1	/19
2	/22
3	/20
4	/17
5	/22

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

6.191 Computation Structures
Spring 2024

Quiz #1

Name	*Athena login name*	*Score*

Recitation section

- ☐ WF 10, 34-302 (Wendy)
- ☐ WF 11, 34-302 (Wendy)
- ☐ WF 12, 34-302 (Adrianna)
- ☐ WF 1, 34-302 (Adrianna)
- ☐ WF 2, 34-302 (Catherine)
- ☐ WF 3, 34-302 (Catherine)
- ☐ WF 12, 35-308 (Shabnam)
- ☐ WF 1, 35-308 (Shabnam)
- ☐ 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 (19 points)

The F module below outputs 6V when $V_A + 0.5 * V_B > 2.5V$ for 25ns and outputs 0.5V when $V_A + 0.5 * V_B < 1.5V$ for 25ns. Furthermore, V_A and V_B are both between 0 and 6V. This is summarized in the equation below:

$$V_{out} = \begin{cases} 6V, & V_A + 0.5 * V_B > 2.5V \\ 0.5V, & V_A + 0.5 * V_B < 1.5V \\ 0 \leq ??? \leq 6V, & otherwise \end{cases}$$

(A) (3 points) If we apply constant V_A, V_B for 25ns and then measure a $\boldsymbol{V_{out} = 1V}$, what can we conclude about V_B?

C1: $V_B < 3V$
C2: $V_B \leq 5V$
C3: $V_B > 5V$
C4: $V_B \geq 3V$
C5: None of the above

Best conclusion about $\boldsymbol{V_B}$ (Circle one): C1 … C2 … C3 … C4 … C5

(B) (3 points) If we apply constant V_A, V_B for 25ns and then measure a $\boldsymbol{V_{out} = 6V}$, what can we conclude about V_A?

C1: $V_A < 1.5V$
C2: $V_A \leq 2.5V$
C3: $V_A > 2.5V$
C4: $V_A \geq 1.5V$
C5: None of the above

Best conclusion about $\boldsymbol{V_A}$ (Circle one): C1 … C2 … C3 … C4 … C5

(C) (3 points) What Boolean expression does the F module implement? Specify an equation using A and B.

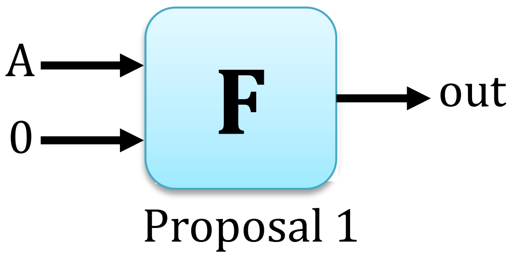
Boolean Expression: out = ________________

(D) (4 points) What are the parameters that produce a maximum noise immunity for the F module shown above?

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

Noise Immunity = ________________

We begin exploring configurations of the F module that will perform as a buffer. We consider two different buffer proposals:



(E) (4 points) Select the proposal that gives the best noise immunity, and specify parameters that produce a maximum noise immunity for that proposal.

Best Proposal (circle one): 1 2

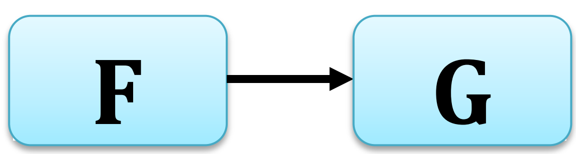
$\boldsymbol{V_{OL}}$ = _____ , $\boldsymbol{V_{IL}}$ = _____ , $\boldsymbol{V_{IH}}$ = _____ , $\boldsymbol{V_{OH}}$ = _____

Noise Immunity: ________________

(F) (2 points) Suppose we have a new device G with signaling thresholds defined relative to G's supply voltage $V_{DD,G}$:

- $V_{OL} = 0.1\ V_{DD,G}$
- $V_{IL} = 0.3\ V_{DD,G}$
- $V_{IH} = 0.8\ V_{DD,G}$
- $V_{OH} = 0.95\ V_{DD,G}$

We want to connect the original device F to this new G device to make the following circuit:



Under what range of supply voltages $V_{DD,G}$ will the connection between F and G have noise margins of at least 0.4V?

