




CSE140: Components and Design Techniques for Digital Systems

Tajana Simunic Rosing

Today

- HW#6 due, HW#7 out
- Midterm #2 on Thursday
 - Chapters 1-8, App. A-C
 - Ok to bring one 8 ½" x 11" handwritten sheet of notes, pencil and eraser; nothing else
- Design examples



CSE140: Components and Design Techniques for Digital Systems

Design examples

Tajana Simunic Rosing

FSM design example: counter

- Design a three bit up/down counter with two inputs: count enable (C, where $C=1$ starts counting) and count direction (D). When $D=1$ count up, else count down in increments of two.

Analysis example

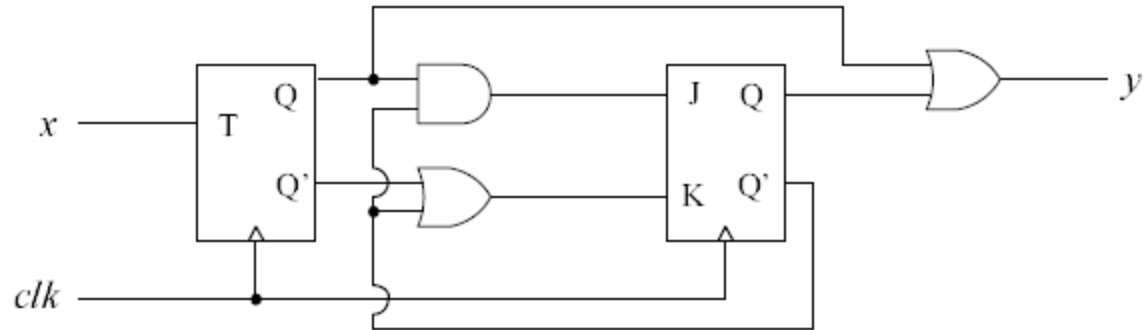
- Write the state table and next state equations for the following:

$$T = x$$

$$J = Q_1'(t)Q_0(t)$$

$$K = Q_1'(t) + Q_0'(t)$$

$$y(t) = Q_0(t) + Q_1(t)$$

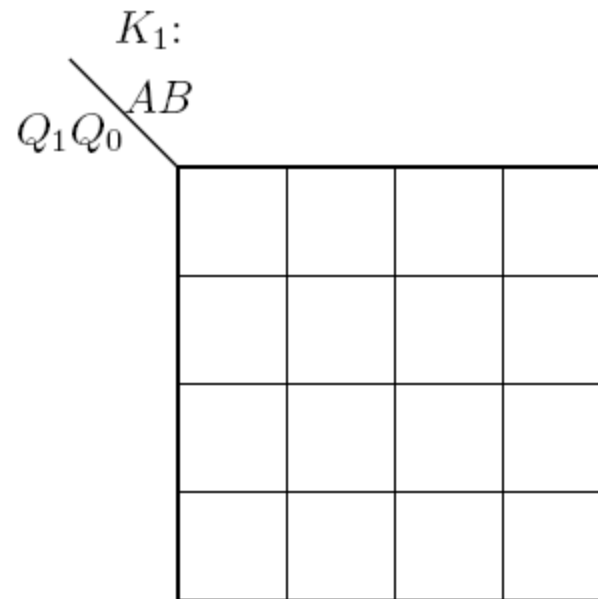
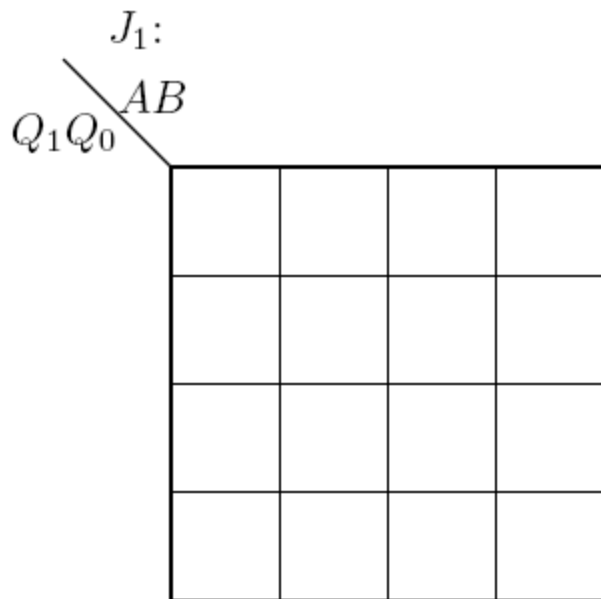


$x(t)$	$Q_0(t)$	$Q_1(t)$	J	K	$Q_0(t+1)$	$Q_1(t+1)$	$y(t)$
0	0	0	0	1	0	0	0
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1	-	-	-	-	-

Finite string pattern recognizer: Next state & Output logic

Present State Q_1Q_0	Next State			
	$AB = 00$	$AB = 01$	$AB = 10$	$AB = 11$
00	00	10	10	01
01	00	01	10	01
10	00	11	10	01
11	11	11	10	01

