18-240: Structure and Design of Digital Systems

Electrical & Computer ENGINEERING

HW5 Solutions [8 problems, 64 points]

Covers lectures L10 – L12 *Due: 24 October 2023*

Drill Problems [32 points]

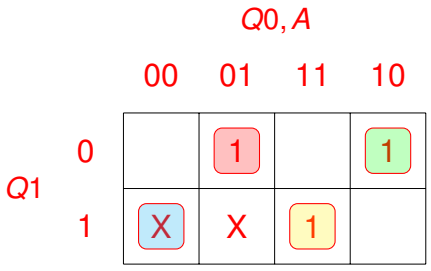
1. [10 points, Lecture 10] Let's have some FSM practice! The STD shown has three states, one input (**A**), and two outputs (**BC**).

Part A: Fully Encoded

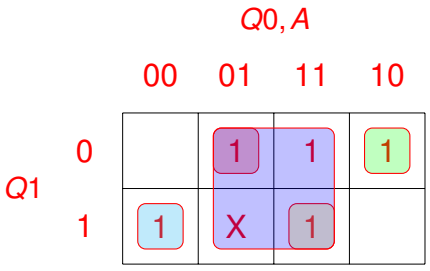
State assignments: **`State1 = 00`**, **`State2 = 01`**, **`State3 = 11`**. If you chose differently your answers will be different, though still possibly correct. The State Transition Table is:

State	Q1	Q0	A	N. State	D1	D0	B	C
State1	0	0	0	State2	0	1	0	1
State1	0	0	1	State3	1	1	0	1
State2	0	1	0	State3	1	1	1	0
State2	0	1	1	State2	0	1	1	0
State3	1	1	0	State1	0	0	0	1
State3	1	1	1	State3	1	1	0	1

It is fairly easy to see the output equations: $B = Q1 \oplus Q0$ and $C = \overline{B}$.



$D1 = Q0 \oplus Q1 \oplus A$



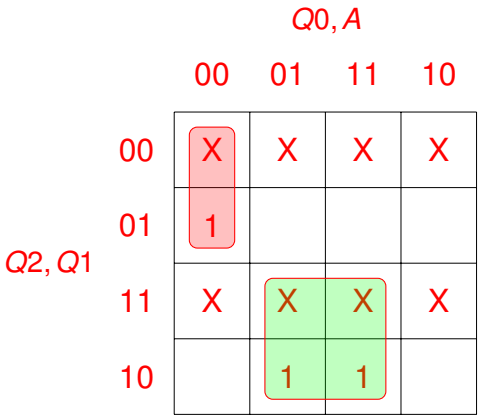
$D0 = D1 + A$

Part B: Output Encoded

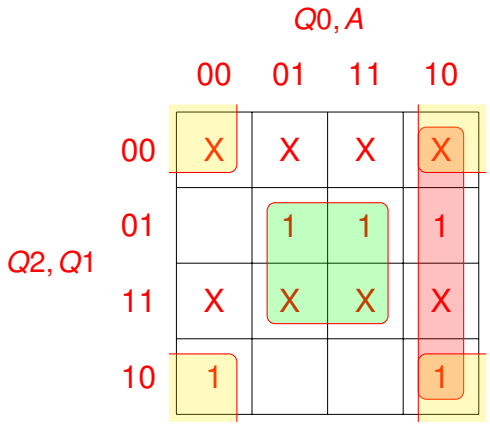
State assignments: **`State1 = 010`**, **`State2 = 10X`**, **`State3 = 011`**. Note that you need a third bit because the output in **`State1`** and **`State3`** is the same.