Range of Supply Voltages: _________V $\leq V_{DD,G} \leq$ _________V

Problem 2. Boolean Algebra (22 points)

(A) (4 points) The function F takes in three Boolean variables, x, y, z and returns:

$$F(x, y, z) = x\,y\,z + \overline{(\bar{x} + y\,z)} + \overline{(\bar{x}\,y)}\,z$$

1. What is the minimal sum-of-products expression for $F(x, y, z)$? **Pay close attention to which variables are included under the underbars.**

   Minimal sum of products for F = ______________________________

2. Use the minimal sum of products for F to help you fill in the truth table below.

x	*y*	*z*	*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	

3. What is the normal form expression for $F(x, y, z)$?

   Normal form for F = ______________________________

(B) (3 points) Draw the circuit that implements F **using 3 or fewer gates**. You **may only use inverters and 2-input OR, NOR, AND,** and **NAND** gates in your circuit.

(C) (12 points) Find **a minimal sum-of-products** expression for each of the following Boolean expressions. **Pay close attention to which variables are included under the underbars.**

1. $\overline{(\bar{a}\bar{c})}\,(a+\bar{c}) + b + \overline{(\bar{b}+\bar{c})}$

2. $\bar{a}c + a\bar{b} + \overline{(a+c)} + ab$

3. $c(ab + a\bar{b}) + \overline{(a+b)}\,c$

4. $((\overline{b+\bar{b}}) + (a+c)(a+\bar{c}))\,(\bar{a}\,(\bar{a}+b))$

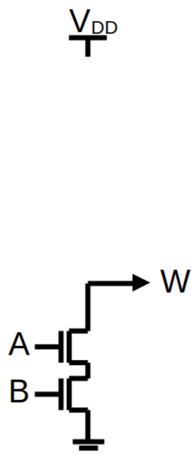
(D) (3 points) Is $\boldsymbol{G(a,b) = \bar{a}b}$ a universal function (i.e., can you build any Boolean function using only *G* gates)? If it is universal, then prove it. If it is not, explain why not.

Problem 3. CMOS Logic (20 points)

(A) (8 points) Octavian the octopus is helping his sister clean up her workshop. He has found a box containing parts meant to be CMOS gates. However, they're all missing some transistors.

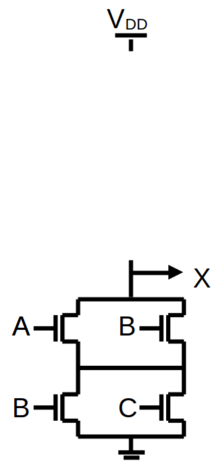
Help Octavian complete each of the following gates by drawing in the missing FETs. Then, provide the Boolean expression computed by the gate (you do **NOT** need to expand the expression into minimal sum of product form).

(i)



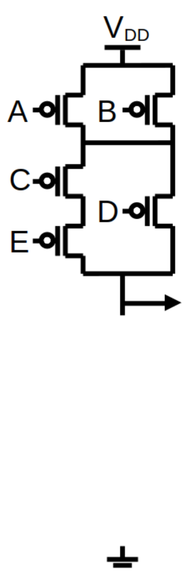
Boolean expression for W:
W(A,B) = ____________________

(ii)



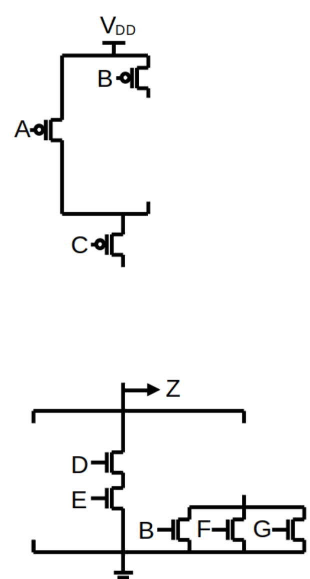
Boolean expression for X:
X(A,B,C) = ____________________

(iii)



Boolean expression for Y:
Y(A,B,C,D,E) = ________________

(iv)



Boolean expression for Z:
Z(A,B,C,D,E,F,G) = ________________

(B) (6 points) Octavian has found a schematic containing a truth table for function F. His sister's notes indicate that F can be implemented as a single CMOS gate.

