1	/18
2	/20
3	/17
4	/19
5	/16
6	/10

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

6.191 Computation Structures
Spring 2025

Quiz #1

Name	*Athena login name*	*Score*

Recitation section

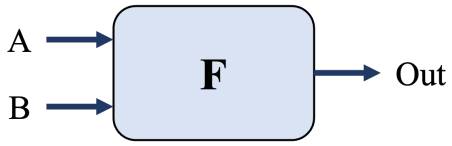
☐ WF 10, 34-302 (Hilary)	☐ WF 2, 34-302 (Raymond)	☐ WF 12, 35-308 (Keshav)
☐ WF 11, 34-302 (Hilary)	☐ WF 3, 34-302 (Raymond)	☐ WF 1, 35-308 (Keshav)
☐ WF 12, 34-302 (Ezra)	☐ WF 10, 35-308 (Harry)	☐ WF 2, 8-205 (Vedantha)
☐ WF 1, 34-302 (Ezra)	☐ WF 11, 35-308 (Harry)	☐ WF 3, 8-205 (Vedantha)
		☐ opt-out

Please enter your name, Athena login name, and recitation section above. Enter your answers in the spaces provided below. Show your work for potential partial credit. You can use the extra white space and the back of each page for scratch work.

Problem 1. Digital Abstraction (18 points)

The F module follows the voltage transfer characteristic given below. Assume that V_A and V_B are within the range of 0V to 6V, inclusive.

$$\mathbf{V_{out}} = \begin{cases} \mathbf{5.9V} & (\mathbf{V_A + 0.8 * V_B \leq 3.6V}) \\ \mathbf{1.0V} & (\mathbf{V_A + 0.8 * V_B > 3.6V}) \end{cases}$$



(A) (3 points) Given the above specification for module F, determine the Boolean expression that module F(A, B) implements. Express your answer in terms of A and B.

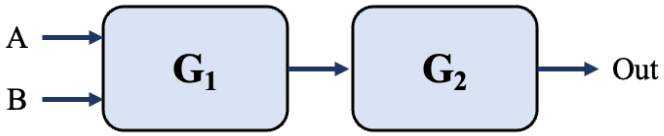
Boolean Expression for F(A, B): Out = _____

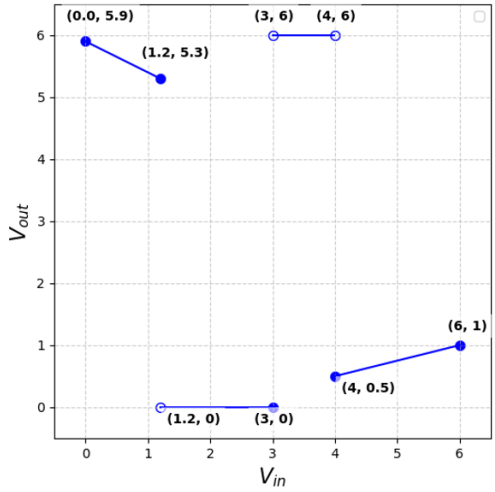
(B) (5 points) Identify the signaling specification parameters, ($V_{OL}, V_{IL}, V_{IH}, V_{OH}$), that maximize both the low noise margin and the high noise margin for module F. Then, provide the noise immunity for module F. You must show your work.

$\boldsymbol{V_{OL}} =$ _____ , $\boldsymbol{V_{IL}} =$ _____, $\boldsymbol{V_{IH}} =$ _____, $\boldsymbol{V_{OH}} =$ _____

Noise Immunity: _____

The G module, which implements the same Boolean function as the F module, consists of two subcircuits G_1 and G_2. G_1 is either the $max(V_A, V_B)$ or $min(V_A, V_B)$. You will need to figure out which. G_2 follows the voltage transfer characteristic shown to the right below.





(C) (3 points) In order for G to implement the same Boolean function as F, which function of (V_A, V_B) should you use for G_1: $max(V_A, V_B)$ or $min(V_A, V_B)$? Explain why.

