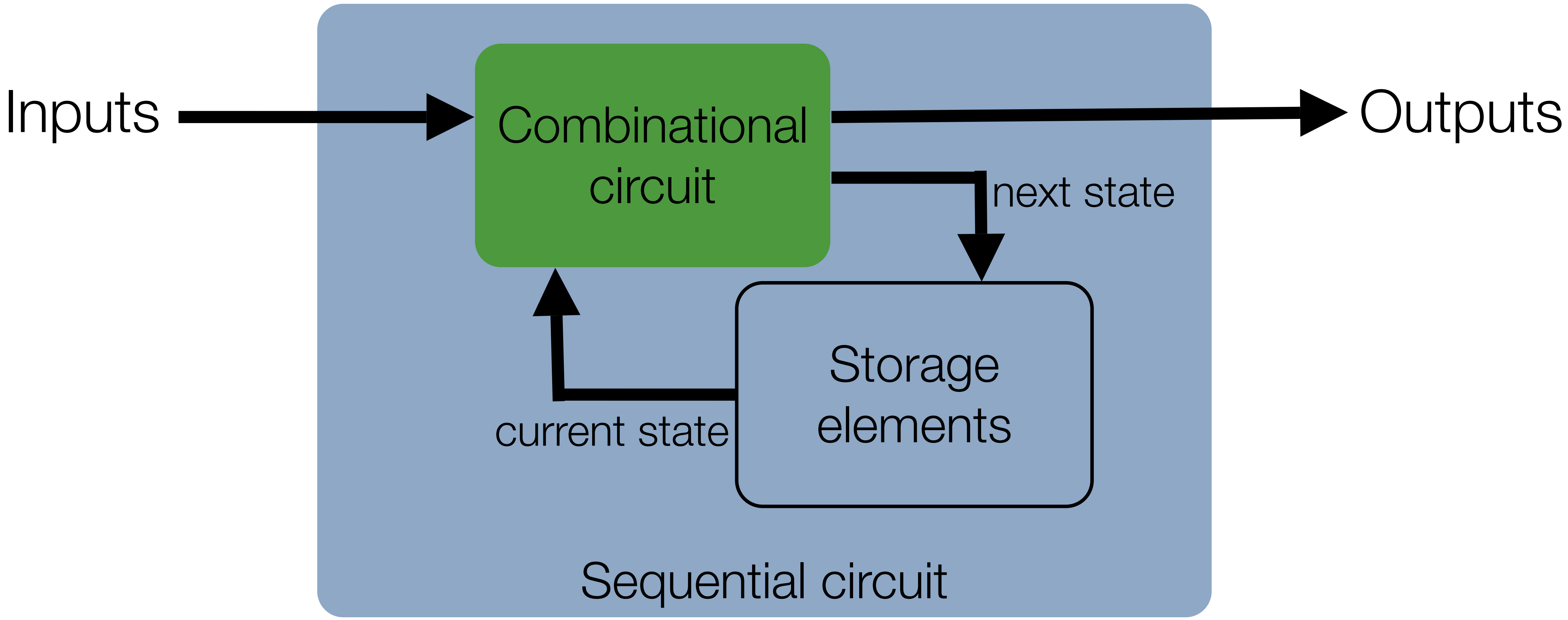
CSEE 3827: Fundamentals of Computer Systems