Unfortunately, a few of the entries in the output column have been smudged beyond recognition. Help Octavian fill out the truth table, and then draw the single CMOS gate that would implement function F. **For full credit, you must use a minimum number of FETs to build your CMOS gate.**

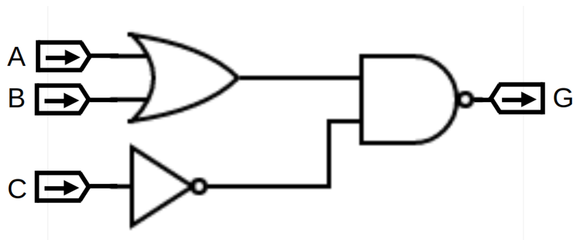
A	**B**	**C**	**F**
0	0	0	
0	0	1	
0	1	0	1
0	1	1	0
1	0	0	
1	0	1	1
1	1	0	0
1	1	1	

CMOS gate drawing:

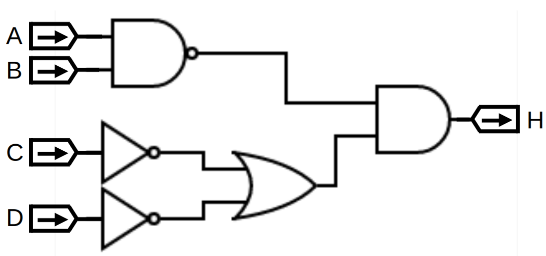
(C) (6 points) Octavian discovered some circuits in an unlabeled box. Help him determine if each of these circuits can be implemented as a single CMOS gate.

If it's possible, draw the single CMOS gate that implements the circuit **using a minimum number of FETs**. Otherwise, explain why not.

(i) **CMOS gate or explanation:**



(ii) **CMOS gate or explanation:**

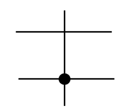


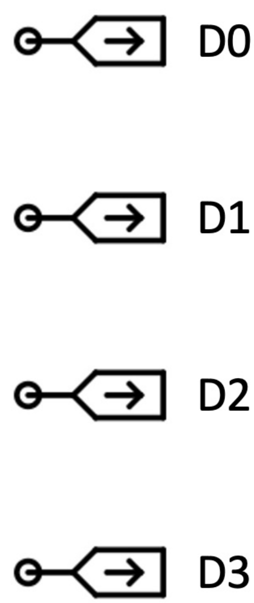
Problem 4: Combinational Circuits in Minispec (17 points)

A decoder is a combinational circuit that uniquely maps an n-bit input to a 2^n-bit output. For each possible input, only one bit of the output is high. This means you can select a single output based on the input values. The truth table for a 2-to-4 decoder is given below. This decoder assigns a value to each output wire and activates the appropriate one based on the value of {B1, B0}.

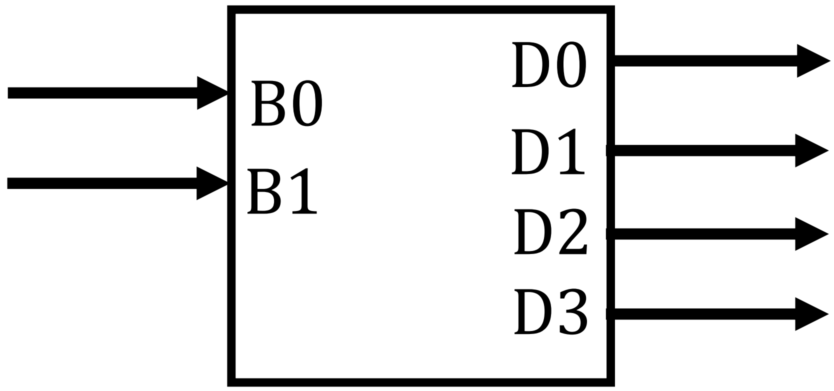
B1	**B0**	**D3**	**D2**	**D1**	**D0**
0	**0**	**0**	**0**	**0**	**1**
0	**1**	**0**	**0**	**1**	**0**
1	**0**	**0**	**1**	**0**	**0**
1	**1**	**1**	**0**	**0**	**0**

(A) (3 points) Draw a circuit that implements a 2-to-4 decoder using only inverters, 2-input ORs, and 2-input ANDs. Make sure to clearly label all inputs (B0, B1) and to connect to the provided output labels (D0, D1, D2, and D3). **If you have crossing wires, make sure to clearly label which wires are connected using a bold dot.** For example, the diagram on the right shows that the vertical wire is connected to the bottom horizontal wire but not to the top one.