Q	$Q_{(next)}$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0



FSM Design Example

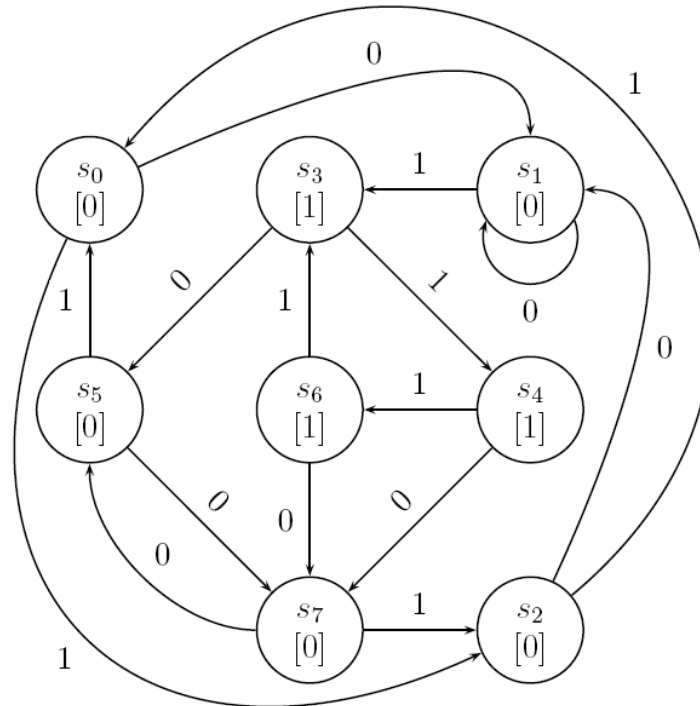
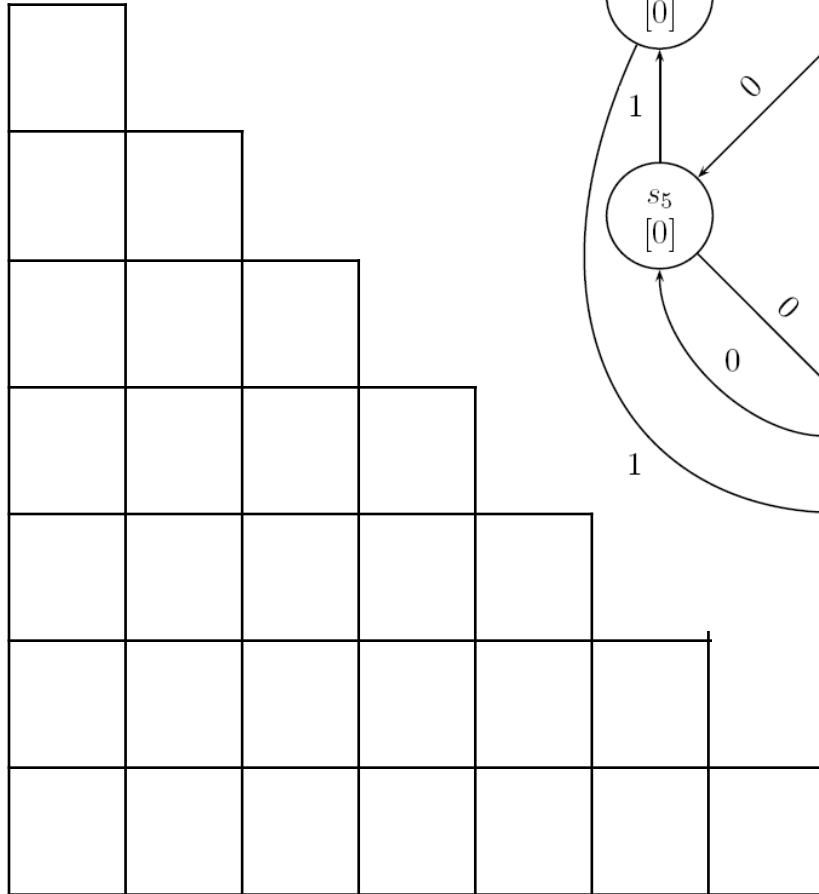
- Write the state table and implement the following state machine with T-FFs:

