Logic Design V

CSCI 2021: Machine Architecture and Organization

Antonia Zhai
Department Computer Science and Engineering
University of Minnesota
http://www.cs.umn.edu/~zhai

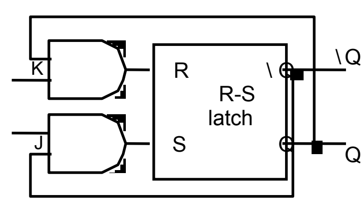
With Slides from Hai Zhou Stephen McCamant and Wei-Chung Hsu

UNIVERSITY OF MINNESOTA

JK Latch Design

How to eliminate the forbidden state?

J(t)	K(t)	Q(t)	Q(t+d)	
0	0	0	0	HOLD
0	0	1	1	
0	1	0	0	RESET
0	1	1	0	
1	0	0	1	SET
1	0	1	1	
1	1	0	1	TOGGLE
1	1	1	0	



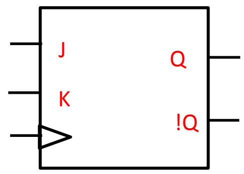
Idea: use output feedback to guarantee that R and S are never both one

J, K both one yields toggle

Characteristic Equation:

$$Q+ = Q\,\overline{K} + \overline{Q}\,J$$

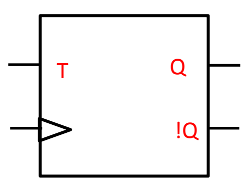
J-K Flip-Flop



- More powerful input type
  - Like combination of S-R and T
- Cases:
  - J = K = 0: no change
  - J = 1: set; K = 1: reset
  - J = K = 1: toggle
- Q' = (Q & !K) | (!Q & J)

Take Home: How to design an edge-triggered JK flip-flop?

T Flip-Flop



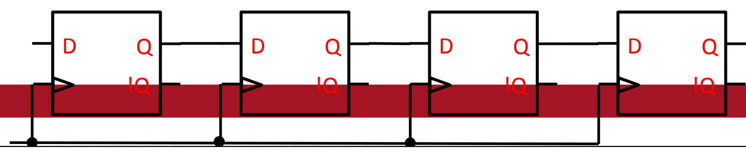
- Another input style, T = "toggle"
- Flip-flop behavior summarized by state update formula
  - Here, Q' = (Q ^ T)
  - T = 0: unchanged; T = 1: value is negated

Build a T Flip-Flop from a JK Flip-Flop

Shift Register

- Flip-flops connected in series
- Behavior:
  - Sequence of bits each move one stage per clock cycle
- Variations:
  - Serial or parallel input
  - Series or parallel output
  - Shift only on some cycles

How to support parallel input?



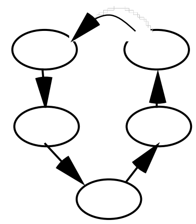
Counters

- Simple kind of time-varying digital system
  - Produces a single sequence of states, repeating
  - Changes every cycle or on a count pulse
- Example: 3-bit binary up-counter
  - Produces 000, 001, 010, 011, 100, 101, 110, 111, 000, 001, …
- Variations:
  - Down-counters
  - Decade counters (for decimal): 0 through 9
  - Gray code: sequence where only one bit changes at a time
  - Ring counter: circular shift register producing one-hot outputs

Counter Design

Step 1: Derive the State Transition Diagram

Count sequence: 000, 010, 011, 101, 110

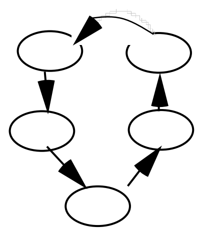


Present State	Next State
0 0 0	0 1 0
0 1 0	0 1 1
0 1 1	1 0 1
1 0 1	1 1 0
1 1 0	0 0 0

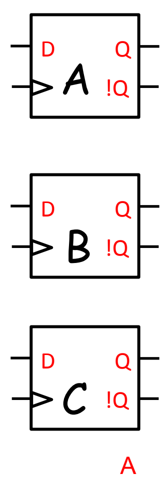
Step 2: State Transition Table

Implementing 5-state counter with D FFs

Continuing with the 000, 010, 011, 101, 110,



Present State A B C	Next State D_A D_B D_C
0 0 0	0 1 0
0 0 1	X X X
0 1 0	0 1 1
0 1 1	1 0 1
1 0 0	X X X
1 0 1	1 1 0
1 1 0	0 0 0
1 1 1	X X X



Implementing 5-state counter with RS FFs

Continuing with the 000, 010, 011, 101, 110

Present State CBA	Next State D_C D_B D_A
0 0 0	0 1 0
0 0 1	X X X
0 1 0	0 1 1
0 1 1	1 0 1
1 0 0	X X X
1 0 1	1 1 0
1 1 0	0 0 0
1 1 1	X X X