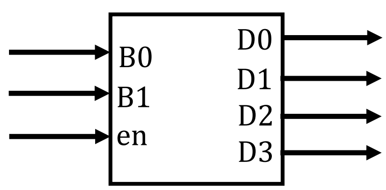


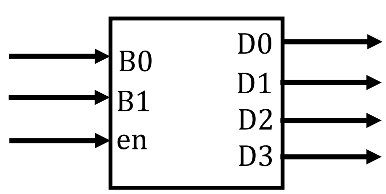


(B) (2 points) Using the block diagram provided to represent your circuit from part (A), add any necessary logic to convert the circuit to a 2-to-4 decoder **with Enable**. If Enable is 0, all output bits should be 0, and if Enable is 1, the outputs should follow the truth table above.



(C) (3 points) Using **only** two 2-to-4 decoders with Enable and a **single inverter**, implement a 3-to-8 decoder. Clearly label all inputs (B0-B2) and all outputs (D0-D7).



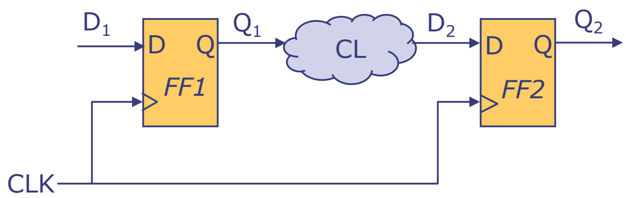


(D) (9 points) Write a parametric recursive function in Minispec that implements a decoder with enable. The two inputs of the function are an n-bit value (**b**) and a one-bit enable signal (**en**). The function outputs a 2^n bit output of all **0**s if the enable is 0 and $\mathbf{2^b}$ if enable is 1. We provided the first line for you to get you started.

```
function Bit#(2**n) decoder#(Integer n)(Bit#(n) b, Bit#(1) en);
```

Problem 5. Timing and FSMs (22 points)

Consider the circuit below, with the timing parameters for flip-flops shown in the table to the right.



Flip-flop timing parameters
t_{SETUP} = 0.3ns
t_{HOLD} = 0.1ns
$t_{PD,FF}$ = 0.2ns
$t_{CD,FF}$ = 0.1ns

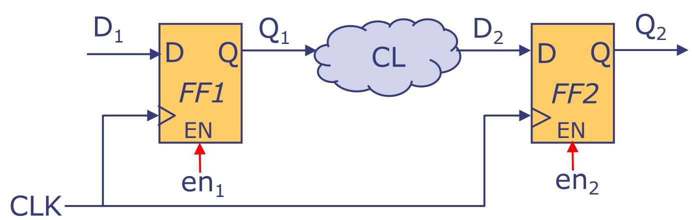
(A) (2 points) If we want t_{CLK}=1ns, what is the maximum propagation delay of the combinational logic CL? What is the minimum contamination delay of CL for the circuit to work properly?

