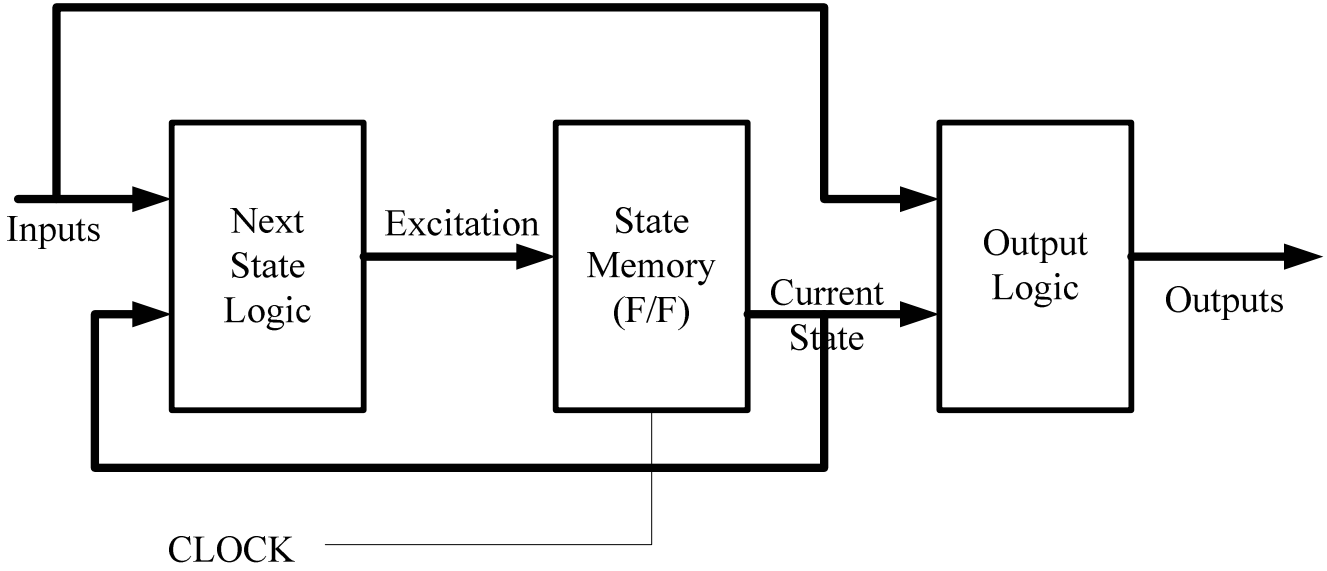
State Machine Design

State Machine Design

- State Machine types and some basics
- State Machine Design Process
- State Machine Design Examples
- State Machine Design in the HDL world

Types of state machines

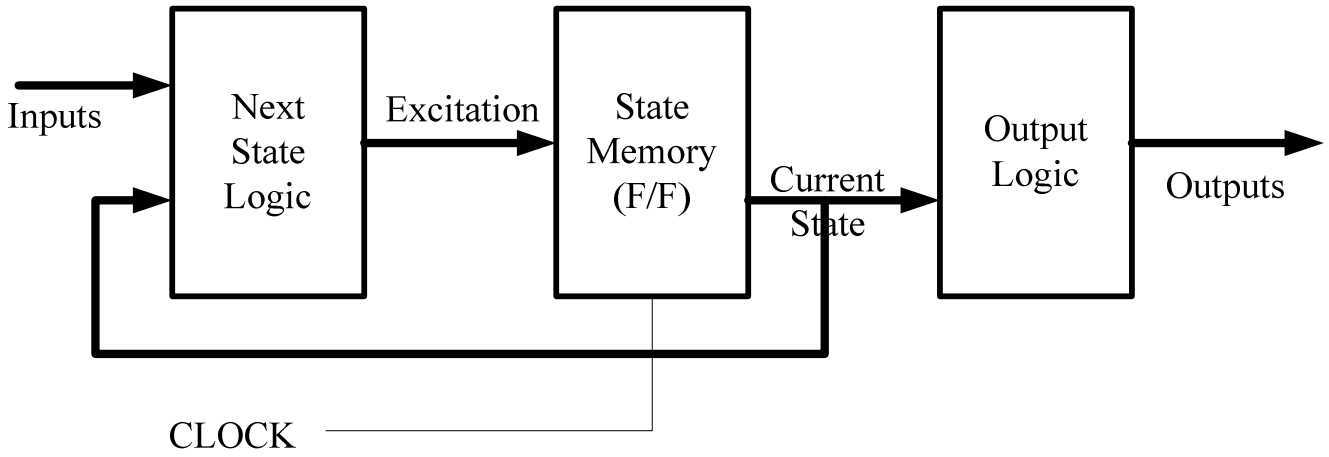
- Mealy Machine



- Characterized by – Outputs are a function of both inputs and current state

State Machine Types

- Moore machine



- Characterized by – Outputs are a function current state only

Mealy and Moore Implementaions

- Both Mealy and Moore machine implementation can be implemented with any sequential element.
- Why choose one elements over another?
  - Efficiency – The next state logic may differ significantly when using different F/F types.
  - Efficiency of implementation is also drastically affected by choice of state assignment