State	Q2	Q1	Q0	A	N. State	D2	D1	D0	B	C
State1	0	1	0	0	State2	1	0	X	0	1
State1	0	1	0	1	State3	0	1	1	0	1
State2	1	0	X	0	State3	0	1	1	1	0
State2	1	0	X	1	State2	1	0	X	1	0
State3	0	1	1	0	State1	0	1	0	0	1
State3	0	1	1	1	State3	0	1	1	0	1

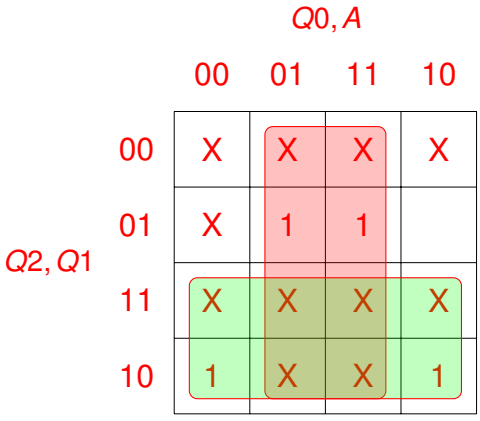
Again, the output equations are somewhat simple (that's the point of output encoding: $B = Q2$ and $C = Q1$.



$$D2 = \overline{Q2} \cdot \overline{Q0} \cdot \overline{A} + Q2 \cdot A$$



$$D1 = Q0 \cdot \overline{A} + Q1 \cdot A + \overline{Q1} \cdot \overline{A}$$



$$D0 = A + Q2$$

Part C: 1-Hot Encoded

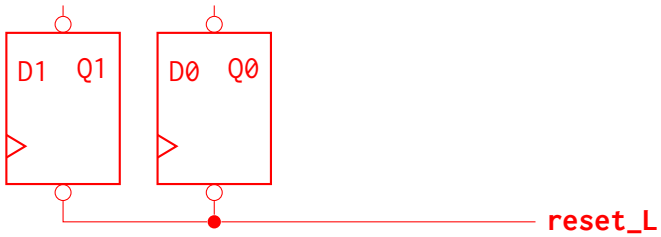
One F/F per state, which I will label **`State1_FF`**, **`State2_FF`** and **`State3_FF`**. The assignments are 1-Hot, so when in State1, **`{State3_FF, State2_FF, State1_FF}`** is **`001`**. State2 is **`010`** and State 3 is **`100`**.

We can observe the STD to formulate the equations. The only way to get to State1 is from State3 when $A = 0$. Therefore, the D input on **`State1_FF`** is $State3\_q \cdot \overline{A}$. Likewise, D for State2 is $State1\_q \cdot \overline{A} + State2\_q \cdot A$. State3 is $State1\_q \cdot A + State2\_q \cdot \overline{A} + State3\_q \cdot A$.

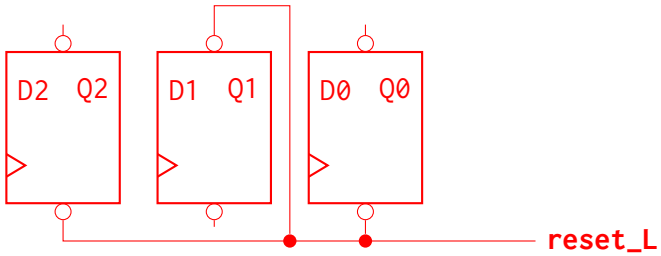
The output equations are similarly easy to read from the STD. $B = State2\_q$ and $C = State1\_q + State3\_q$.

Part D: Reset Connections

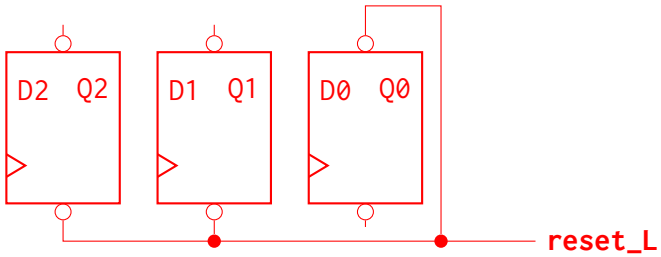
1A: Fully encoded



1B: Output encoded



1C: 1-Hot encoded



Part E: SV enums

enum logic [1:0] State1 = 2'b00, State2 = 2'b01, State3 = 2'b10 state, nextState;

enum logic [2:0] State1 = 2'b010, State2 = 2'b100, State3 = 2'b011 state, nextState;