G_1 is (select one): $\mathbf{max(V_A, V_B)}$ $\mathbf{min(V_A, V_B)}$

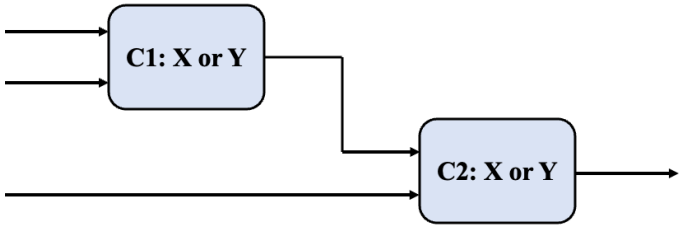
Explanation:

(D) (4 points) Identify the signaling specification parameters, $(V_{OL}, V_{IL}, V_{IH}, V_{OH})$, that maximize the noise immunity of G_2. Also, specify the resulting noise immunity.

$\boldsymbol{V_{OL}} =$ _____ , $\boldsymbol{V_{IL}} =$ _____ , $\boldsymbol{V_{IH}} =$ _____ , $\boldsymbol{V_{OH}} =$ _____

Noise Immunity: _____

(E) (3 points) Now, consider two new modules, X and Y, which have the following signaling specification parameters: Module X $(V_{OL}, V_{IL}, V_{IH}, V_{OH})$ = (1V, 2V, 5V, 6V), module Y $(V_{OL}, V_{IL}, V_{IH}, V_{OH})$ = (0.5V, 1.5V, 4V, 4.5V). You want to combine one X module with one Y module as shown below. Determine whether the X module should be placed in the position labeled C1 or C2 assuming that Y will go in the other one, so that the resulting circuit behaves as a valid combinational device. Explain your answer.

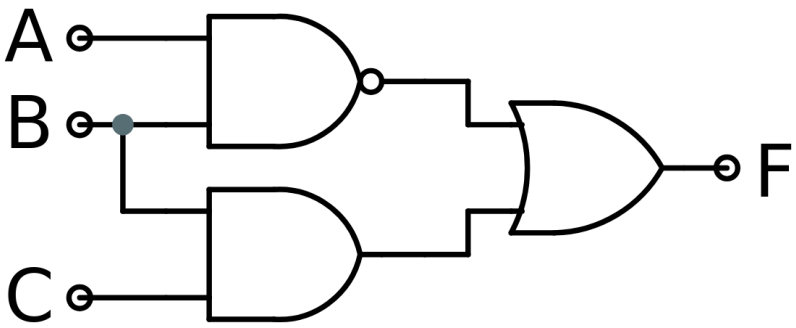


Placement of Module X : C1 … C2

Explanation:

Problem 2. Boolean Algebra (20 points)

(A) (3 points) Consider the following circuit that implements the Boolean expression $F(a, b, c) = \overline{ab} + bc$.



1. Fill in the truth table for expression F below.

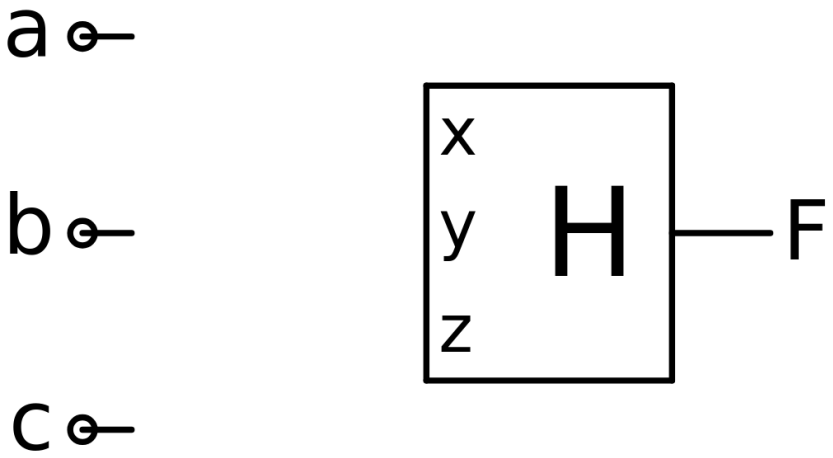
a	b	c	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

