18-240: Structure and Design of Digital Systems

Electrical & Computer ENGINEERING

HW5 [8 problems, 64 points]

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

Homework sets are due at 5:00PM on the due date. Upload your answers, to Gradescope by then. No late homework will be accepted. Remember, we let you drop two homework assignments over the semester.

By now, you know how to use **`handin240`** to create the files for the Gradescope upload. Refer to the course wiki or to previous homework if you need refresher instructions.

Discussions about homework in small groups are encouraged — think of this as giving hints, not solutions, to each other. However, homework must be written up individually (no copying is allowed). If you discussed your homework solutions with someone else, either as the giver or receiver of information, your write-up must explicitly identify the individuals and the manner information was shared.

You must show details of your work. There is no credit for just writing down an answer.

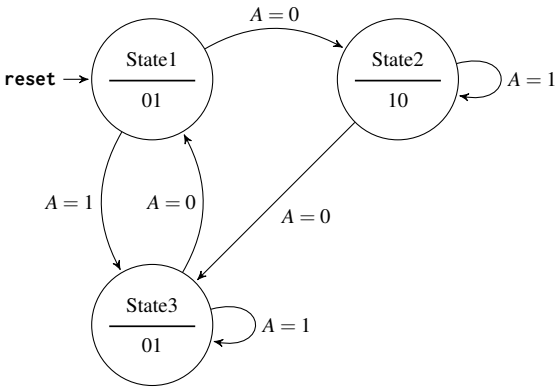
Drill Problems [32 points]

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

   (a) Specify a fully-encoded state assignment for this FSM. Write the output and next-state equations in reduced SOP style. Use don't cares if available.

   (b) Specify an output-encoded state assignment for this FSM. Write the output and next-state equations in reduced SOP style. Use don't cares if available.

   (c) Specify a 1-hot-encoded state assignment for this FSM. Write the output and next-state equations in reduced SOP style. Use don't cares if available.



   (d) Show how the reset line (called **reset_L**) is connected to the state F/Fs for each of the above situations.

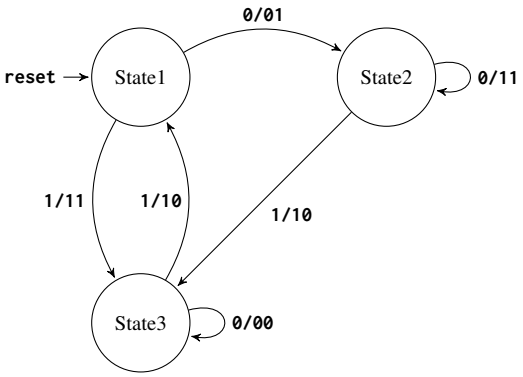
   (e) For each of parts a-c, write the SV **enum** code used to specify the state assignment.

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**).

   (a) Specify a full-encoded state assignment for this FSM. Write the output and next-state equations in reduced SOP style. Use don't cares if available.

   (b) Specify a 1-hot-encoded state assignment for this FSM. Write the output and next-state equations in reduced SOP style. Use don't cares if available.

   (c) Why can't you implement this FSM with an output-encoded style?



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.

   Then, write a testbench (an implicit-style FSM). Every legal state should be visited and every transition traversed. To show it working, have the testbench print out the current state along with the inputs and outputs. Your testbench module should be named properly (hint, hint: There's a rule about this in the course coding guidelines). Also, your testbench should have no inputs or outputs and should instantiate the FSM insude the testbench module.