$$Q_0(t+1) = Q_0(t) + Q_1'(t) x(t),$$

$$Q_1(t+1) = Q_0' Q_1(t) + x'(t),$$

$$y(t) = Q_0(t) Q_1(t).$$

Example of state minimization

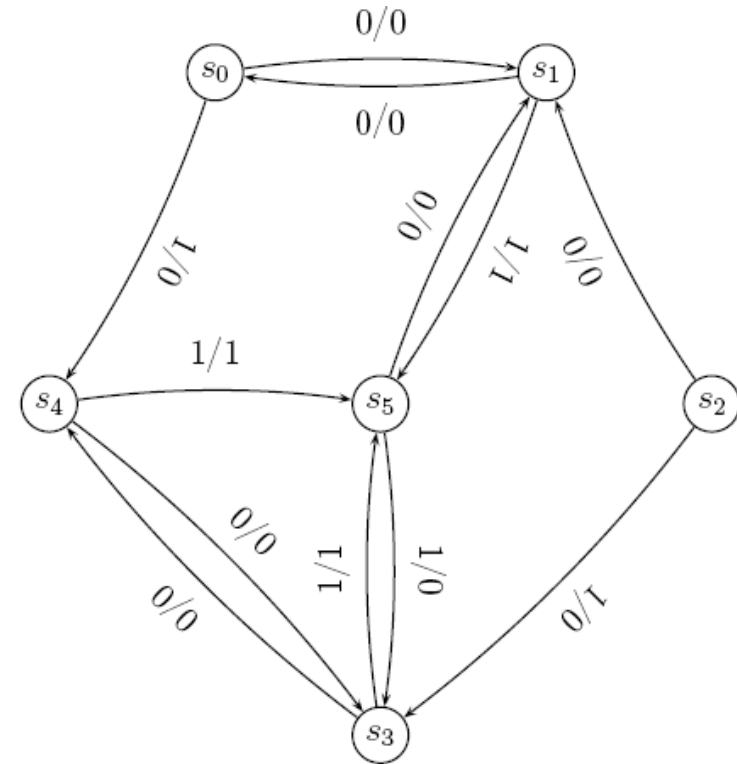


CS	NS x=0	NS x=1	Output
0			
1			
2			
3			
4			
5			
6			
7			

Example: state assignment


State Name	Assignment		
	Q_2	Q_1	Q_0
s_0			
s_1			
s_2			
s_3			
s_4			
s_5			

CS	NS $x=0$	NS $x=1$
0		
1		
2		
3		
4		
5		



Summary

- FSM design
 - Understanding the problem
 - Generating state diagram
 - Create state table
 - Minimize state table
 - State assignment
 - Select FF type
 - Minimize logic implementation
 - Draw logic diagram
 - Analyze timing



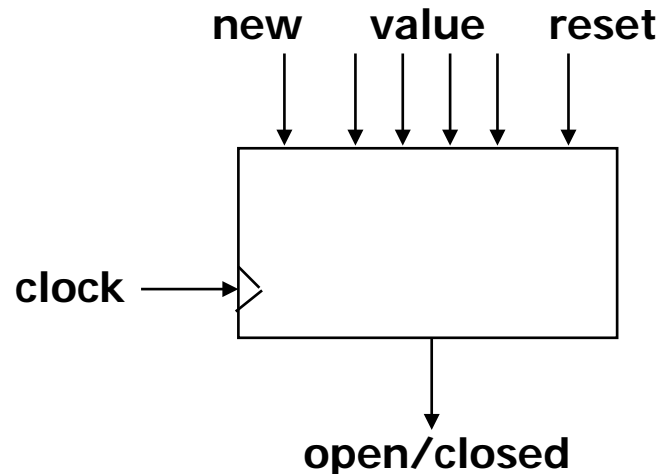
CSE140: Components and Design Techniques for Digital Systems

FSMs and implementations

Tajana Simunic Rosing

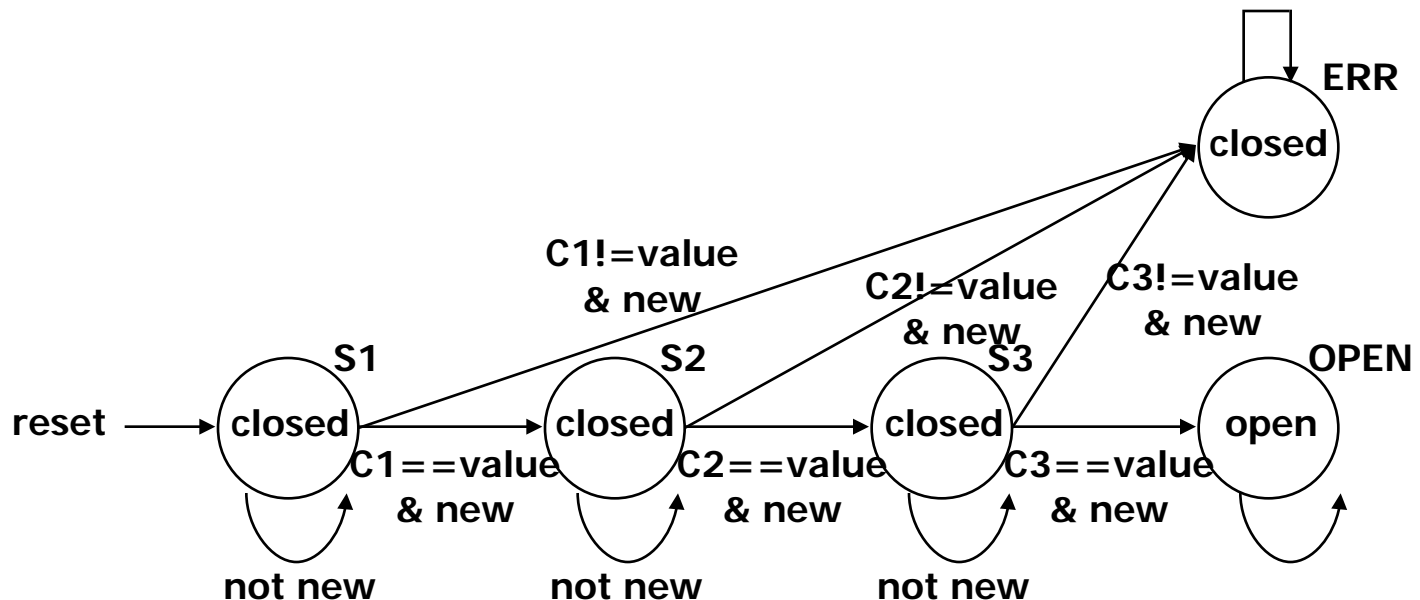
Digital combinational lock

- Door combination lock:
 - Press reset, then punch in 3 4 bit digits followed by enter, if the values match the code, the door opens; if there is an error the lock must be reset; once the door opens the lock must be reset
 - inputs: sequence of input values, reset
 - outputs: door open/close
 - memory: must remember combination or always have it available



