Lecture #7: Intro to Synchronous Sequential State Machine Design

Paul Hartke

Phartke@stanford.edu

Stanford EE121

January 29, 2002

Administrivia

- Midterm #1 is next Tuesday (February 5th) in class.
  - Will not include state machines.
- Lab 3 Design Post-Mortem
  - Comments/Issues?
- Lab 4 handout
  - Due next week as normal.
- HW3 handout
  - Due Next Thursday February 7th
  - Read it over and I'll answer any questions on Thursday.

Two Types of Logic Circuits

- Combinational
  - A circuit whose outputs depend only on its current inputs
- Sequential
  - A circuit whose outputs depend not only on its current inputs, but also on the past sequence of inputs, possibly arbitrarily far back in time.

Readings

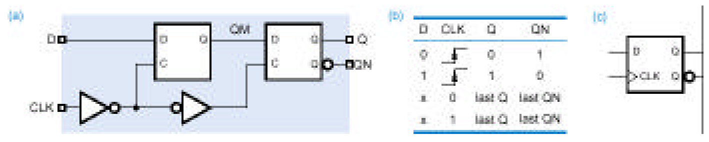
- In DDPP, Chapter 7 Intro, 7.3 (not 7.3.5), 7.4 (not 7.4.5), 7.5, 7.7
- We’ll do 7.8 next time

States and State Variables

- "The state of a sequential circuit is a collection of state variables whose values at any one time contain all the information about the past necessary to account for the circuit's future behavior."
- The states are normally encoded as binary numbers so for n state variables, there are 2^n possible states.
  - Since there is a finite number of states, these circuits are also called finite-state machines (FSM).

Basic Sequential Element

- Need an element that remembers: D Flip Flop (DFF)



D	CLK	Q	QN
0	↑	0	1
1	↑	1	0
x	0	last Q	last QN
x	1	last Q	last QN

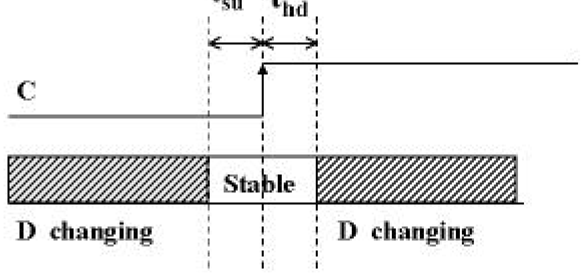
- Lots of way of building this element (or an analogous one)—we'll talk about ways later.

Simultaneous Input Changes

- Q: What if D and CLK change at the same time?
  - A: Bad things so do not change the input near the clock transition
- Setup Time: the amount of time the synchronous input (D) must be stable before the active edge of the clock.
- Hold Time: the amount of time the synchronous input (D) must be stable after the active edge of the clock.

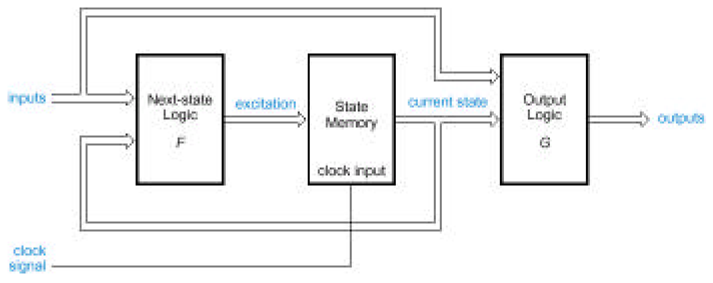
Setup and Hold Time Diagram

- If changes on D input violate either setup or hold time, then correct FF operation is ***not*** guaranteed.
- If they are violated, metastability results.



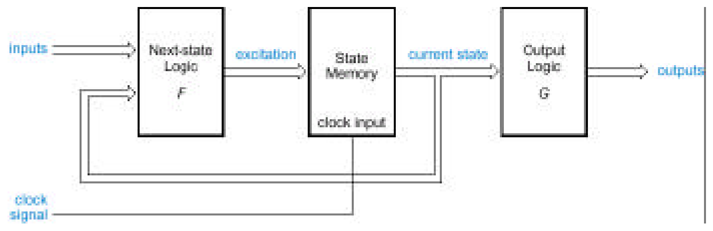
Mealy State Machine

- A Mealy state machine's output depends on both the state variables and the current input.