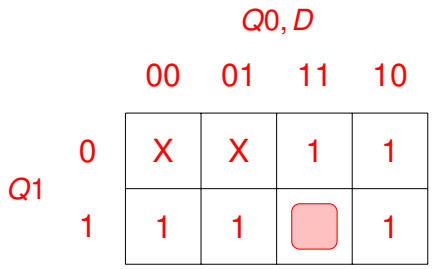
enum logic [2:0] State1 = 2'b001, State2 = 2'b010, State3 = 2'b100 state, nextState;

2. [5 points, Lecture 10] More FSM practice, this time a Mealy machine! The STD shown has three states, one input (**D**), and two outputs (**EF**).

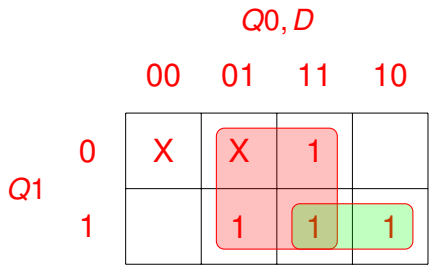
Part A: Fully Encoded

State assignments: **State1 = 01**, **State2 = 10**, **State3 = 11**. If you chose differently your answers will be different, though still possibly correct. The State Transition Table is:

State	Q1	Q0	D	N. State	D1	D0	E	F
State1	0	1	0	State2	1	0	0	1
State1	0	1	1	State3	1	1	1	1
State2	1	0	0	State2	1	0	1	1
State2	1	0	1	State3	1	1	1	0
State3	1	1	0	State3	1	1	0	0
State3	1	1	1	State1	0	1	1	0

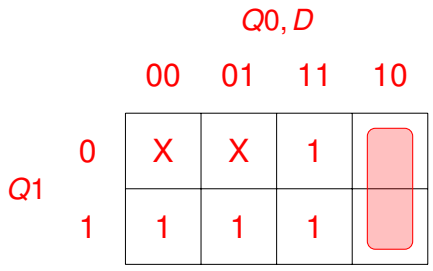


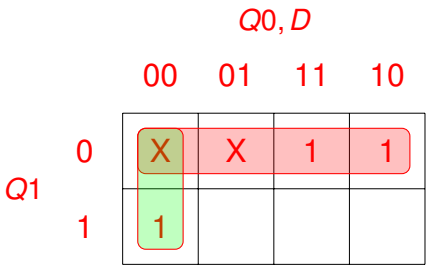
$D1 = \overline{Q1} + \overline{Q0} + \overline{D}$



$D0 = D + Q1 \cdot Q0$

The output equations aren't as obvious as in the Moore machines, so they will require some K-maps also.





$E = \overline{Q1} + D$

$F = \overline{Q1} + \overline{Q0} \cdot \overline{D}$

Part C: Output Encoded

Because this is a Mealy machine, whose output depends on the state as well as the inputs, we can't use an output-encoded style.

The output encoded style is defined as having a subset of the state bits that are also the FSM's outputs. All possible output patterns must be encoded in the state assignment. Thus there is no output logic (no logic between the state FFs and the outputs). However, since this is a Mealy machine, some of the outputs must also be a function of the inputs, requiring output logic.

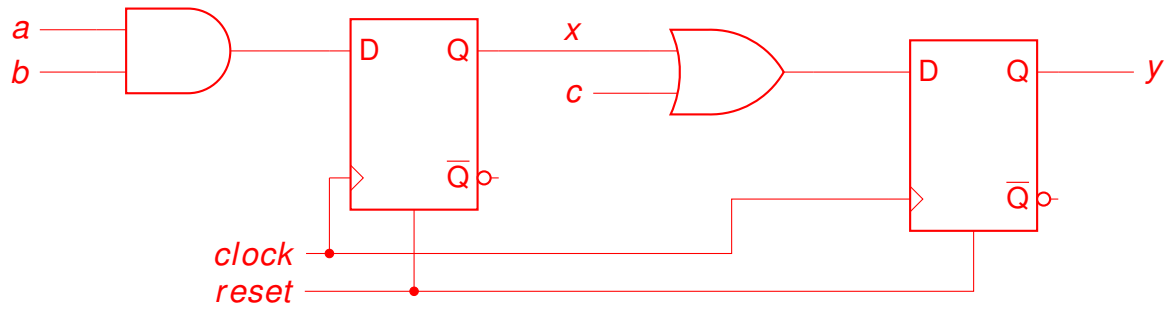
Can't do both.