   Use the following module declaration for the FSM.

```
module hw5prob3
  (input  logic A, clock, reset_L,
   output logic B, C);
```

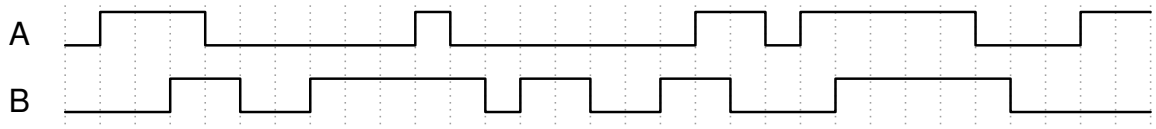
   ▷ **Submit your work as a file named `hw5prob3.sv`.**

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

```
module hw5prob4
  (input  logic clock, reset, a, b, c,
   output logic y);

  logic x;

  always_ff @(posedge clock, posedge reset)
    if (reset)
      {x, y} <= 2'b00;
   else
      begin
        x <= a & b;
        y <= x | c;
      end

endmodule : hw5prob4
```

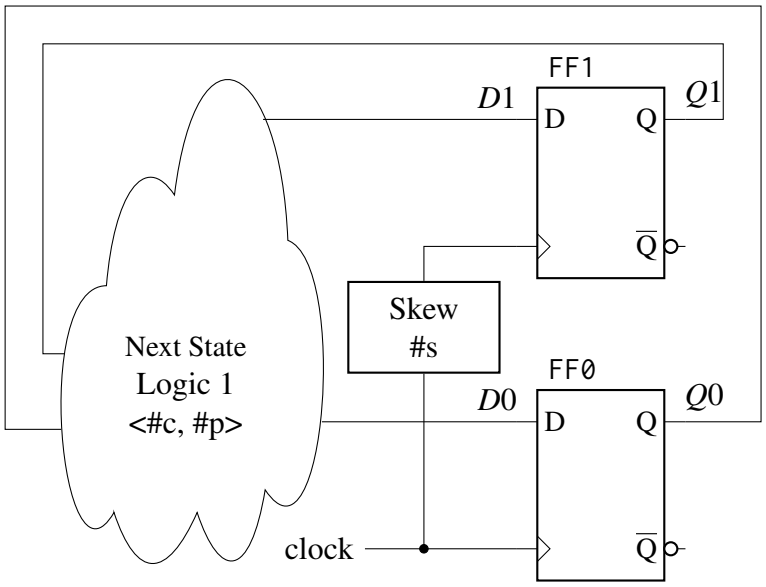
5. [5 points, Lecture 10] In HW4, you determined the outputs of a FF when connected to the waveform below. Provide a screenshot of the VCS waveform viewer to confirm your answer. I have provided, in **hw5prob5.sv**, an **initial** block that generates **A** and **B** as shown (plus some timing information). 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.

   ▷ **Submit one or two screenshots in your hw5.pdf file.**



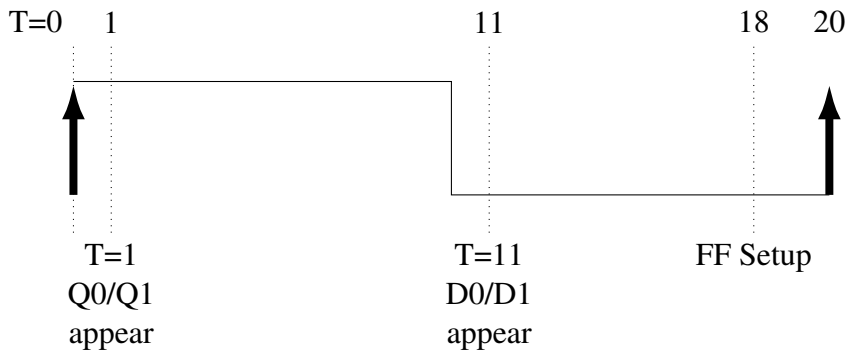
Non-Drill Problems [32 points]

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.



For each part of this problem, I will specify contamination delay values for the next state logic (#c), propagation delay (#p) and the amount of skew between the clocks (#s). A positive skew means **FF0** sees the clock later than **FF1**. Unless otherwise stated, the clock period T = #20 and both flip-flops have setup time of #2, hold time of #3 and clk2Q of #1.

In each part of this problem, I want you to draw a "timing diagram" that shows if the design's timing can work or not. I also want you to explain if the circuit works in each circumstance and if it does not, tell me why not. For the last part, use your diagram to answer the question. To be concrete about my expectations, imagine that $c = p = 10$ and $s = 0$. The diagram I'm looking for would look like this:



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?