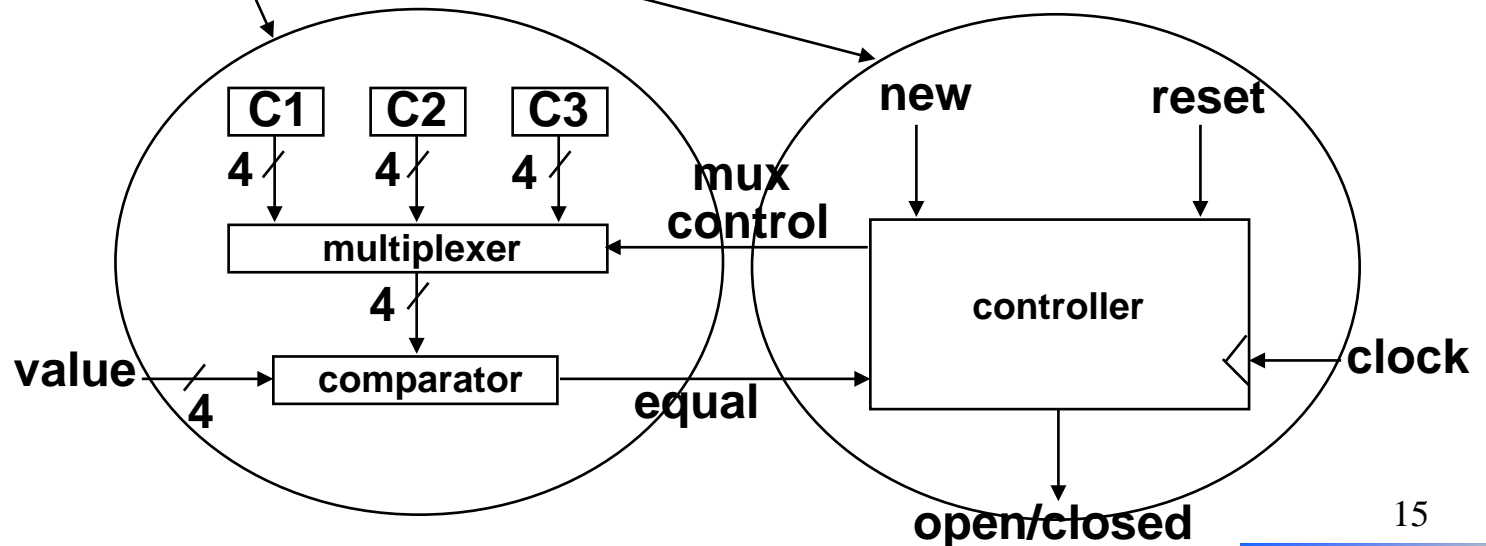
Digital combination lock state diagram

- States: 5 states
 - represent point in execution of machine
 - each state has outputs
- Inputs: reset, new, results of comparisons
- Output: open/closed



Data-path and control structure

- Data-path
 - storage registers for combination values
 - multiplexer
 - comparator
- Control
 - finite-state machine controller
 - control for data-path (which value to compare)



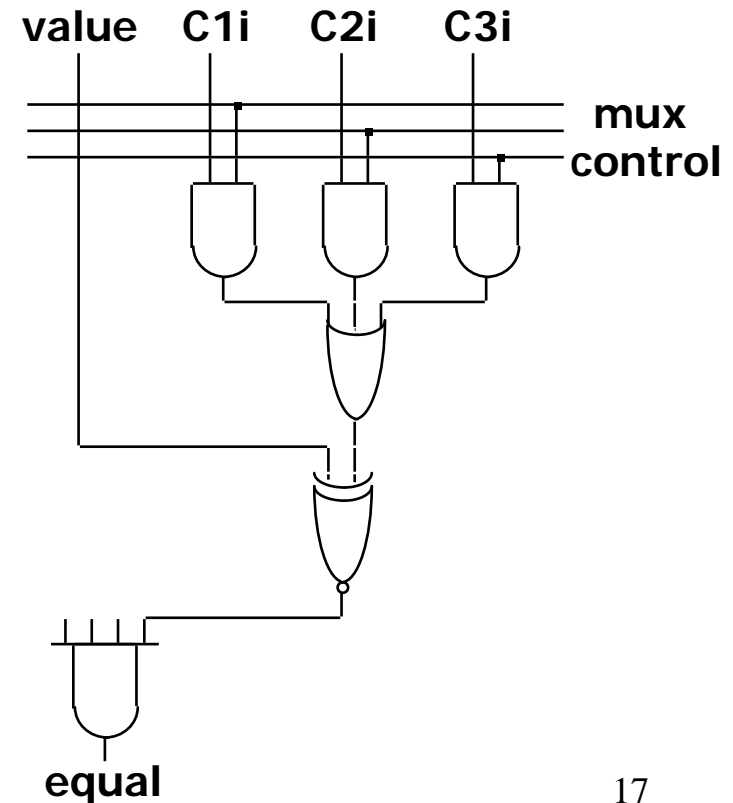
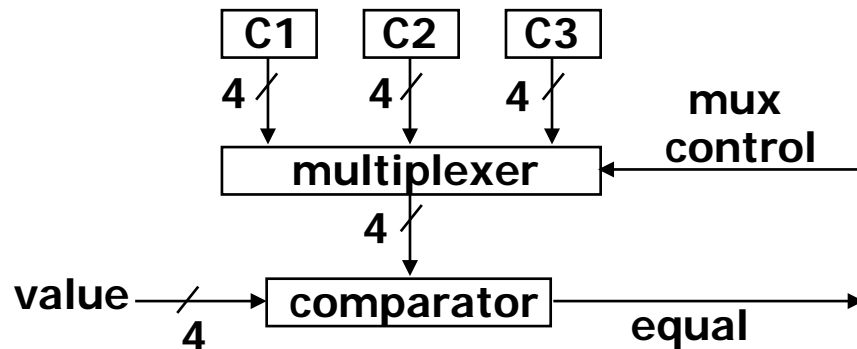
State table for a combination lock