3. [6 points, Lecture 10] Write a SystemVerilog model for the STD of question 1B. Your design should be an explicit style FSM, using the **`enum`** keyword.

   You can find **`hw5prob3.sv`** on Canvas.

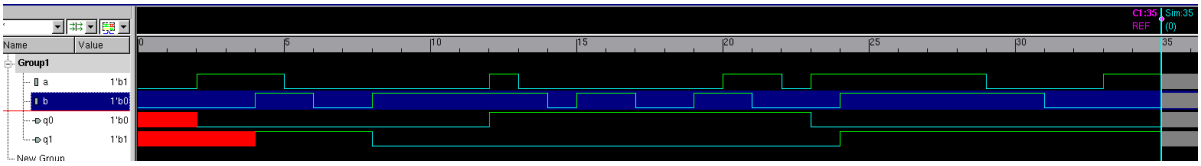
4. [6 points, Lecture 10] Draw a schematic of the hardware described by the following SystemVerilog module.

   The non-blocking assignment (i.e. <=) represents a value being stored in a flip-flop. In this case, two flip-flops, as `x` and `y` get stored in separate flip-flops.



5. [5 points, Lecture 10] In HW4, you determined the outputs of a FF when connected to the waveform below. Provide a screenshot of the VCS waveform viewer to confirm your answer. I have provided, in **`hw5prob5.sv`**, an **`initial`** block that generates **`A`** and **`B`** as shown (plus some timing information). Your task is to find the code for a D Flip-flop (hmm... I wonder where you might find that?) and modify **`hw5prob5.sv`** to include the FF and the connections from **`A`** and **`B`**. Then, use the waveform viewer to show **`Q`** in each of the connection situations.

   `hw5prob5_answer.sv` is posted to Canvas. Writing that was the easy part. The point of the problem was to ensure you understood how to use the waveform viewer. Here is my screenshot.

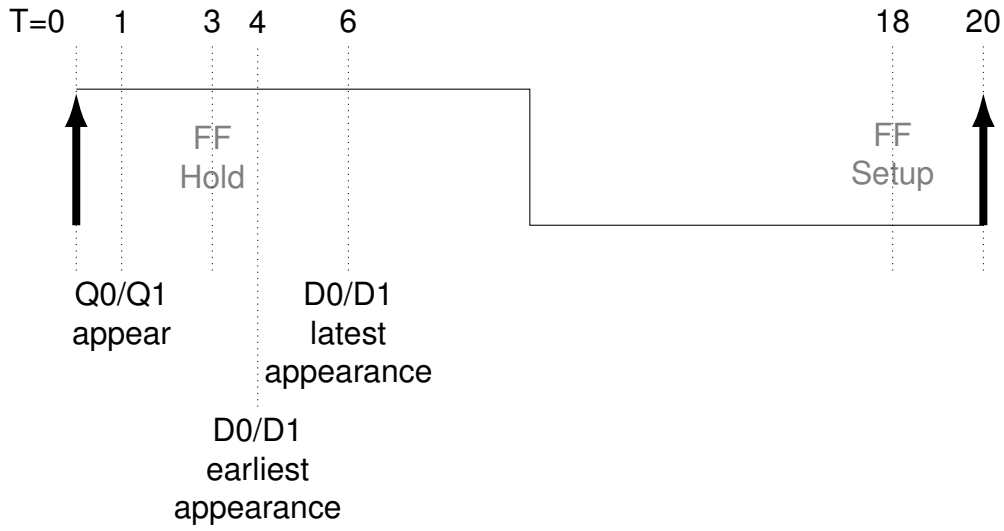


Non-Drill Problems [32 points]

6. [20 points, Lecture 11] Consider the simple synchronous design shown. It is an abstraction of every FSM design you've done, where some combinational circuitry looks at the state values held in the state register and calculates a new value to load into the state register at the next clock edge. All I've done here is to show a few of the timing parameters.