2. What is a minimal sum-of-products expression for $F(a, b, c)$?

Minimal sum-of-products form for F = _____

(B) (3 points) Oh no! Somehow you lost all of your OR gates! Draw a circuit implementation for F without using OR gates. **You may only use inverters and 2-input AND and NAND gates. Use the minimal number of gates for full points.**

(C) (2 points) You found some OR gates, but in the process you accidentally dropped and ruined all your NAND gates! Luckily, you also found gate H, which implements the expression $H(x, y, z) = \overline{x} + yz$. It turns out you can still implement F by using an H gate. Using a **single** additional gate which can be an **inverter**, or a **2-input AND** gate, or a **2-input OR** gate, connect the inputs *a*, *b*, and *c* to the inputs of gate H to produce the function F. **Hint:** You may want to implement something other than the minimal sum of products for F.



(D) (2 points) Use Boolean algebra or truth tables to demonstrate that your implementation of F out of H plus one gate, from part (C), is equivalent to F.

(E) (2 points) Determine whether the gate $H(x, y, z) = \overline{x} + yz$ is functionally complete (universal). If it is, prove it. Otherwise, explain why it is not.

(F) (4 points) Find **a minimal sum-of-products** expression for each of the following Boolean expressions. **Pay close attention to which variables are under the overbars.** To get credit for your answer, you must **use Boolean algebra axioms and properties and show your work!**

1. $a\overline{cd} + \overline{b}cde + \overline{a}\,\overline{c}\,e + de$

2. $\overline{b(\,\overline{ac} + b)}\left(a\overline{b}c + \overline{a}\right)d$

(G) (4 points) Determine whether each of the Boolean expressions below are satisfiable. If it is satisfiable, give an input assignment that make the expression satisfiable. You must provide a valid value for each variable. If it is not satisfiable, show why it is not.

1. $\left(\overline{d + \overline{a}\overline{b}}\right)(ab + cd)(ad + d)$

2. $\left(abc + \overline{d}\right)(\overline{c} + a)\left(\overline{d + \overline{b}}\right)$

Problem 3. CMOS Logic (17 points)

(A) (10 points) For each of the following four functions (specified as a Boolean expression or as a truth table), determine if it can be implemented as a single CMOS gate. If it can, draw the CMOS gate using a minimal number of FETs. If not, describe why it cannot be implemented as a single CMOS gate.

1. $F(A,B,C) = \bar{A}\bar{C} + A\bar{B}\bar{C}$

2. $\overline{F(A,B,C,D)} = AB + \overline{CD}$

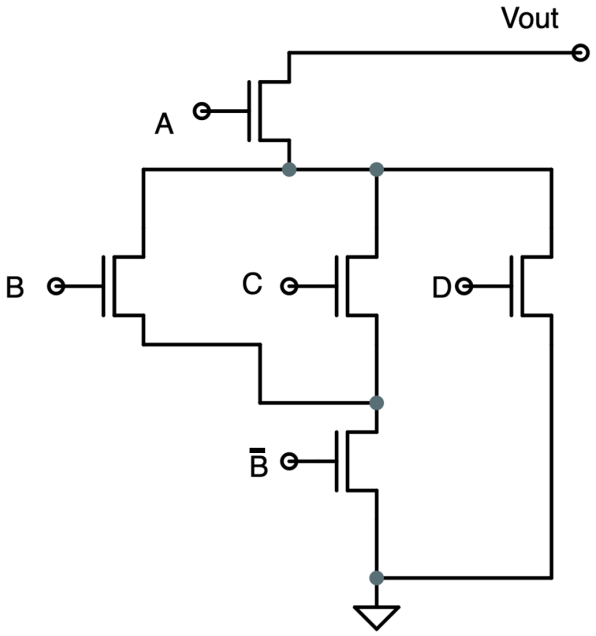
3.