- Finite-state machine
 - refine state diagram to take internal structure into account
 - state table ready for encoding

reset	new	equal	state	next state	mux	open/closed
1	–	–	–	S1	C1	closed
0	0	–	S1	S1	C1	closed
0	1	0	S1	ERR	–	closed
0	1	1	S1	S2	C2	closed
...						
0	1	1	S3	OPEN	–	open
...						

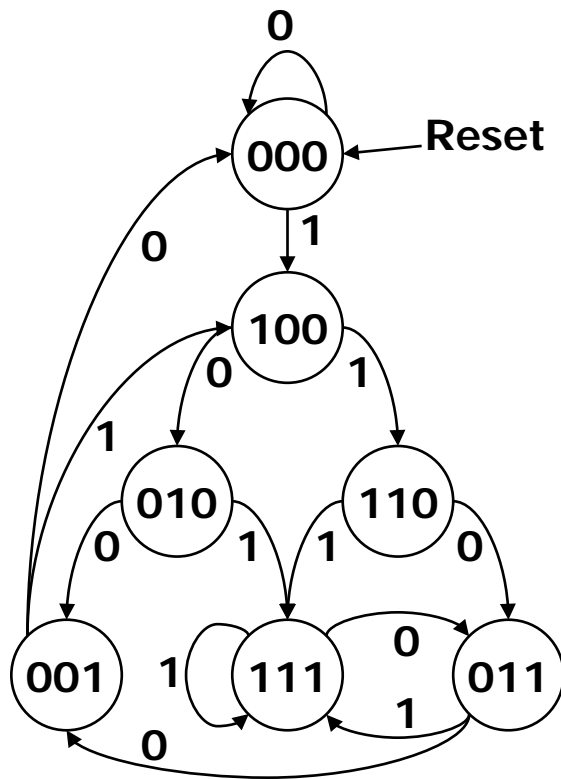
Datapath part of combination lock

- Multiplexer
 - easy to implement as combinational logic with few inputs



Median filter FSM

- Remove single 0s between two 1s (output = NS3)



I	PS1	PS2	PS3	NS1	NS2	NS3
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	X	X	X
0	1	1	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	0
1	1	0	1	X	X	X
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Median filter FSM

- Realized using the standard procedure and individual FFs and gates

I	PS1	PS2	PS3	NS1	NS2	NS3
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	X	X	X
0	1	1	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	0
1	1	0	1	X	X	X
1	1	1	0	1	1	1
1	1	1	1	1	1	1

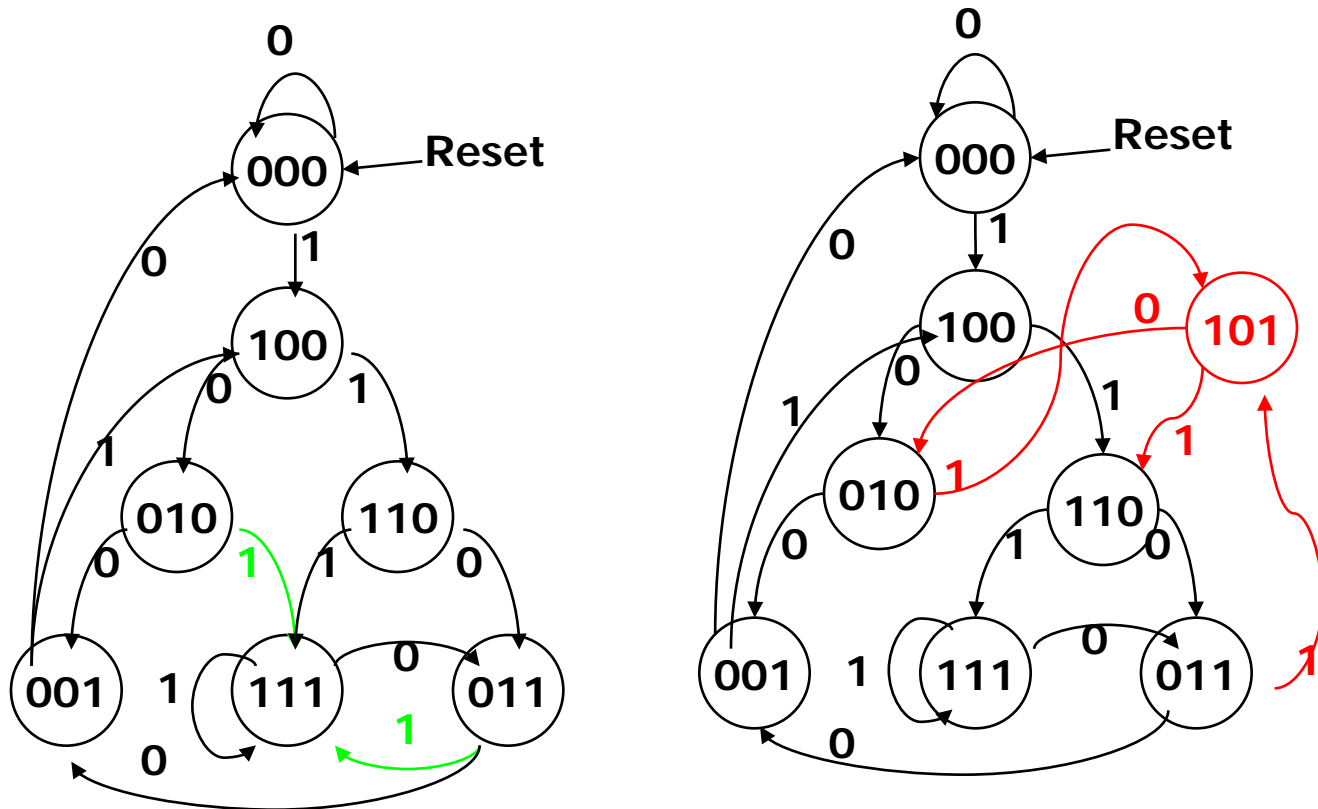
NS1 = Reset' (I)