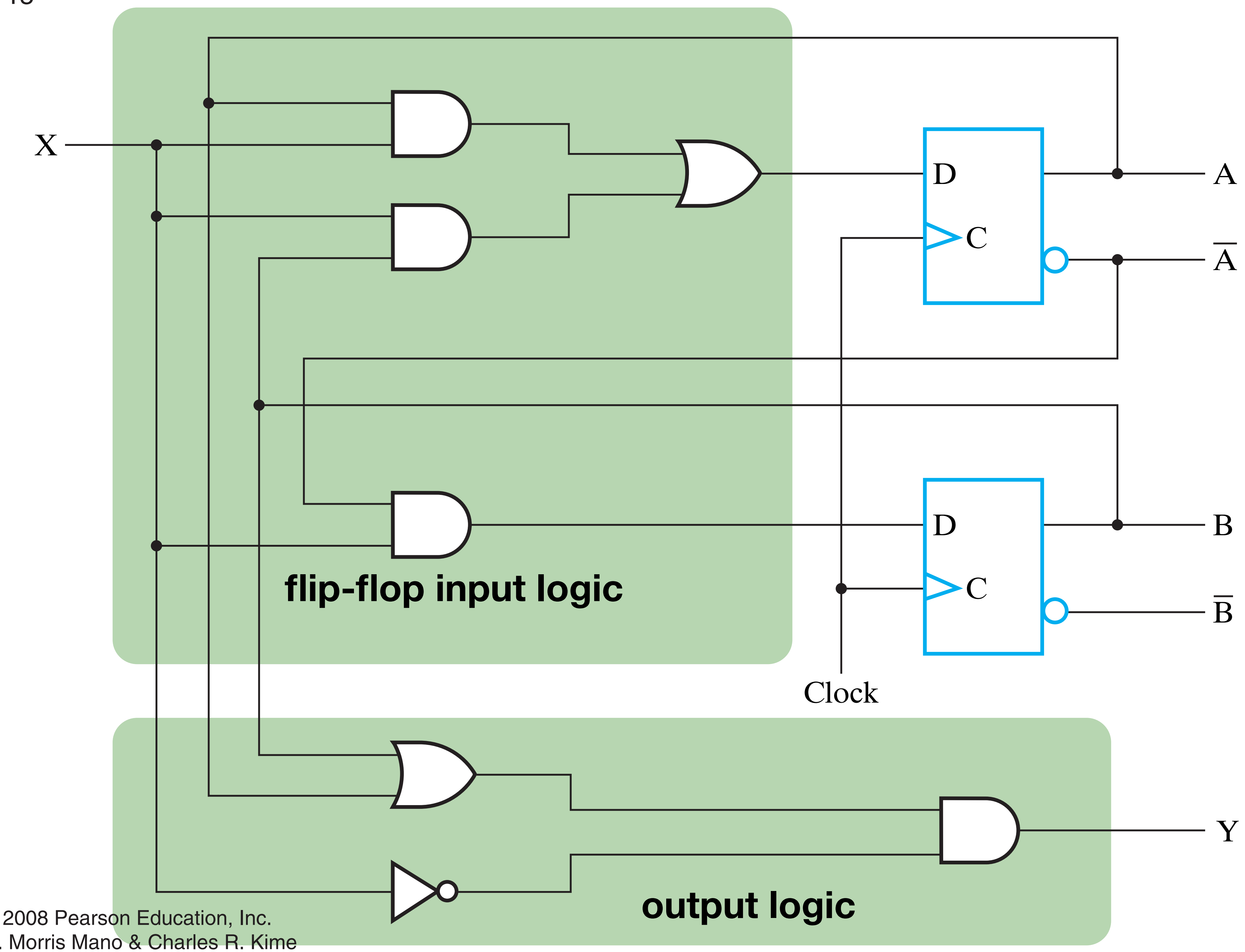
Finite State Machine Design

Recall: Sequential circuit

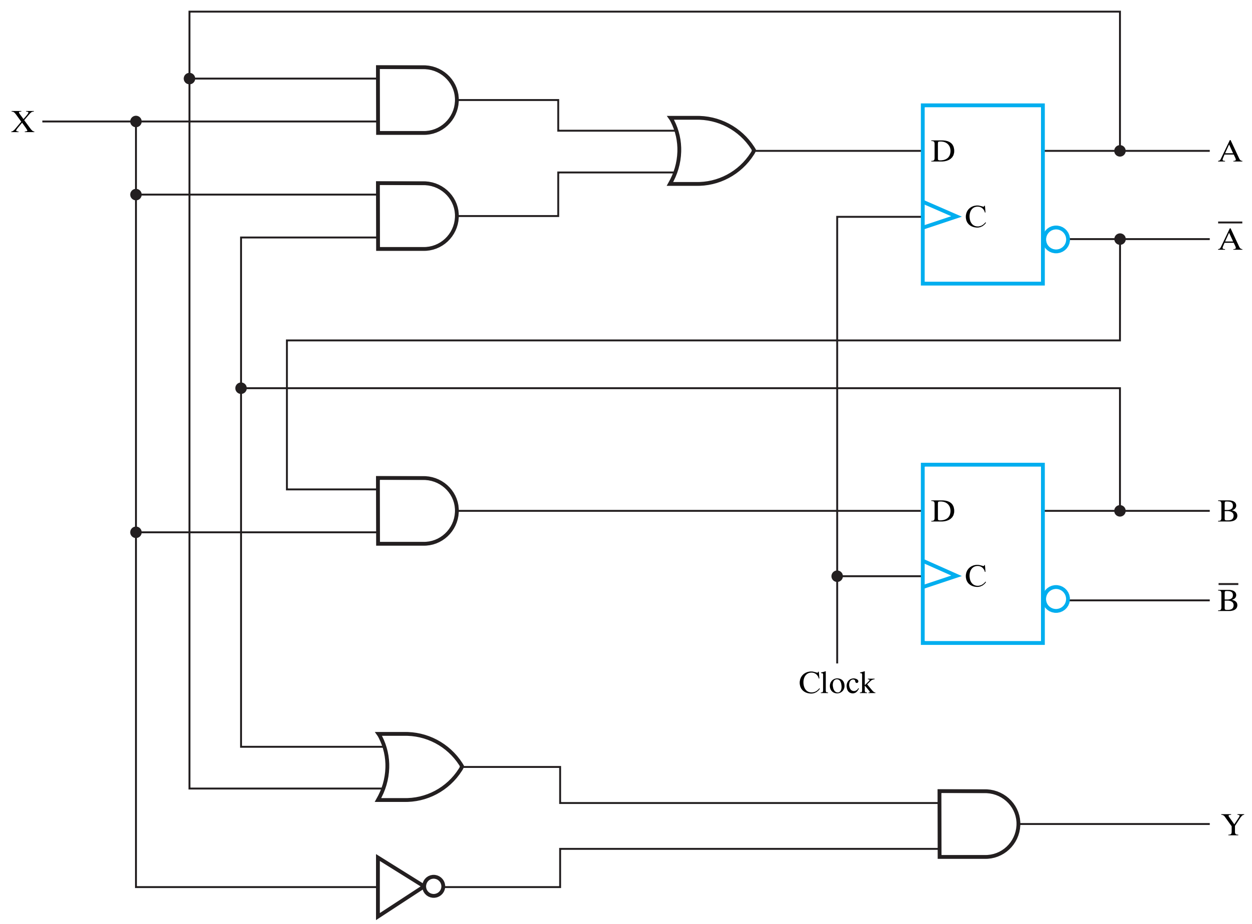


Example sequential circuit (schematic)

5-15



Reverse engineering a sequential circuit