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

4.

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(B) (2 points) Evaluate the output of the below circuit at the following points. We have only drawn the pull-down circuit. Assume the pullup is complementary. Notice that one of the inputs is $\overline{B}$.



What is Vout when A=1, B=1, C=0, D=0? _____

What is Vout when A=1, B=0, C=1, D=0? _____

(C) (3 points) Given that a function $F(A, B, C)$ can be implemented as a single CMOS gate, complete the truth table below. For each row, enter a 0 or 1 if you can deduce the value of F for that set of inputs, and enter ? if you cannot.

A	**B**	**C**	**F**
0	0	0	
0	0	1	0
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	1
1	1	1	

(D) (2 points) If some function G can **NOT** be implemented as a single CMOS gate, when can $\bar{G}$ be implemented as a single CMOS gate? Select one of the options and explain your answer. Use examples and/or counter examples in your explanation.

Always **Sometimes** **Never**

Explanation:

4. Combinational Minispec (19 points)

(A) (8 points) Complete the truth table for the following Minispec function.

```
function Bit#(3) f(Bit#(1) a, Bit#(2) b);
    Bit#(3) ret = 3'b100;
    case ({b[0], a})
        0: ret = {1'b1, zeroExtend(a) & b};
        1: ret = signExtend(a) + zeroExtend(b);
        3: ret = {~a, b};
        default: ret = 3'b110;
    endcase
    return ret;
endfunction
```

a	b[1]	b[0]	ret[2]	ret[1]	ret[0]
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

(B) (3 points) Below is an implementation of a parametric function `add_mod3#(n)` which takes two n-bit inputs `a` and `b` and returns a 2-bit output `(a+b) mod 3`.

```
function Bit#(2) add_mod3#(2) (Bit#(2) a, Bit#(2) b);
   Bit#(3) sum = zeroExtend(a) + zeroExtend(b);
   return truncate(sum % 3);
endfunction

function Bit#(2) add_mod3#(Integer n) (Bit#(n) a, Bit#(n) b);
   Bit#(2) r = 0;
   for(Integer i = 0; i < n; i = i + 2) begin
      Bit#(2) s = add_mod3#(2)(a[i+1:i], b[i+1:i]);
      r = add_mod3#(2)(r, s);
   end
   return r;
endfunction
```

The delay of this implementation of add_mod3#(n) in terms of n is Θ(_____)

Justification:

(C) (8 points) Implement a recursive version of `add_mod3#(n)` which achieves a better asymptotic delay. The base case `add_mod3#(2)` is provided for you. You can assume that `n` is a power of 2 and that $n \geq 2$. What is the delay of your implementation of `add_mod3#(n)`? Justify your answer.

```
function Bit#(2) add_mod3#(2) (Bit#(2) a, Bit#(2) b);
   Bit#(3) sum = zeroExtend(a) + zeroExtend(b);
   return truncate(sum % 3);
endfunction

function Bit#(2) add_mod3#(Integer n) (Bit#(n) a, Bit#(n) b);




endfunction
```

The delay of your implementation of add_mod3#(n) in terms of n is Θ(_____)

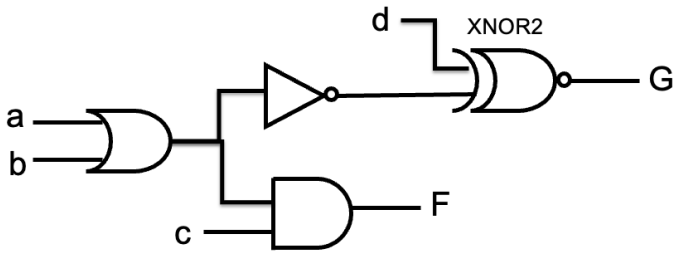
Justification:

Problem 5. Combinational and Sequential Logic Timing (16 points)