NS2 = Reset' (PS1 + PS2 I)

NS3 = Reset' PS2

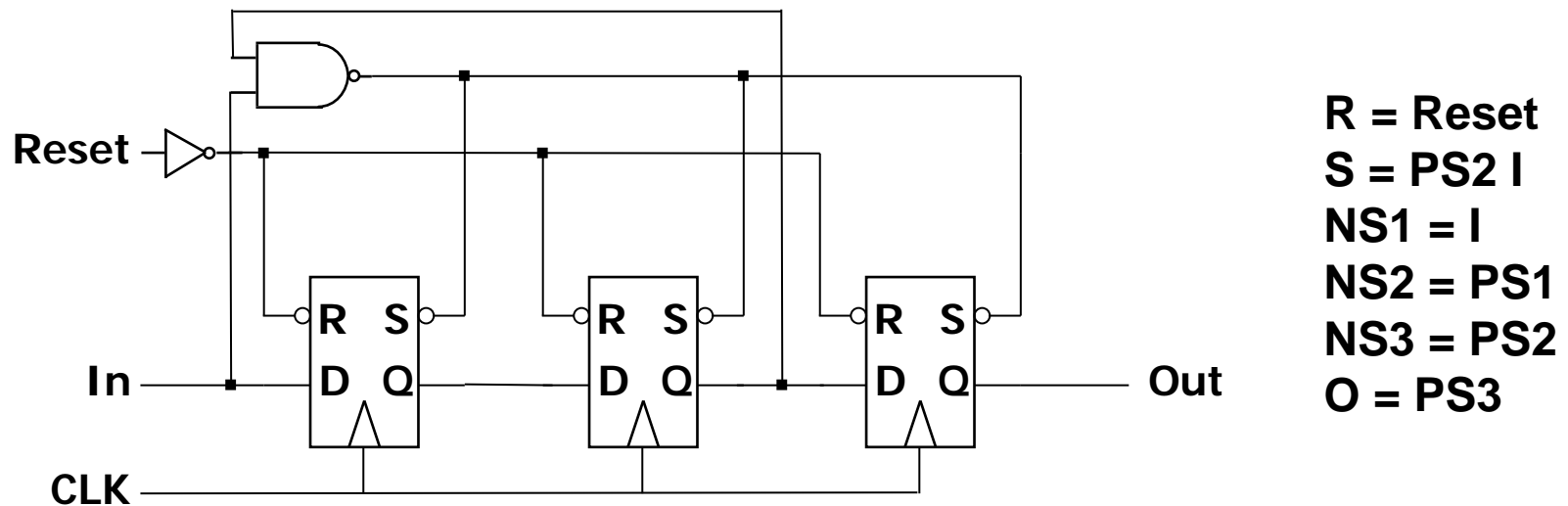
O = PS3

Median filter FSM



- But it looks like a shift register if you look at it right

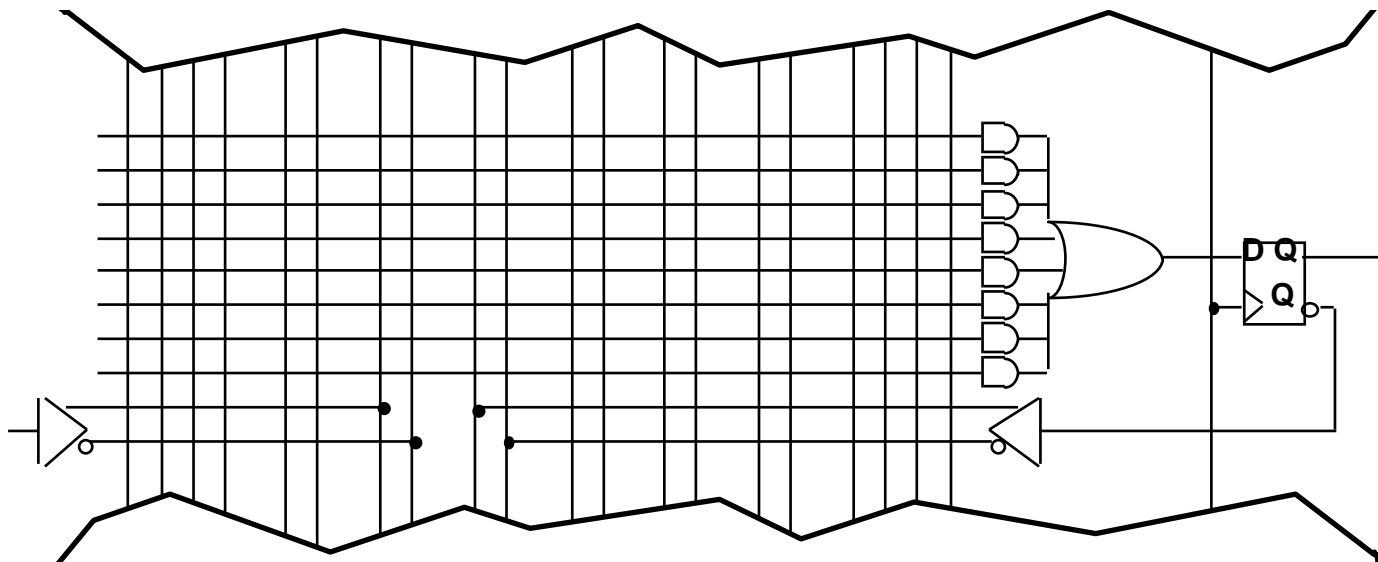
Median filter FSM – with SR FFs



The set input (S) does the median filter function by