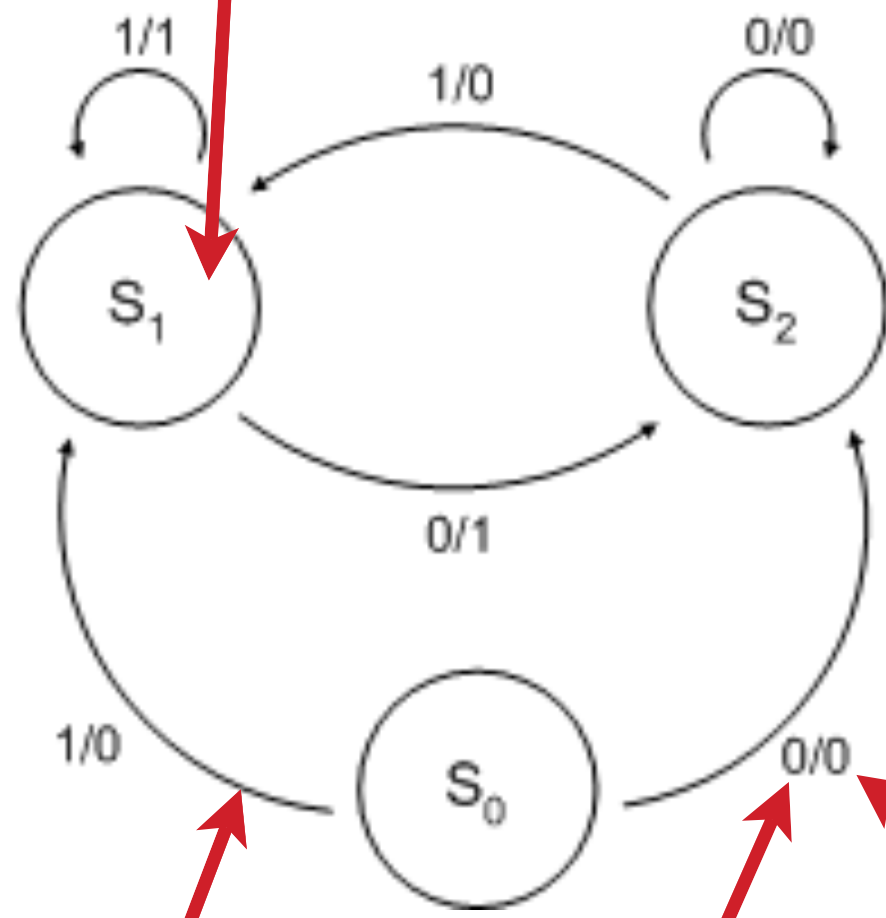


State machine

A state machine model of a system's behavior in terms of ***states*** *and* ***transitions*** *between those states that are triggered by* ***actions****.*