D_C

A\BC	00	01	11	10
0	0	X	0	0
1	X	1	X	1

D_B

A\BC	00	01	11	10
0	1	X	0	1
1	X	1	X	0

D_A

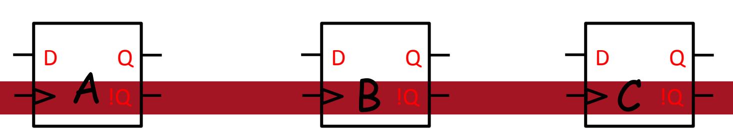
A\BC	00	01	11	10
0	0	X	0	1
1	X	0	X	1

Implementing 5-state counter with D FFs

DC = A

DB = A' C' + B'

DA = B C'



Implementing 5-state counter with RS FFs

Continuing with the 000, 010, 011, 101, 110

Q	Q+	R	S
0	0	X	0
0	1	0	1
1	0	1	0
1	1	0	X

$Q+ = S + R\,\overline{Q}$

Present State	Next State	RC	SC	RB	SB	RA	SA
		Rmepped next state					
0 0 0	0 1 0	X	0	0	1	X	0
0 0 1	X X X	X	X	X	X	X	X
0 1 0	0 1 1	X	0	0	X	0	1
0 1 1	1 0 1	0	1	1	0	0	X
1 0 0	X X X	X	X	X	X	X	X
1 0 1	1 1 0	0	X	0	1	1	0
1 1 0	0 0 0	1	0	1	0	X	0
1 1 1	X X X	X	X	X	X	X	X

 Remapped Next State Functions

Implementation with RS FFs

RS FFs Continued

RC

A \ CB	00	01	11	10
0	X	X	1	X
1	X	0	X	0

SC

A \ CB	00	01	11	10
0	0	0	0	X
1	X	1	X	X

RB

A \ CB	00	01	11	10
0	0	0	1	X
1	X	1	X	0

SB

A \ CB	00	01	11	10
0	1	X	0	X
1	X	0	X	1

RA

A \ CB	00	01	11	10
0	X	0	X	X
1	X	0	X	1

SA

A \ CB	00	01	11	10
0	0	1	0	X
1	X	X	X	0

$RC = \overline{A}$

$SC = A$

$RB = A\,B + B\,C$

$SB = \overline{B}$

$RA = C$

$SA = B\,\overline{C}$

Basic Design Approach

1. Understand the statement of the Specification
2. Obtain an abstract specification of the FSM
3. Perform a state mininimization
4. Perform state assignment
5. Choose FF types to implement FSM state register
6. Implement the FSM

Example: Vending Machine FSM

deliver package of gum after 15 cents deposited

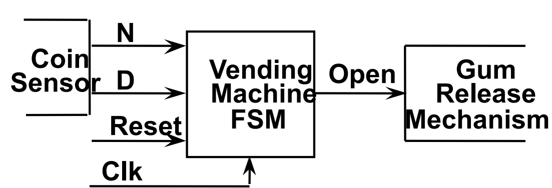
single coin slot for dimes, nickels

no change

Step 1. *Understand the problem:*

Draw a picture!

Block Diagram



Vending Machine Example

Step 2. ***Map into more suitable abstract representation***

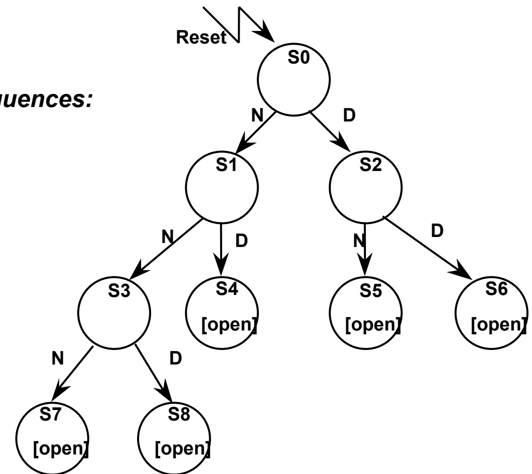
Tabulate possible input sequences:

three nickels
nickel, dime
dime, nickel
two dimes
two nickels, dime

Draw state diagram:

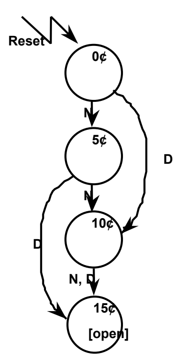
Inputs: N, D, reset

Output: open



Vending Machine Example

Step 3: State Minimization



reuse states whenever possible

Present State	Inputs D	N	Next State	Output Open
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	X	X
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	0
	1	1	X	X
10¢	0	0	10¢	0
	0	1	15¢	0
	1	0	15¢	0
	1	1	X	X
15¢	X	X	15¢	1