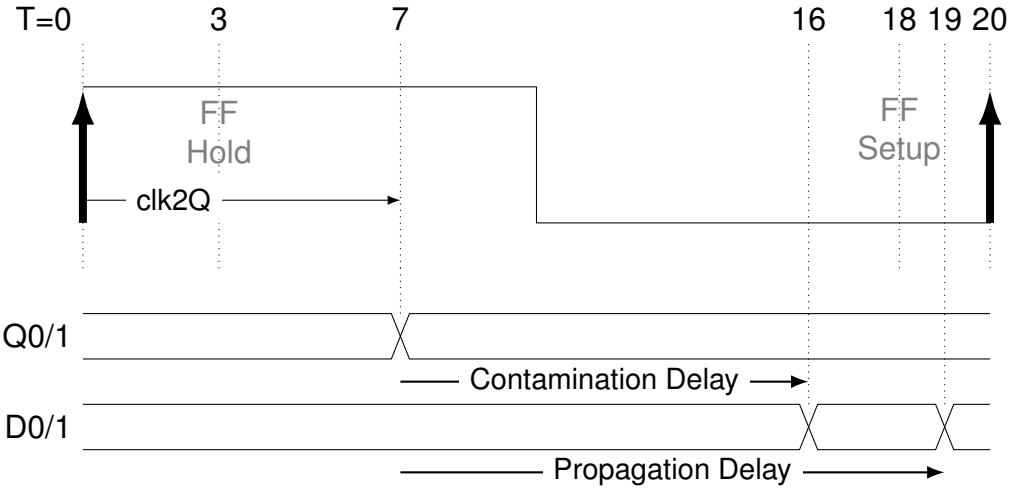
   In cases where $s \neq$ #0, you'll have to show a timing diagram for each flip-flop.

   (a) What happens when $c =$ #3, $p =$ #5, and $s =$ #0?



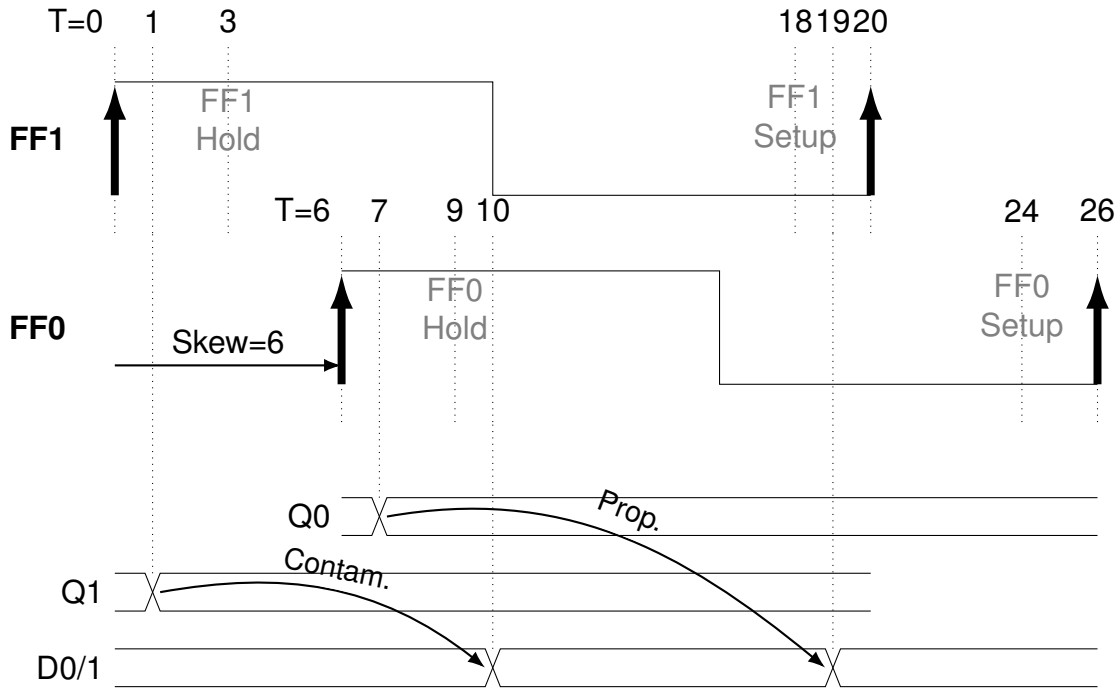
   No problem at all (except you're running the clock too slow).