Moore State Machine

- A Moore state machines outputs only depend on the state variables.



Mealy vs. Moore

- Moore machine guarantees the outputs are steady for a full clock cycle.
- However, a change at the input takes at least one clock cycle to affect the output.
- Moore machine might require more states since not dependent on the input.
- Most of the time, I use a Moore machine.

State Machine Design Process

1. Determination of inputs and outputs.
2. Determination of machine states.
3. Create State/Bubble Diagram—should this be a Mealy or Moore machine?
4. State Assignment—assign each state a particular value.
5. Create Transition/Output Table
6. Derive Next State Logic for each state element—using K-maps as necessary.
7. Derive Output logic.
8. Implement in Xilinx.

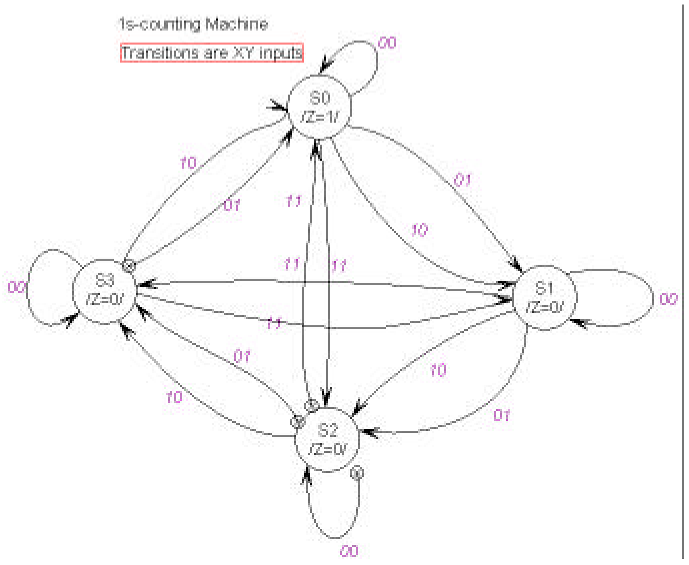
1’s Counting Machine

- Design a clocked synchronous state machine with two inputs, X and Y, and one output, Z. The output should be 1 if the number of 1 inputs on X and Y since reset is a multiple of 4, and 0 otherwise.
  - Section 7.4.6 in DDPP

Machine States

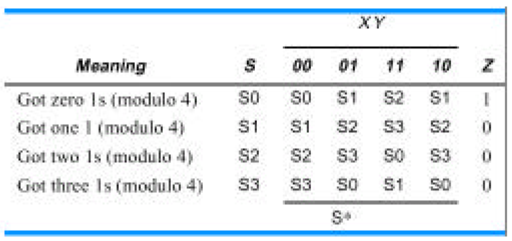
- S0 → Got zero 1s (modulo 4)
- S1 → Got one 1 (modulo 4)
- S2 → Got two 1s (modulo 4)
- S3 → Got three 1s (modulo 4)

Bubble Diagram



State and Output Table

- S* is the next state given the current state and the inputs.



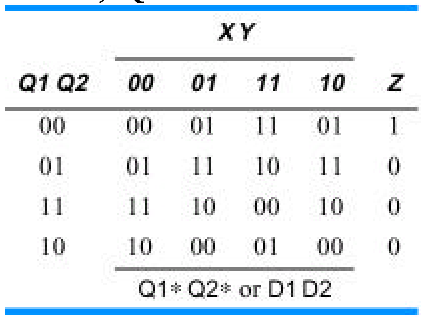
		XY				
Meaning	**S**	**00**	**01**	**11**	**10**	**Z**
Got zero 1s (modulo 4)	S0	S0	S1	S2	S1	1
Got one 1 (modulo 4)	S1	S1	S2	S3	S2	0
Got two 1s (modulo 4)	S2	S2	S3	S0	S3	0
Got three 1s (modulo 4)	S3	S3	S0	S1	S0	0
		S*				