making the next state 111 whenever the input is 1 and PS2 is 1 (1 input to state x1x)

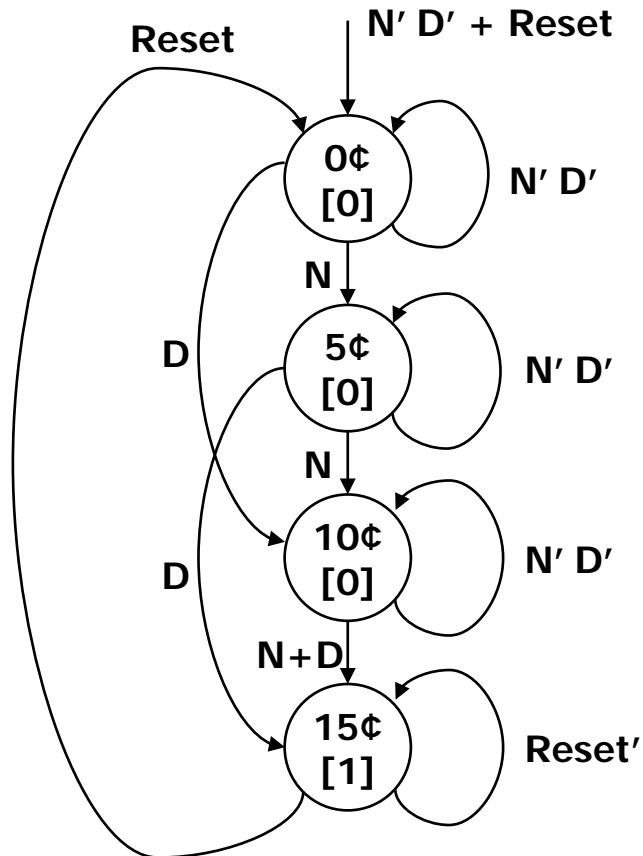
Implementation using PALs

- Programmable logic building block for sequential logic
 - macro-cell: FF + logic
 - D-FF
 - two-level logic capability like PAL (e.g., 8 product terms)