   (b) What happens when **`clk2Q`** is #7, $c =$ #9, $p =$ #12, and $s =$ #0?



Here, the `clk2Q` is 7, so the `Q` output is stable after that time. Contamination delay #9 later (i.e., 7 + 9 = 16), `D1/2` will start to change to their new values. Propagation delay #12 later (i.e. 7 + 12 = 19), `D1/2` are guaranteed to be stable with the new values. Unfortunately, that is after the required setup time, so the circuit misses timing.

(c) What happens when $c = \#9$, $p = \#12$, and $s = \#6$?



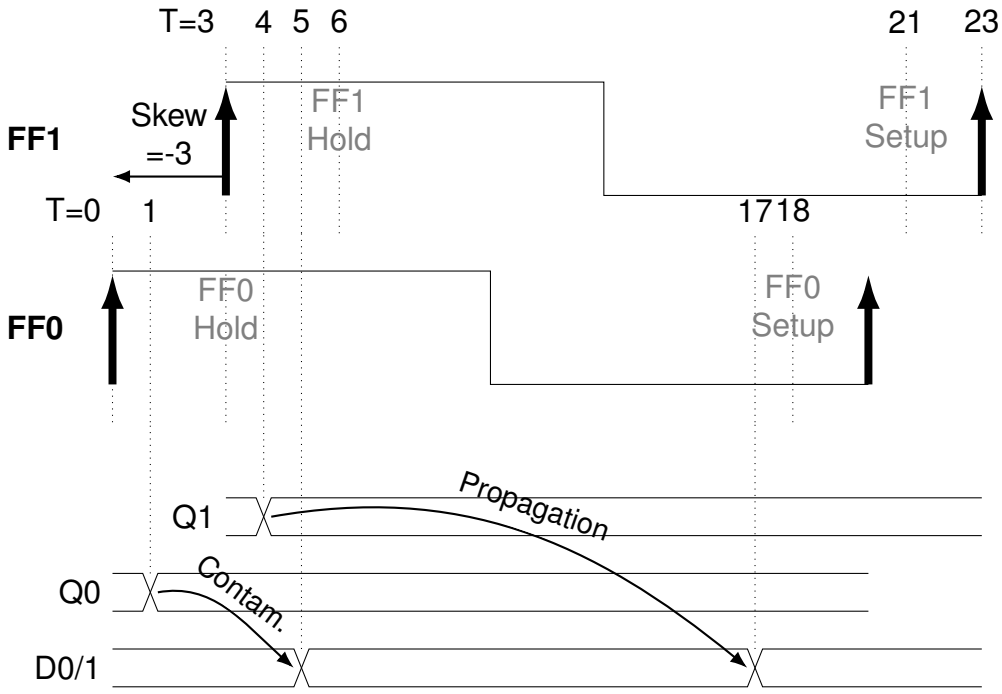
Setup violation on FF2!

Here's how it happened. Without any skew, `D1/2` would be stable after `clk2Q` + propagation delay (i.e., 1 + 12), which is before the setup time (18).

But, FF1 got delayed due to the skew. So, `D1/2` won't be stable until after FF1's `clk2Q` + propagation delay. That is, T6 + #13 or T=19. That's after FF2's setup time.