Characteristic Equations

- The *Characteristic Equation* formally specifies the flip-flop's next state as a function of its current state and inputs
- Q* means the next state value for the Q output of the F/F

Characteristic Equations

• S-R Latch	• $Q^* = S + R' Q$
• D Latch	• $Q^* = D$
• D F/F	• $Q^* = D$
• D F/F with Enable	• $Q^* = EN\ D + EN'\ Q$
• J-K F/F	• $Q^* = J\ Q' + K'\ Q$
• T F/F	• $Q^* = Q'$

Designing a Synchronous System

- Steps for designing a clocked synchronous state machine starting from a word description or specification
- First understand the description or specification and resolve any questions

- Step 1: Construct a state/output table corresponding to the description/spec. (Or create a state diagram)

Example

- Description
  - Design a clocked synchronous state machine with two inputs A and B, and a single output Z that is 1 if:
    - A had the same value at each of the two previous clocks
    - *Or*
    - B has been 1 since the last time that the first condition was true
  - Otherwise the output is 0

Evolution of a state table

- Figures 7-46 and 7-47 of text
- Set up table having columns for the relevant info. As we have 2 inputs need the 4 choices for inputs.

		AB				
Meaning	S	00	01	11	10	Z
Initial State	INIT	next state				0

First input

- What happens when first input arrives
- A0 – have one 0 on A
- A1 – have one 1 on A

(b)

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0					0
Got a 1 on A	A1					0

S*

Second Input

- Now what happens when in state A0? May have a value of 0 or 1 on the next A input. New state OK

(c)

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK	OK	A1	A1	0
Got a 1 on A	A1					0
Got two equal A inputs	OK					1

- OK says have 2 of the same on A

Second input (cont)

- Now if you are in state A1 what happens at next input?

(d)

Meaning	S	A B 00	01	11	10	z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK	OK	A1	A1	0
Got a 1 on A	A1	A0	A0	OK	OK	0
Got two equal A inputs	OK					1
		S*				

The next input

- Now resolve state OK

(a)

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK	OK	A1	A1	0
Got a 1 on A	A1	A0	A0	OK	OK	0
Got two equal A inputs	OK	?	OK	OK	?	1
		S*				

- May have to split state OK

Next input (1)

- Refine state OK to indicate if A is 0s or 1s

(b)

Meaning	S	A B: 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0					1
Two equal, A=1 last	OK1					1
		S*				

Refined state OK

- Two 0s on the A input

(c)

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1					1
		S*				

Refined state OK (2)

- Fill in state OK1

(d)

Meaning	S	A B: 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1	A0	OK0	OK1	OK1	1
		S*				

Next step

- Step 2 - Minimize the number of states – called *state minimization*
  - This step was a major part of state machine design.
  - With current HDL synthesis tools no so much so today
- Step 3 – Choose a set of state variables and assign state-variable combinations to named states.

The final steps

- Step 5 – choose a F/F type – today almost always a D type F/F
- Step 6 – Construct an excitation table
- Step 7 – Derive excitation equations from the table.
- Step 8 – Derive output equations from the table
- Step 9 – Draw a logic diagram

Example of finishing design

- State and output table to be implemented

Table 7-5
State and output table for example problem.

	A B				
S	00	01	11	10	Z
INIT	A0	A0	A1	A1	0
A0	OK0	OK0	A1	A1	0
A1	A0	A0	OK1	OK1	0
OK0	OK0	OK0	OK1	A1	1
OK1	A0	OK0	OK1	OK1	1
	S*				

Implement with D F/F

- Assign coding to state
- Why are 3 F/F used?

Table 7-8
Excitation and output table for Table 7-7 using D flip-flops.