Two 6.1910 students have built a circuit that sends them reminders to complete the lecture questions at 9:50am. They use a logic family with the following characteristics for their circuits (in *nanoseconds*):

Gate	t_{PD}	t_{CD}	t_{SETUP}	t_{HOLD}
AND2	1.7	0.4	—	—
OR2	1.8	0.3	—	—
XNOR2	2.1	0.2	—	—
INV1	0.5	0.3	—	—
REG	2.4	0.9	1.2	0.8

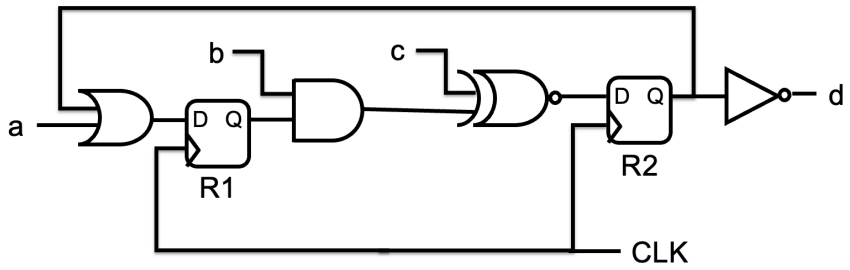
(A) (2 points) They initially design the combinational circuit below with two outputs, F and G. What are the t_{PD} and t_{CD} values of the combinational circuit?



t_{PD} (ns): _____

t_{CD} (ns): _____

(B) (4 points) They learn about sequential circuits and come up with the new circuit below. What is the minimum $\mathbf{t_{CLK}}$ value for which this circuit operates correctly? Write "N/A" if the circuit is not valid. For full credit, you must show that all setup and hold timing constraints are fully satisfied, or that one of the constraints is not satisfied.



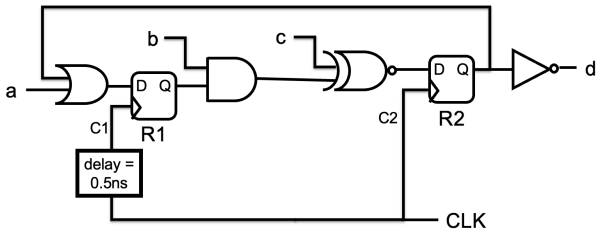
Minimum $\mathbf{t_{CLK}}$ (ns): _____

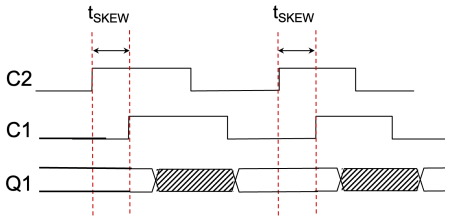
(C) (2 points) What are the $\mathbf{t_{PD}}$ and $\mathbf{t_{CD}}$ values of the sequential circuit?

$\mathbf{t_{PD}}$ (ns): _____

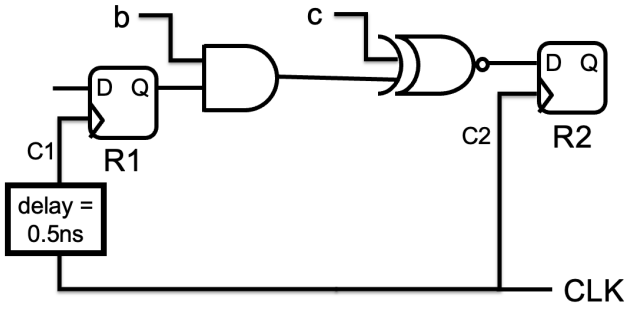
$\mathbf{t_{CD}}$ (ns): _____

The students realize that because the CLK → R1 wire is much longer than the CLK → R2 wire, the clock signal does not arrive at R1 and R2 at the same time. As shown below, this circuit has clock skew where the CLK signal arrives at register R1 $\mathbf{t_{SKEW}}$ = 0.5ns after it arrives at register R2.