Symbolic State Table

Vending Machine Example

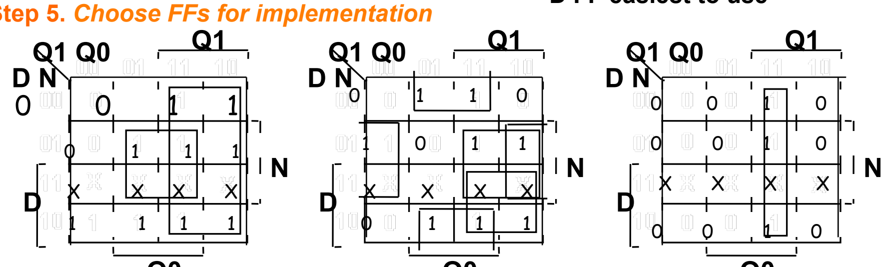
Step 4: State Encoding

Present State Q_1	Present State Q_0	Inputs D	Inputs N	Next State D_1	Next State D_0	Output Open
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	X	X	X
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	0
		1	1	X	X	X
1	0	0	0	1	0	0
		0	1	1	1	0
		1	0	1	1	0
		1	1	X	X	X
1	1	0	0	1	1	1
		0	1	1	1	1
		1	0	1	1	1
		1	1	X	X	X

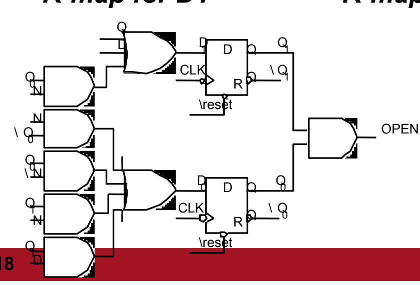
Vending Machine Example

D FF easiest to use

Step 5. *Choose FFs for implementation*



K-map for D1 *K-map for D0* *K-map for Open*

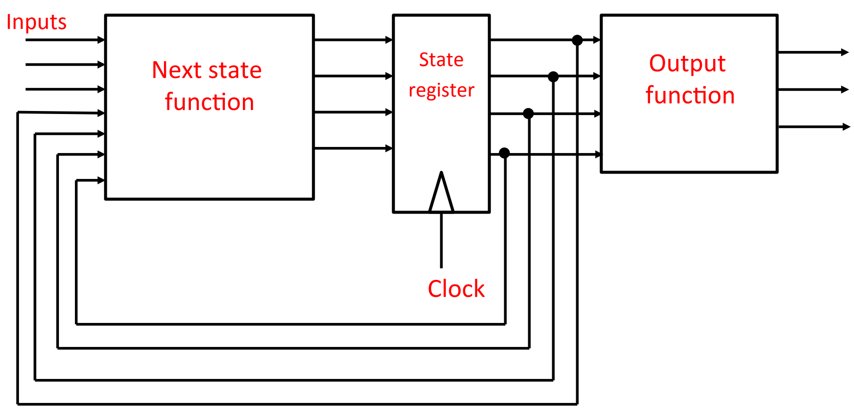


D1 = Q1 + D + Q0 N

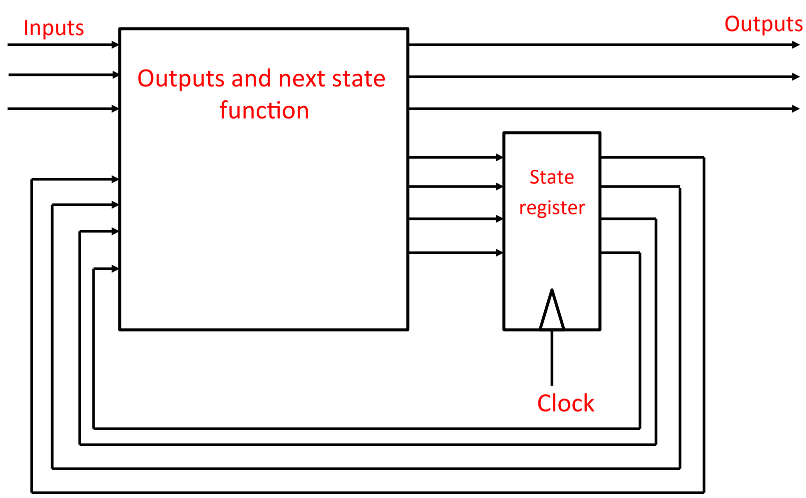
D0 = N $\overline{\text{Q0}}$ + Q0 $\overline{\text{N}}$ + Q1 N + Q1 D

OPEN = Q1 Q0

Moore Machine Block Diagram



Mealy Machine Block Diagram



Basic Design Approach

1. Understand the statement of the Specification
2. Obtain an abstract specification of the FSM
3. Perform a state mininimization
4. Perform state assignment
5. Choose FF types to implement FSM state register
6. Implement the FSM