	A B				
Q1 Q2 Q3	00	01	11	10	Z
000	100	100	101	101	0
100	110	110	101	101	0
101	100	100	111	111	0
110	110	110	111	101	1
111	100	110	111	111	1
	D1 D2 D3				

Develop excitation equations

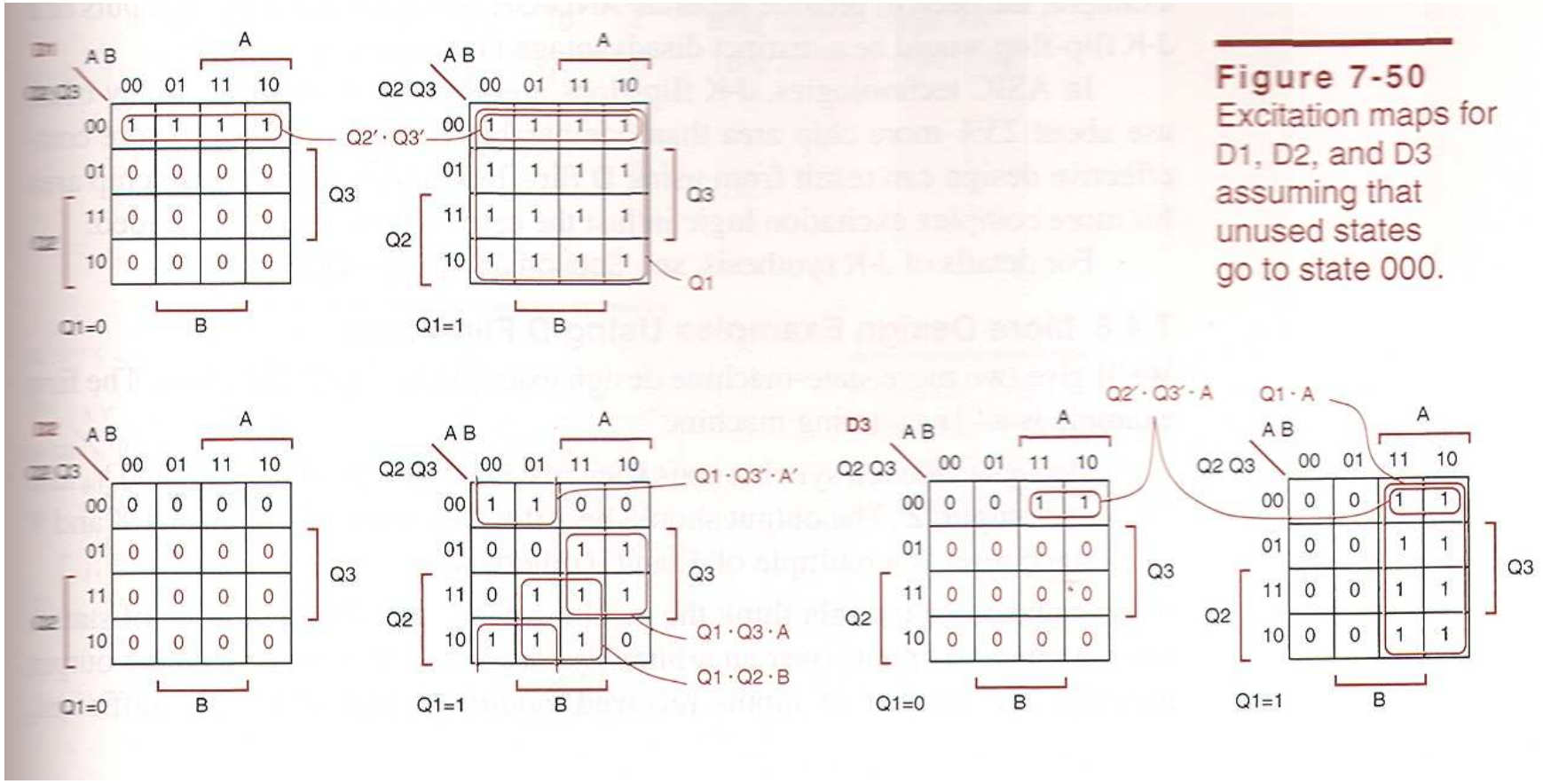


Figure 7-50
Excitation maps for D1, D2, and D3 assuming that unused states go to state 000.

A note on maps

- These are 5 variable maps
- Each is a function of 5 variables – input A, input B, and the 3 F/F outputs Q1,Q2,Q3
- End up with
  - D1 = Q1 + Q2’ Q3
  - D2 = Q1 Q3’ A + Q1 Q3 A + Q1 Q2 B
  - D3 = Q1 A + Q2’ Q3’ A
  - Z = Q1 Q2 Q3’ + Q1 Q2 Q3 = Q1 Q2

Assignment

- Prob 7.46 from text – Due Wednesday Oct 8

7.46 Design a clocked synchronous state machine with the state/output table shown in Table X7.46, using D flip-flops. Use two state variables, Q1 Q2, with the state assignment A = 00, B = 01, C = 11, D = 10.

Table X7.46

	X		
S	0	1	Z
A	B	D	0
B	C	B	0
C	B	A	1
D	B	C	0
	S*		