State diagrams represent state machines

one or more states, indicated by nodes



edges between states

machine output at transition

input value that triggers transition on edge

Finite state machine (FSM)

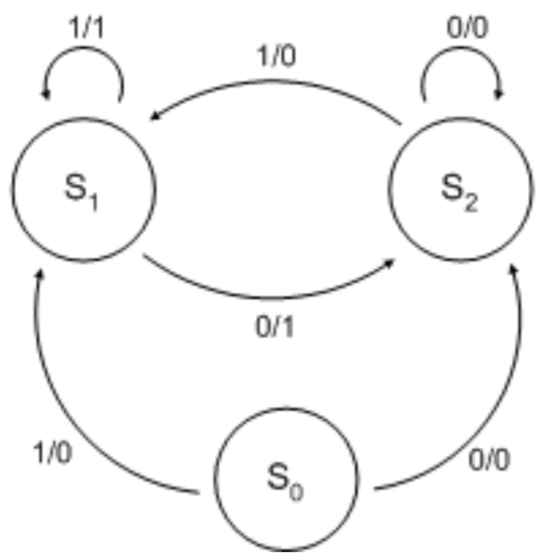
A state machine that has a finite number of states

* *Any finite state machine can be implemented with sequential logic*

* *All sequential circuits implement finite state machines*

Implementing a finite state machine

1. describe operation



2. convert to truth table

S	in	S+	out
0 0	0	1 0	0
0 0	1	0 1	0
0 1	0	1 0	1
0 1	1	0 1	1
1 0	0	1 0	0
1 0	1	0 1	0

3. choose type of flip-flop

4. annotate table with flip-flop inputs for next state

S	in	S+	out	T1	T2
0 0	0	1 0	0	1	0
0 0	1	0 1	0	0	1
0 1	0	1 0	1	1	1
0 1	1	0 1	1	0	0
1 0	0	1 0	0	0	0
1 0	1	0 1	0	1	1

5. derive “next state” and “output” logic

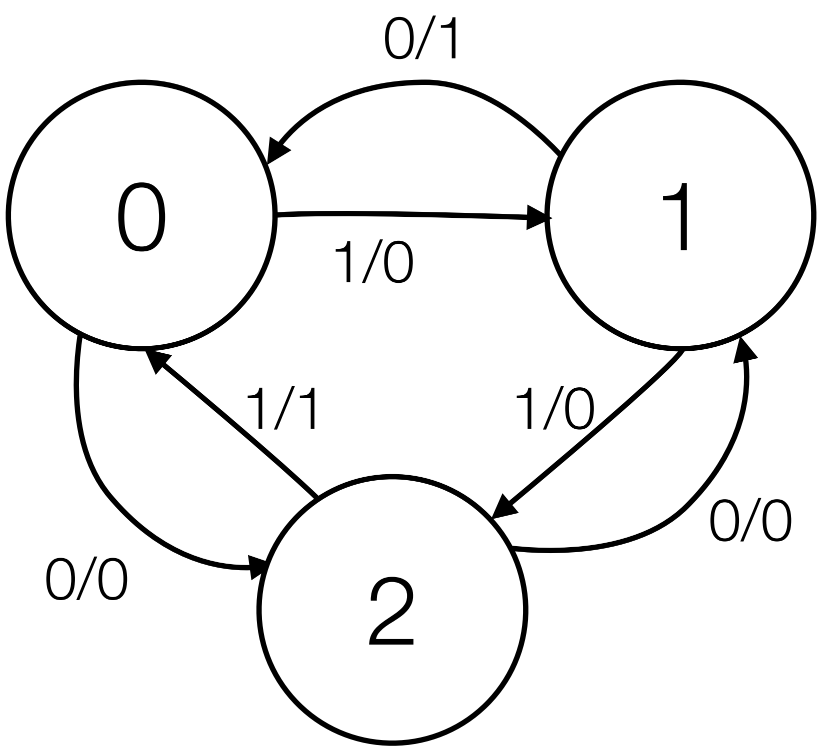
6. wire circuit and flip-flops together together