State Assignment

- Use Binary Encoding but in K-map order.
  - Requires two state elements/DFFs.
  - Semi-Arbitrary decision.
- S0 → 00
- S1 → 01
- S2 → 11
- S3 → 10

Transition/Excitation Table

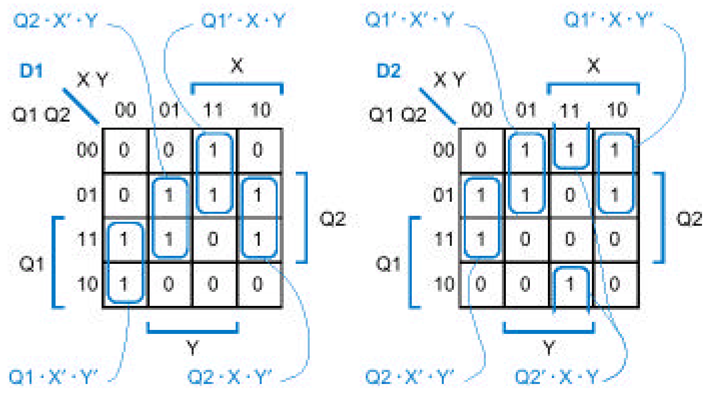
- Since we will only use D-flip flops, the transition and excitation tables are the same.
  - For a DFF, Q* = D



	XY				
Q1 Q2	00	01	11	10	Z
00	00	01	11	01	1
01	01	11	10	11	0
11	11	10	00	10	0
10	10	00	01	00	0
	Q1* Q2* or D1 D2				

Derive Next State Logic

- Use a K-Map for each state register input



Logic Equations

- AKA “Next-State” Logic
- D1 = Q2*X’*Y + Q1’*X*Y + Q1*X’*Y’ + Q2*X*Y’
- D2 = Q1’*X’*Y + Q1’*X*Y’ + Q2*X’*Y’ + Q2’*X*Y
- Z = Q1’*Q2’

One-Hot Encoding

- Alternative encoding of state variables.
- Use one state element for each state variable
- S0 → 0001
- S1 → 0010
- S2 → 0100
- S3 → 1000

One-Hot Transition Table

- Same as before…

Q1Q2Q3Q4	XY=00	XY=01	XY=11	XY=10	Z
0001	0001	0010	0100	0010	1
0010	0010	0100	1000	0100	0
0100	0100	1000	0001	1000	0
1000	1000	0001	0010	0001	0

Logic Equations

- D1 = Q1*X’*Y’ + Q2*X’*Y + Q2*X*Y’ + Q3*X*Y
- D2 = Q2*X’*Y’ + Q3*X’*Y + Q3*X*Y’ + Q4*X*Y
- D3 = Q3*X’*Y’ + Q4*X’*Y + Q4*X*Y’ + Q1*X*Y
- D4 = Q4*X’*Y’ + Q1*X’*Y + Q1*X*Y’ + Q2*X*Y
- Z = Q4

One-Hot Value?

- One-Hot decreases the decode depth required for next state logic at the expense of more DFFs and logic.
  - Decode depth of next state logic largely determines speed of state machine.
  - Why didn’t this help on this example?
- Lab 4 will have a state machine that should show the tradeoffs better.

Modified One-Hot

- In modified one-hot, the reset sequence is all zeros which transitions to the first state at the first clock edge.
- Needed for FPGAs whose FF’s powerup as zeros.

Don’t Forget ***Synchronous*** Resets!!

- The Xilinx FPGAs are designed so that on powerup, the DFFs initialize to logic 0.
  - We do not want to depend on that!!
- If your library supports it, use one that has a synchronous reset and tie it to the global reset pin.
- Else, explicitly design reset signal into your FSM.
- Can gate the reset signal to parts of the design on other events.