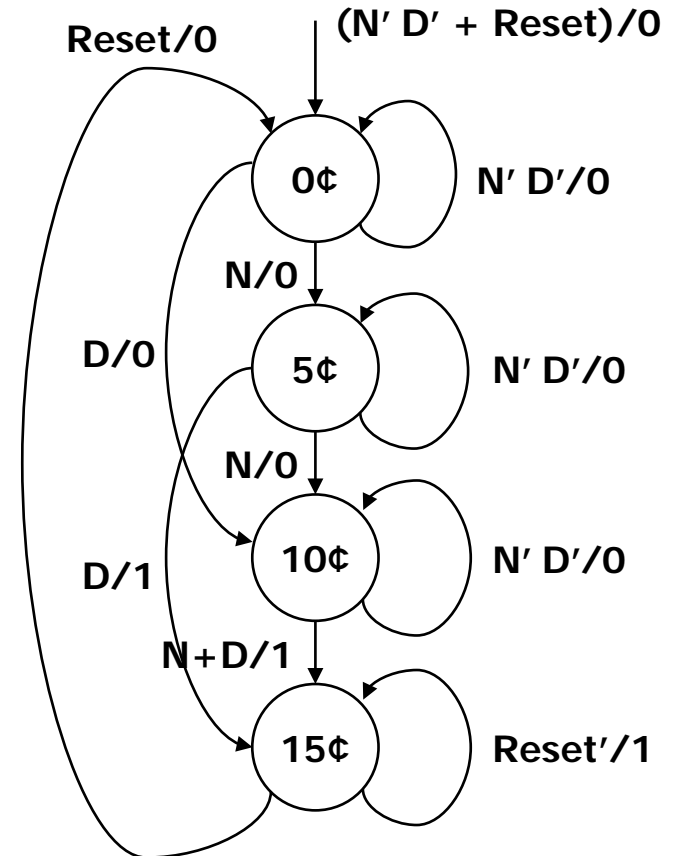


Equivalent Mealy and Moore state diagrams

- Moore machine
 - outputs associated with state



- Mealy machine
 - outputs associated with transitions

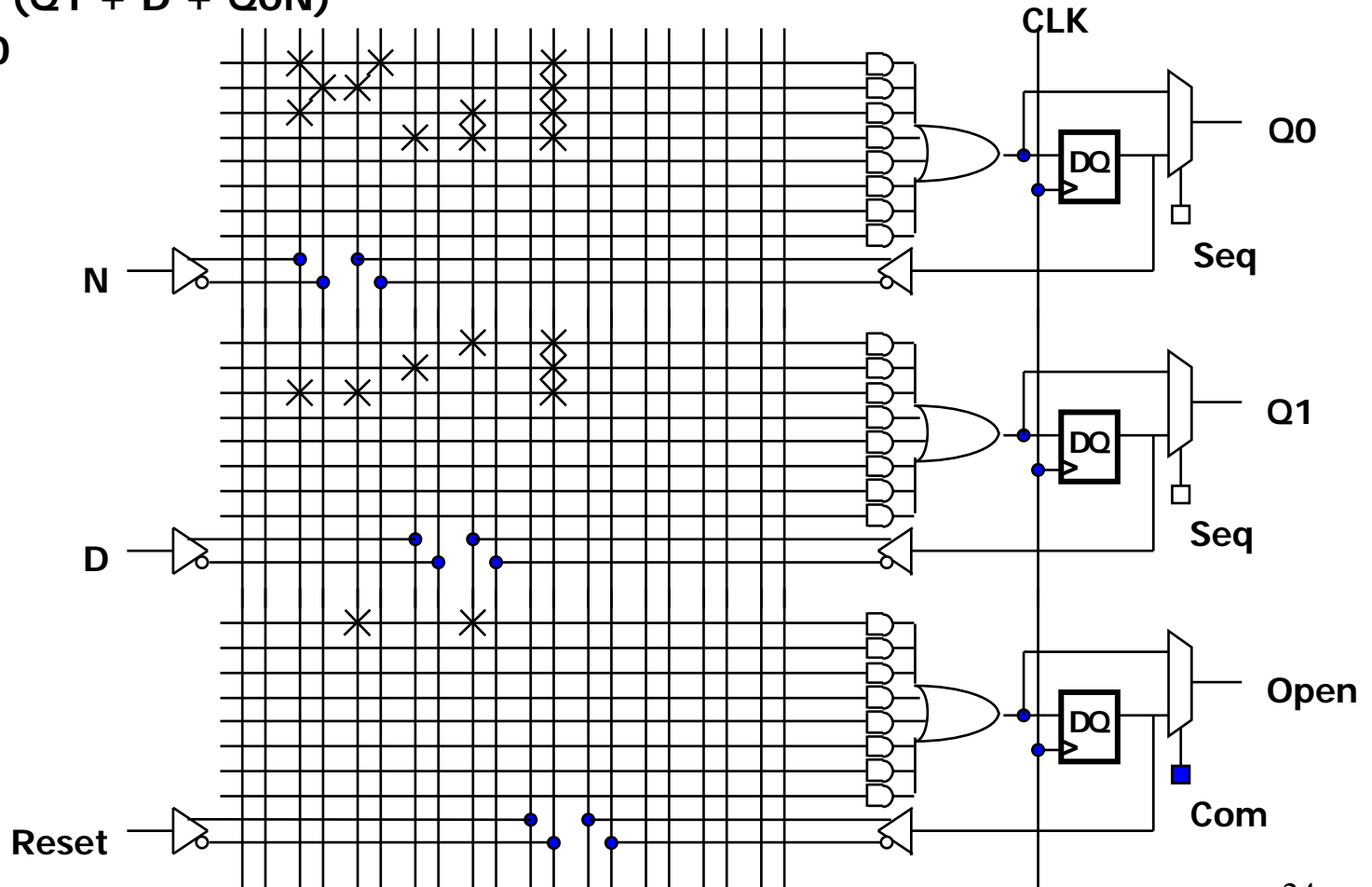


Vending machine example (Moore PLD mapping)

D0 = $\text{reset}'(\text{Q0}'\text{N} + \text{Q0N}' + \text{Q1N} + \text{Q1D})$

D1 = $\text{reset}'(\text{Q1} + \text{D} + \text{Q0N})$

OPEN = Q1Q0



Vending machine (synch. Mealy PLD mapping)

$$\text{OPEN} = \text{reset}'(\text{Q1Q0N}' + \text{Q1N} + \text{Q1D} + \text{Q0}'\text{ND} + \text{Q0N}'\text{D})$$

