FSM Design – One more

That problem is a bit more complex than works well for a simple example, so let's go with:

> Design a state transition diagram (the lines and circles) which solves the following problem. There is one input, X, and one output, A. A is high if the last 3 inputs on X were 101.
>
> X: 11101011010001111010
>
> A: 00000101001000000001
>
> As before, the output is going high _after_ the input pattern occurs due to the delay inherent in a Moore machine.

State Transition Diagram:

(How many flip-flops are needed?)

State/Output Table:

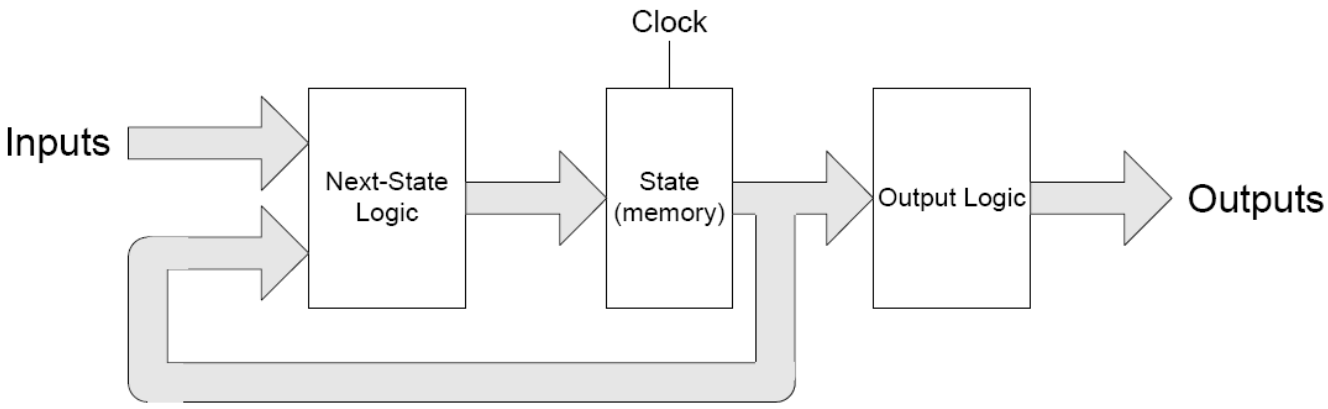
Current State	Input X=0	Input X=1	Output A

State assignment:

State	Q1	Q0
	0	0
	0	1
	1	0
	1	1

State table in a different format

Q1	Q0	X	D1	D0
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		



Q1	Q0	X	D1	D0
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

D1=_____

D0=_____

A=_____

Where do each of those things go in the above architecture?

Draw the circuit:

Yet another example (I doubt we'll do this in class, but it's here for practice)

A) Draw a state transition diagram for a state machine that takes two inputs "A" and "B"
   a. Output X is a 1 if, and only if, A>B thus far where A and B are treated as unsigned numbers, MSB first.
   b. Output Y is a 1 if, and only if, B>A thus far where A and B are treated as unsigned numbers, MSB first.

   So if A were "11000" and B were "10100" the output of X would be 01111 and B would be 00000.

B) Now design the gates and all the rest of the state machine!

More on Flip-Flops and Controllers: (section 3.5)

It is well worth reading this section of the book. It covers a large number of issues that might plague you in the future (I've hit pretty much all of them other than metastability). The problem is that 99% of the time these things don't come up, so you get burned by the 1% of the time they do because you aren't paying attention…

We'll come back later and discuss some of these issues later. For now we just want you aware that they exist...

Other flip-flop types:

- **SR Flip-Flop:** Like the SR latch but changes only on the rising edge. S=1 and R=1 is again not allowed.
- **JK Flip-Flop:** Like an SR flip-flop (J corresponds to set and K to reset) but S=1 and R=1 causes Q^*=!Q.
- **T Flip-Flop:** Toggle flip-flop with enable named "T". If T=0, Q^*=Q, if T=1 Q*=!Q

Set-up and hold

The basic idea is simple: you can't change the input (D) "near" the rising edge of the clock. The basic idea is that at best it becomes unclear if the new or old value of D should be used. In fact, as was covered before, it can cause a flip-flop to possibly oscillate or go metastable for a time. So there is some time before the *rising* edge of the clock (set-up time) during which the input must be held constant and some time *after* the rising edge of the clock (hold time) during which the input must be held constant.

Later we will see what impact set-up and hold time have on the design of sequential circuits. (Quite a lot!)

Synchronizers

Please take the time to read pages 132-134. It turns out that this is something embedded designers need to deal with quite often.

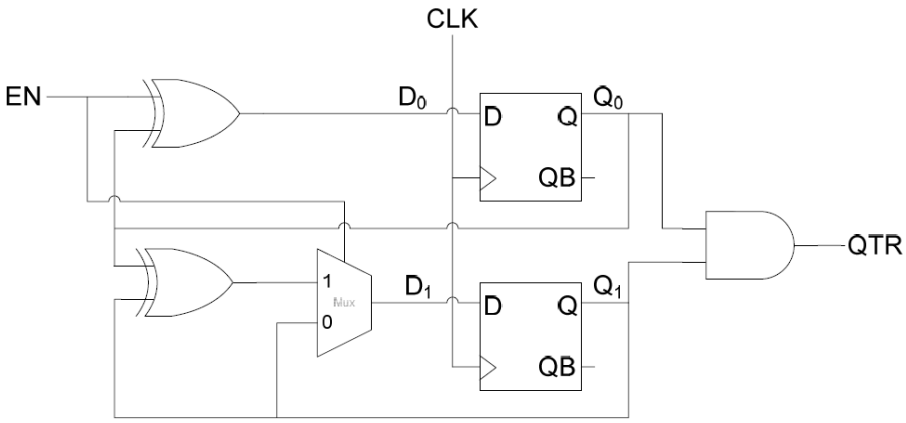
Initial state of an FSM

One key question is "how do I force the FSM into the initial state I want?" The most basic idea is to have a reset signal that goes high at the "start of time". But where does that reset signal come from? On occasion this can get a bit tricky.

Glitching

We've covered this a bit too. And this **will** bite you in lab 7 if you aren't careful.

Circuit analysis



Q1	Q0	EN	D1	D0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0		
1	0	1		
1	1	0		
1	1	1		

- Step 1: Write the **excitation equations**. That is, determine the value of the *inputs* to the flip-flops. Let's go ahead and convert to SoP form.
  - D_0 = _____
  - D_1 = $EN*(Q_0 \oplus Q_1)+ !EN \cdot Q_1$
- Step 2: Write the **transition equations.** That is, determine what the *outputs* of the flip-flops will be after the rising edge of the clock. For D flip-flops this is trivial.
- Step 3: Write the **output equations**. That is, determine what the value of the output (just QTR in this case) is.
  - QTR=_____

- Step 4: Create the **transition/output table**.

Current State		Input		Output
Q_1	Q_0	EN=0	EN=1	QTR
0	0	00	01	0
0	1	01	10	0
1	0			
1	1			

$Q_1^+Q_0^+$

- Step 5 and 6: Assign **state labels** and create **state/output table.**

Q_1	Q_0	State name
0	0	A
0	1	B
1	0	C
1	1	D

Current State	Input		Output
	EN=0	EN=1	QTR
A	A	B	0
B	B	C	0
C			
D			

Next State

- Step 7: Draw the **State Transition Diagram**.

Notice this whole process is ***very*** mechanical. We can, and do, get the computer to do this for us. That said, it's important to understand what the computer is doing. We work small problems, let the computer work the large ones.