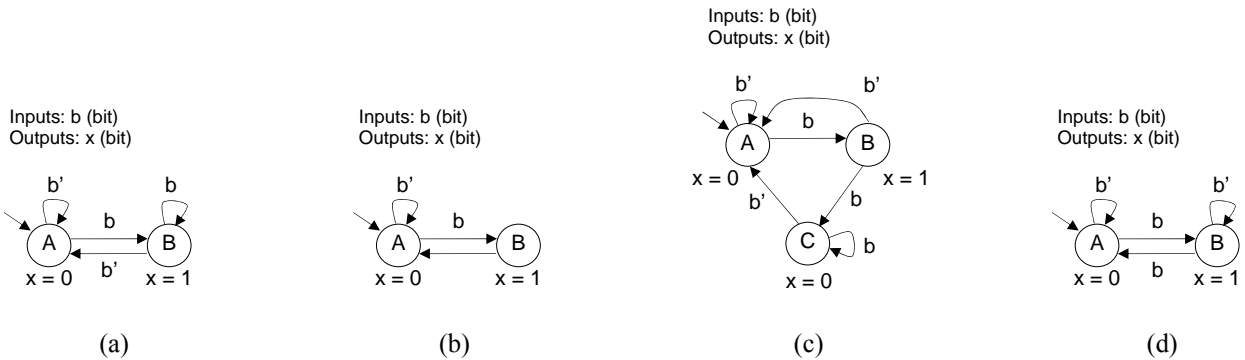
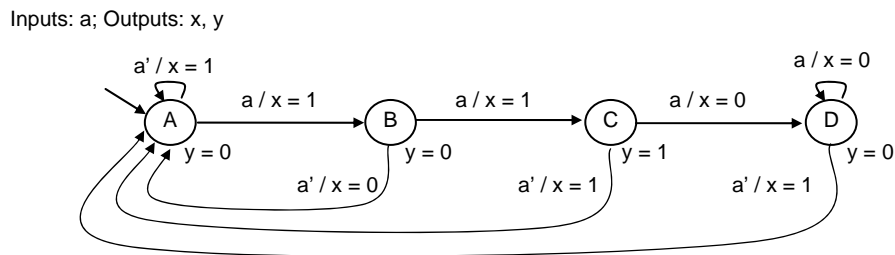


PRACTICE PROBLEMS 2
Lecture 3 - Lecture 4

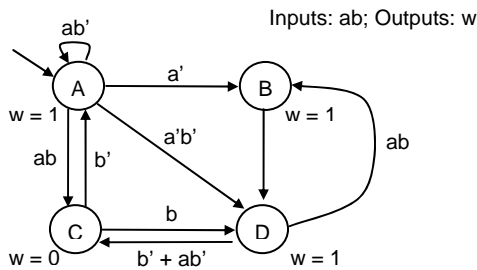
1. Which of the following FSMs implement a button synchronizer, converting each unique button press into a single cycle "1", regardless of the time the button is actually pressed.



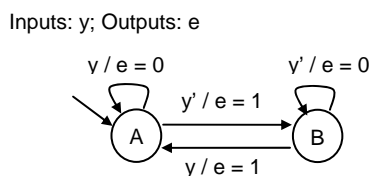
2. Create a state table for the following FSM. You can assume the state register is implemented with D flip-flops. You **DO NOT** need to implement the combinational logic.



3. Convert the following Moore FSM to a Mealy FSM.

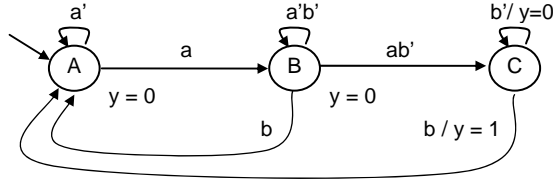


4. Convert the following Mealy FSM to a Moore FSM



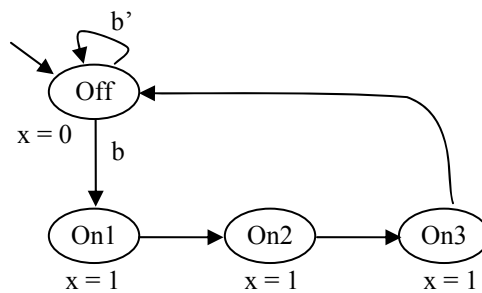
5. Implement a Mealy FSM that detects the input sequence pattern $z = 1, 0, 1, 0$. Whenever the input pattern is detected immediately output $f = 1$ (do not wait until the next clock cycle). You only need to show the FSM, you do not need to implement the architecture. *Hint: What happens when you detect input $z = 1, 0, 1, 0, 1, 0$?*
6. Describe the FSM provided using the formal specification $M = (X, Y, S, \delta, \lambda, s_0)$.

Inputs: a, b; Outputs: y



7. You want to design a laser surgery system where a surgeon activates the laser by pressing a button. The laser should then stay on for exactly 3 cycles, then turn off. The FSM below describes the three-cycles high laser timer controller. (*Same system from lecture 3, slide 4*)

Inputs: b (bit)
Outputs: x (bit)



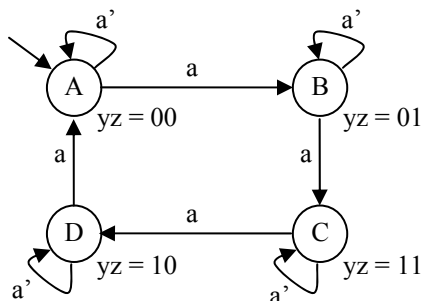
To model this FSM in Verilog we can use either a 2-process model or a 1-process model. Download the three-cycles high laser time code provided on the course page and simulate both FSMs using the Testbench provided.

- Do both the 1-process model and 2-process model accurately describe the functionality of the desired system?
- Is there any difference in output between the 1-process model and 2-process model? If so, why?

ECE 574A Questions

8. Derive a state table for the FSM provided. Assume the state register is implemented with T flip-flops and the state encodings for states A, B, C, and D are 00, 01, 10, and 11 respectively. A truth table for T flip-flops is provided.

Inputs: a (bit)
Outputs: y, z (bit)



T flip-flop and corresponding truth table

T	Q	Q _{next}
0	0	0
0	1	1
1	0	1
1	1	0