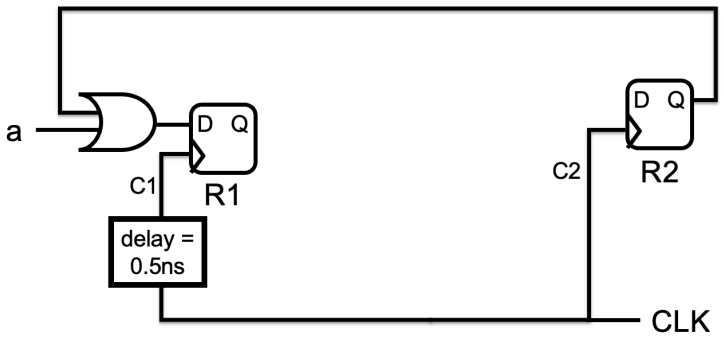


(D) (3 points) To ensure their circuit operates correctly, they remove some logic to focus only on the R1 → R2 path. What is the minimum $\mathbf{t_{CLK}}$ value for which this circuit operates correctly? Write "N/A" if the circuit is not valid. For full credit, you must show that all setup and hold timing constraints are fully satisfied, or that one of the constraints is not satisfied.



Minimum t_{CLK} (ns): _____

(E) (2 points) They then analyze the R2 → R1 path by creating the simplified circuit below. What is the minimum $\mathbf{t_{CLK}}$ value for which this circuit operates correctly? Write "N/A" if the circuit is not valid. For full credit, you must show that all setup and hold timing constraints are fully satisfied, or that one of the constraints is not satisfied.



Minimum $\mathbf{t_{CLK}}$ (ns): _____

(F) (3 points) The students find new registers REGA and REGB that they consider substituting in for both R1 and R2 in their sequential circuit with clock skew. Which register should they use to ensure that their circuit operates correctly and minimizes the value of $\mathbf{t_{CLK}}$? Circle which register type to select and explain why you should choose that register type.

Gate	$\mathbf{t_{PD}}$	$\mathbf{t_{CD}}$	$\mathbf{t_{SETUP}}$	$\mathbf{t_{HOLD}}$
REG	2.4	0.9	1.2	0.8
REGA	2.8	0.9	1.0	0.6
REGB	2.2	0.6	0.8	0.5

Circle one: REG REGA REGB

Explanation:

Problem 6. A Variant Lock (10 points)

In lecture, we saw an example of a digital combination lock which was designed such that as long as the last 4 inputs were correct, then the lock would unlock. You are now tasked with designing a different type of lock which samples 3 inputs at a time and **resets** to its initial state every time you enter an invalid 3-input combination. This lock has the following specifications:

- It begins in the `Start` state.
- It receives a 1-bit input ("0" or "1") on every cycle
- It produces a 1-bit output ("Unlock" signal) on every cycle
- Inputs are sampled in groups of 3
- UNLOCK (U) is 1 if the sampled group of 3 inputs matched the "combination": 101, otherwise UNLOCK = 0.
- If the sampled group of 3 inputs did not match the combination, then go back to the Start state.

The following examples should clarify the expected operation of this lock:

- **Receive 101.** Lock opens. FSM reaches a state where Unlock=1 after receiving 3 bits.
- **Receive 1101.** FSM in non-Start state with Unlock = 0.
- **Receive 10001.** FSM in non-Start state with Unlock=0.
- **Receive 100100.** FSM back in start state with Unlock=0.
- **Receive 000101.** Lock opens.
- **Receive 101101.** Lock opens after 3rd bit, and then again after 6th bit.

(A) (10 points) Draw the state transition diagram for this combination lock. Your starting state should be labeled `Start`. Label all other states with a unique letter (e.g., X). Make sure to specify the value of the output U for each state. Also, make sure to label each transition with a 0, 1, or 0/1 (if arc can be followed for either a 0 or 1 input). Your FSM must handle all possible input combinations. For full credit, you should minimize the number of states used in your FSM. *The following page has been intentionally left blank in case you want more space to draw your FSM.*

END OF QUIZ 1!