Max $t_{PD,CL}$ (ns):________________

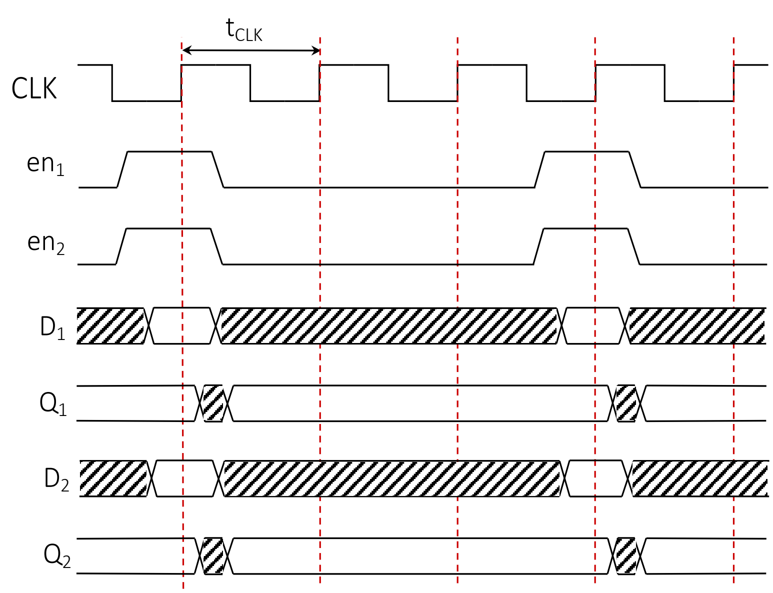
Min $t_{CD,CL}$ (ns):________________

Suppose we have an implementation of CL that is too slow for our target t_{CLK}. This would normally require increasing t_{CLK}, which can be undesirable (e.g., if this is part of a larger circuit, CL is the slowest component, and CL is used infrequently). Instead, in this problem we'll explore an alternative option: using a *multicycle path*.

The circuit to the right is similar to the one above, except that both flip-flops have an *enable* signal, EN. When EN=0, the flip-flop does not update the Q output at the rising edge of the clock, and the D input can change arbitrarily around a rising edge without affecting Q. In precise terms, the EN input must obey the flip-flop's setup and hold constraints, but the D input does not when EN=0.



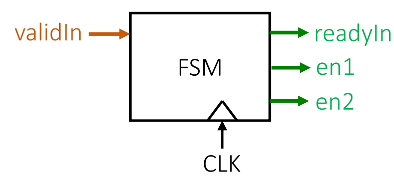
The timing diagram on the right shows how this circuit works for a 3-cycle multicycle path: by enabling the registers every third cycle, CL can spend three clock cycles computing each output.



(B) (4 points) If we want t_{CLK}=1ns, what is the maximum propagation delay of the combinational logic CL in the above 3-cycle path? What is the minimum t_{CD} for the circuit to work properly?

Max $t_{PD,CL}$ (ns):________________

Min $t_{CD,CL}$ (ns):________________

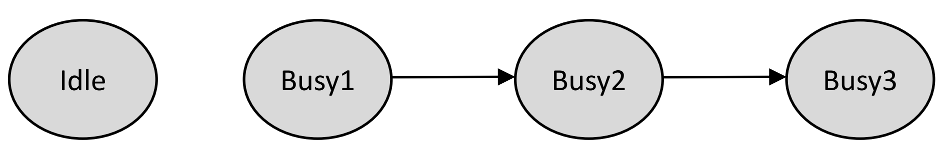


Instead of enabling both registers every third cycle, let's design an FSM that controls the registers' enable signals.

The FSM shown to the right has a single input, *validIn*, and three outputs: *readyIn*, *en1*, and *en2*. The FSM should work as follows:

- *validIn=1* indicates that the D input of FF1 has a new value for the circuit to process.
- *readyIn=1* indicates that the circuit is ready to process a new input value (i.e., it has completed processing the previous value).
- A new computation starts only when *readyIn* and *validIn* are both 1. To start a computation, the FSM sets *en1=1*, which makes FF1 sample its input value.
- Three cycles after the computation starts, the FSM must sample the output of CL, by setting *en2=1*. **The FSM must not sample CL earlier than 3 cycles, as that may cause metastability.**
- Immediately after the output of CL is sampled and available at the output Q2, the FSM can accept a new input. If *validIn* is always 1, **the FSM should start, and finish, a new computation every 3 cycles.**

(C) (7 points) Implement the FSM by (1) drawing and labeling all the missing transitions in the state-transition diagram, and (2) writing the Boolean equations for all the outputs (**note that the outputs may depend on both the state and the *validIn* input**).