Note that in my drawing, The arrow from `Q2` to the first change in `D1/2` is due to the contamination delay — after this point, `D1/2` may start changing. The other arrow, from `Q1` to the last change of `D1/2` is due to the propagation delay. After this point, `D1/2` cannot change any more.

(d) What happens when $c = \#4$, $p = \#13$, and $s = -\#3$ (a negative skew means the clock gets to **`FF0`** before it gets to **`FF1`**).

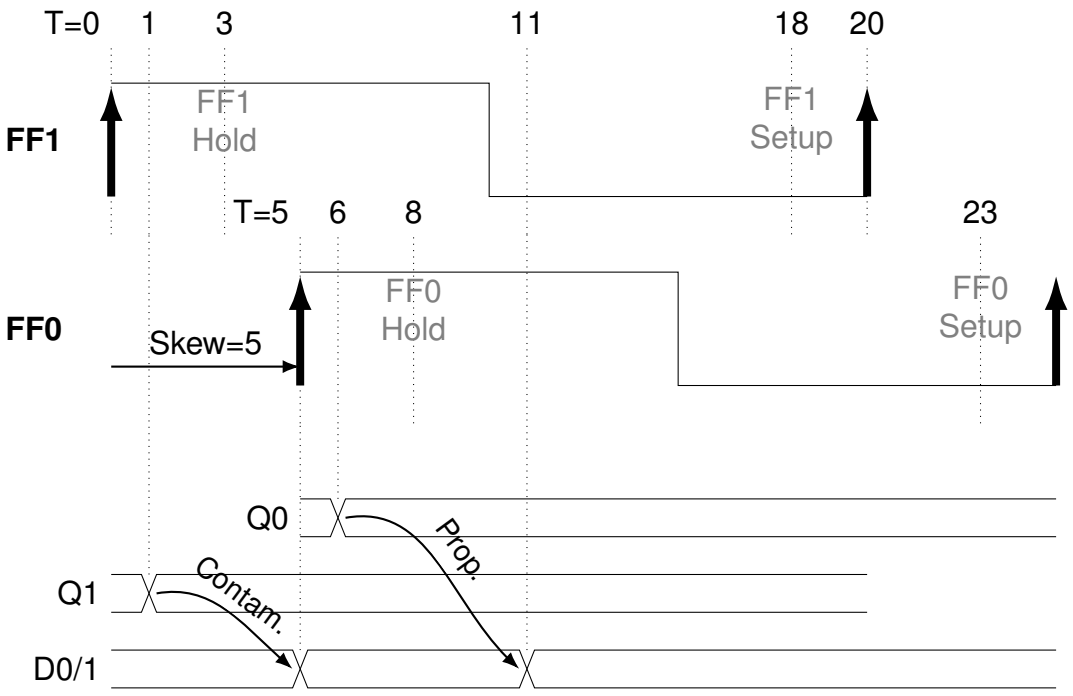


Hold time violation!

FF1's output changes at T=0 + **`clk2Q`** = 1. **`D1/2`** are contaminated #4 later (i.e. at T=5). **`Q2`** will be stable at T=3 + **`clk2Q`** = 4. Therefore **`D1/2`** will settle down and be stable #13 after this point (i.e. T=17). That means that the inputs will be stable before FF1's setup time, which is good.

Less good is that **`D1/2`** changed before FF2's hold time! Ooops.

(e) If $c = \#4$, $p = \#5$, and $s = \#5$, what is the smallest **T** can be?



This problem was supposed to be about the propagation delay, which causes changes to `D` to happen as late as T=11. That would mean we could crank the clock down until T=11 is just before FF2's setup time, so a clock period of > 13 is good.

However, note that the hold time is not met on FF1. It doesn't matter how small (or large) the clock period is, the circuit won't work because of this hold time violation.

