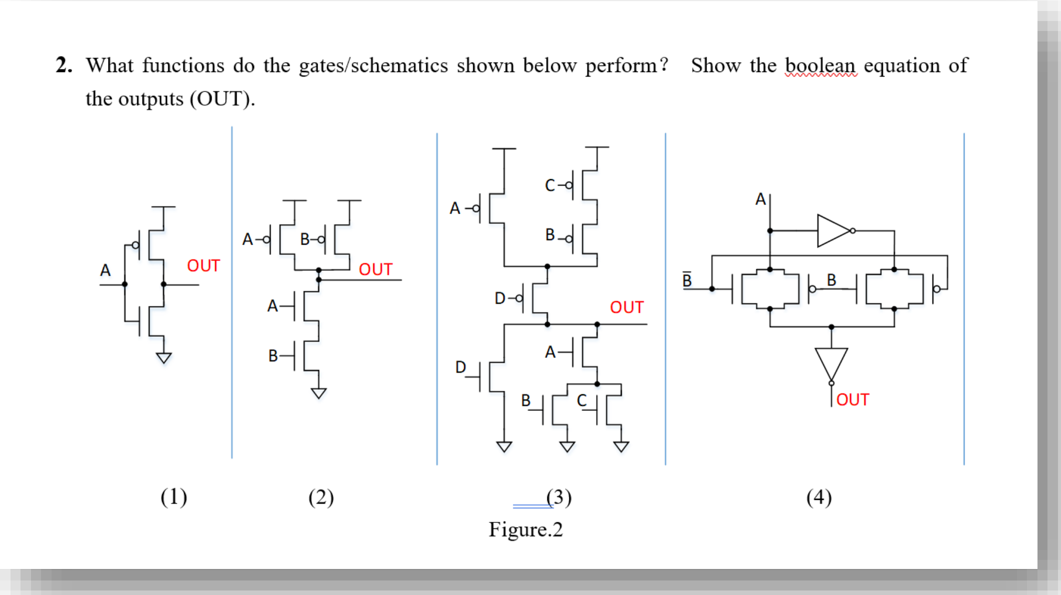
4. Timing

A field programmable gate array (FPGA) uses configurable logic blocks (CLBs) rather than logic gates to implement combinational logic. The Xilinx Spartan 3 FPGA has propagation and contamination delays of 0.61 and 0.30 ns, respectively, for each CLB. It also contains flip-flops with propagation and contamination delays of 0.72 and 0.50 ns, and setup and hold times of 0.53 and 0 ns, respectively.

(a) If you are building a system that needs to run at 40 MHz, how many consecutive CLBs can you use between two flip-flops? Assume there is no clock skew and no delay through wires between CLBs.

(b) Suppose that all paths between flip-flops pass through at least one CLB. How much clock skew can the FPGA have without violating the hold time?

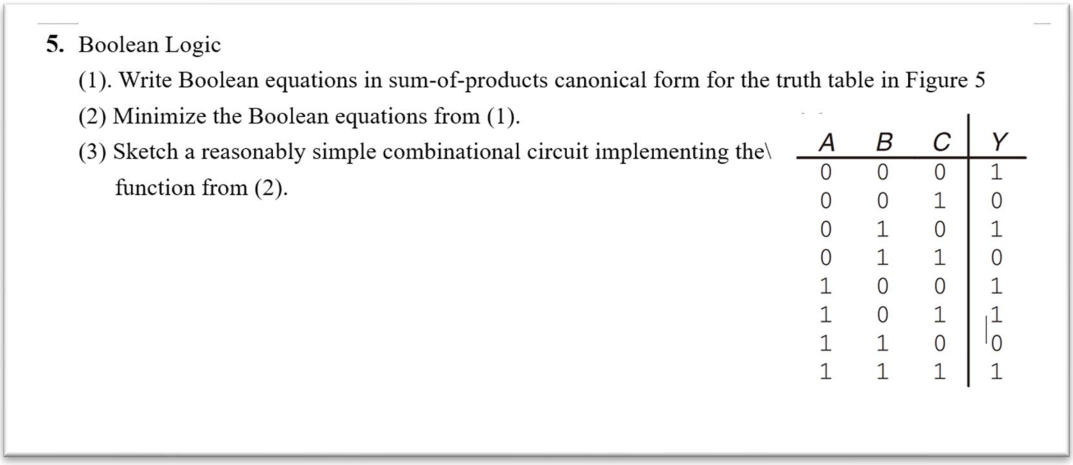
1. Draw the symbol, Boolean equation, transistor level schematic and truth table for a three-input NOR gate.



2. What functions do the gates/schematics shown below perform？ Show the boolean equation of the outputs (OUT).

(1) (2) (3) (4)

Figure.2



5. Boolean Logic

(1). Write Boolean equations in sum-of-products canonical form for the truth table in Figure 5

(2) Minimize the Boolean equations from (1).

(3) Sketch a reasonably simple combinational circuit implementing the\ function from (2).

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1