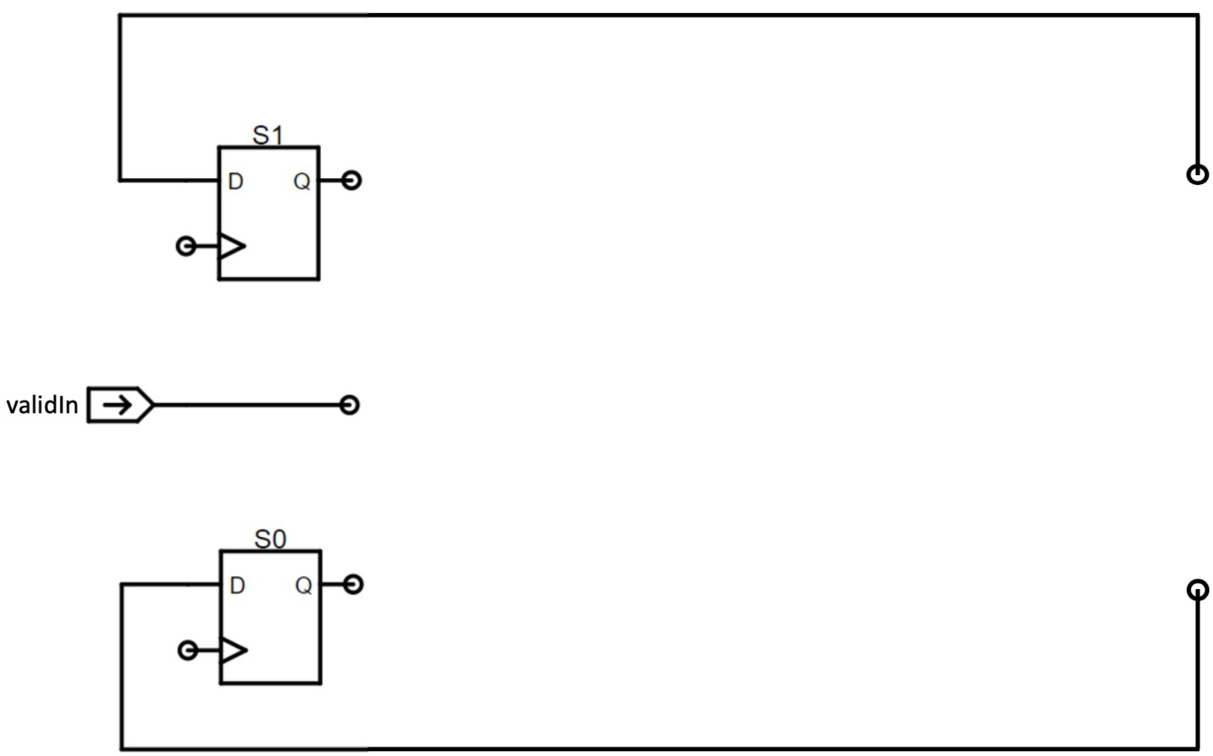



```
typedef enum {Idle, Busy1, Busy2, Busy3} State;

function Bit#(3) computeFsmOutputs(State state, Bool validIn);

  Bool readyIn = (state == __________ || state == ___________);

  Bool en1 = ____________________________________________________;

  Bool en2 = ____________________________________________________;
  …
```

(D) (5 points) Assume that states are encoded using 2-bit values as shown to the right. Use your FSM from part C to fill in the truth table below. Then implement the combinational circuit that computes the next state (*nextS1, nextS0*) given the current state (*S1,S0*) and the *validIn* input. Use the **validIn** input and **S1** and **S0** registers provided below. You can use NOT, AND, OR, and XOR gates. For full credit, your circuit should not use more than **eight** gates.

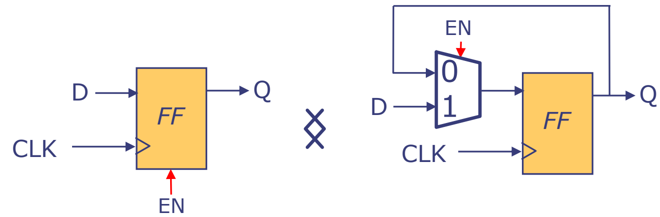
State	Encoding
Idle	00
Busy1	01
Busy2	10
Busy3	11

validIn	S1	S0	nextS1	nextS0
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		



(E) (2 points) Suppose that we implement the flip-flop with enable using a normal D flip-flop without an enable and a mux, as shown below. Will any mux implementation work correctly? If so, explain why. If not, give an example of how an inappropriate mux implementation could cause metastability.

Hint: Remember that D can change during the rising edge of the clock if EN=0, and think about the guarantees of the combinational contract.



(F) (2 points) To simplify our FSM, we want to get rid of **en2** and use a normal flip-flop that samples D every rising edge of the clock. What is the minimum contamination delay for CL that would let us do this?

Hint: Look back at the timing diagram and determine what D_2 must look like so that FF2 can sample it every cycle.

Min $t_{CD,CL}$ (ns):__________________

END OF QUIZ 1!