7. [8 points, Lecture 10] For the circuit below, draw the STD. You don't know what the state assignment is, so give the states names based on the binary value **{Q1, Q0}** (i.e. **State00**, **State01**, etc)

For this problem, you need to run the FSM recipe backwards. You have a circuit, from which you can generate the next-state and output equations:

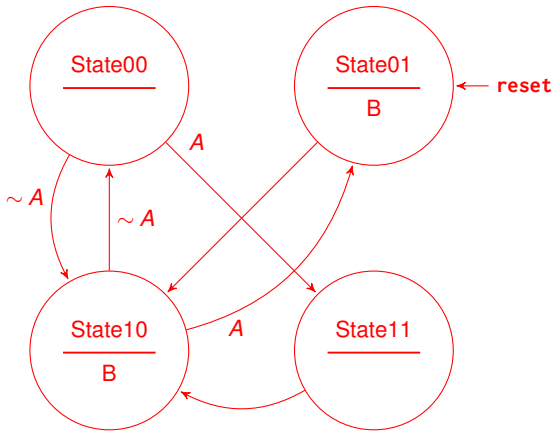
$$D1 = \overline{Q1} + Q0$$
$$D0 = A \cdot \overline{Q0}$$
$$B = Q1 \oplus Q0$$

From that you can generate the State Transition Table(s), this time with the state-encoded bits on the left:

Q1	Q0	A	D1	D0	B
0	0	0	1	0	0
0	0	1	1	1	0
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	0	0
1	1	1	1	0	0

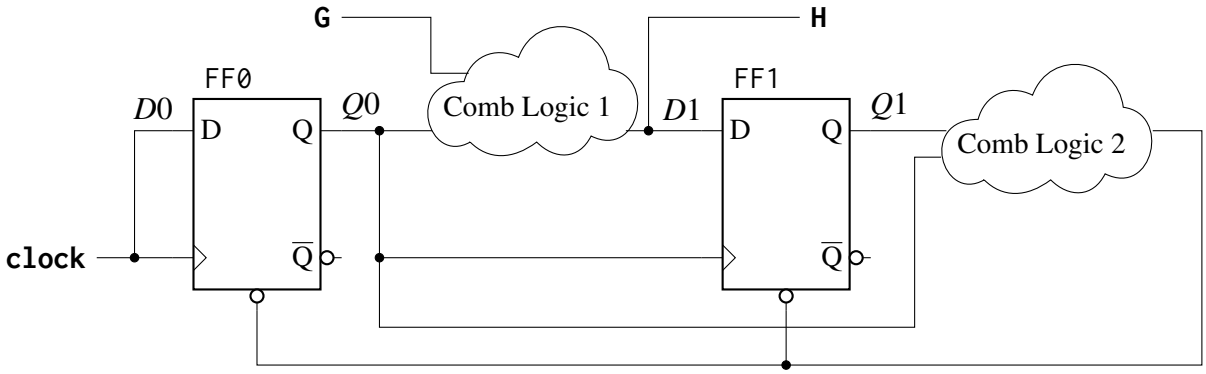
Current State	A	Next State	B
State00	0	State10	0
State00	1	State11	0
State01	0	State10	1
State01	1	State10	1
State10	0	State00	1
State10	1	State01	1
State11	0	State10	0
State11	1	State10	0

From this last table, we can easiy develop the STD, one state transition per line of the table. Check the output equation to see this is a Moore machine. Because the **reset** input is tied to the reset on FF1 and the preset on FF0, the reset state must be State01.



8. [4 points, Lecture 11] Your random lab partner has suggested this FSM design as the solution to the next lab. However, you've been faithfully attending every lecture and paying attention.[1] You know the design violates a lot of rules for synchronous design, and the professor would sacrifice you to the HH tower god if you tried this.

   Tell us exactly what is wrong with the design, based on the industrial strength rules from class. Be clear about each screw-up and what the problem is. Oh, you might be interested to know that input **G** is connected to a pushbutton. And that output **H** is connected to an ATM cash dispenser.



- Input **G** is asynchronous and needs to be synchronized.
- The **clock** and **D1** are tied together, which will means **D1** will change on the clock edge and lead to metastable situations.
- The **clock** of FF2 comes from a logic signal
- The **reset** of both FF1 and FF2 is driven by a logic signal

[1] work with me here.