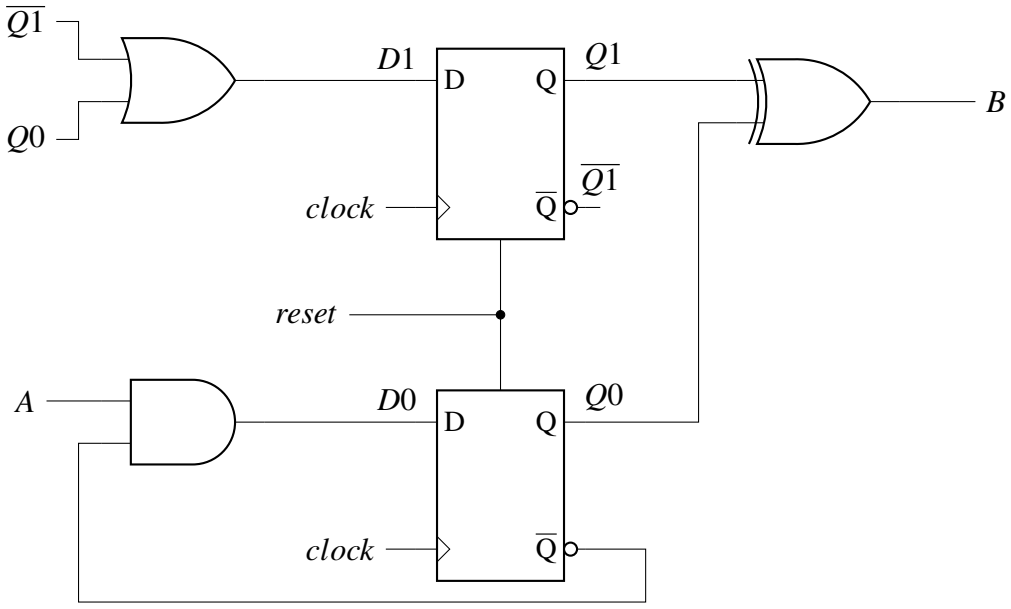
(b) What happens when **`clk2Q`** is #7, $c =$ #9, $p =$ #12, and $s =$ #0?

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

(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`**).

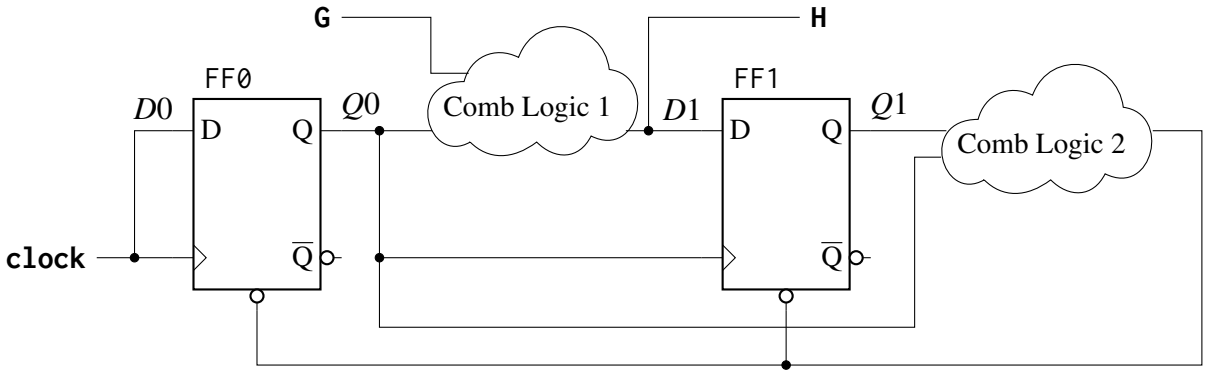
(e) If $c =$ #4, $p =$ #5, and $s =$ #5, what is the smallest **`T`** can be?

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)



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.



[1] work with me here.