From State Machine to Sequential Circuit

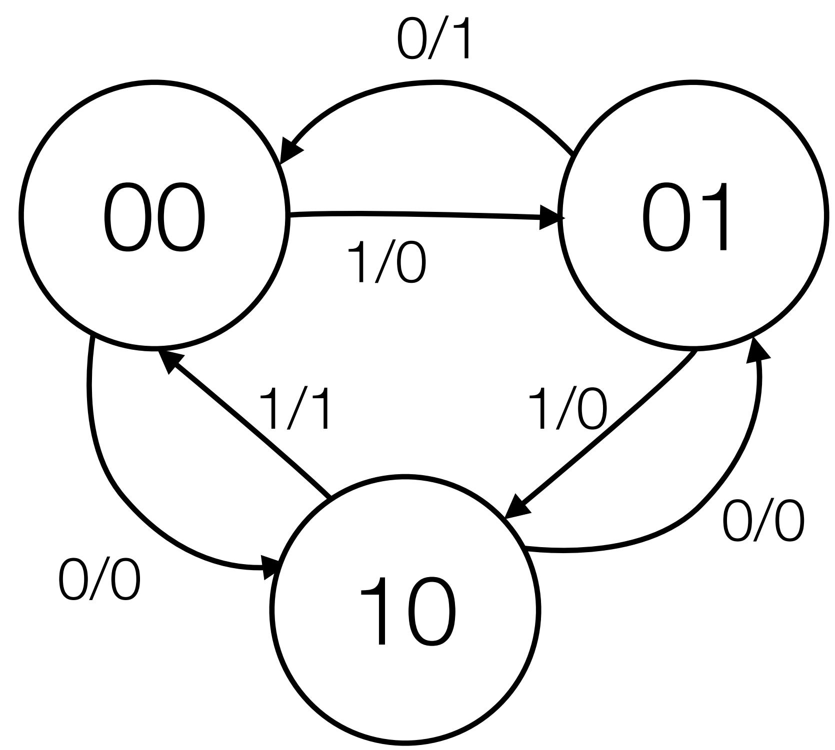
- Specify behavior of state machine (including input and output values)
- Encode states
  - *figure out how many bits needed to store state (one FF per bit)*
  - *assign state values to states*
- Select FF type (i.e., D, T, JK, etc.)
- Compute combinational logic functions
  - *"next state" logic: S+ = F(S,inputs)*
  - *"output" logic:*
    - Mealy machine: output = G(S,inputs)
    - Moore machine: output = H(S)

Example State Machine

- 1 input, 1 output
- Let X = # of 1's input so far, Y = # of 0's input so far.
- Output 1 whenever X-Y = 0 (mod 3)



State label is current value of X-Y mod 3



Binary Labeling

Design with D Flip-Flops

- D FF’s are easy: we input the value to the FF that we want it set to

A_{cur}	B_{cur}	In	A_{next}	B_{next}	Out
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	X	X	X
1	1	1	X	X	X

A_{cur}	B_{cur}	In	A_{next}	D_A	B_{next}	D_B	Out
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	0
0	1	0	0	0	0	0	1
0	1	1	1	1	0	0	0
1	0	0	0	0	1	1	0
1	0	1	0	0	0	0	1
1	1	0	X	X	X	X	X
1	1	1	X	X	X	X	X

New columns indicate what values feed into D FF (mimic “next” values for that FF)

Design with D Flip-Flops Cont'd

- Build K-Maps, get equations for output and FF input vals (in terms of inputs and previous F vals)

A_{cur}	B_{cur}	In	A_{next}	D_A	B_{next}	D_B	Out
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	0
0	1	0	0	0	0	0	1
0	1	1	1	1	0	0	0
1	0	0	0	0	1	1	0
1	0	1	0	0	0	0	1
1	1	0	X	X	X	X	X
1	1	1	X	X	X	X	X

D_A:

In

	1	0	1	0
A_{cur}	0	0	X	X

B_{cur}

D_B:

In

	0	1	0	0
A_{cur}	1	0	X	X

B_{cur}

Out:

In

	0	0	0	1
A_{cur}	0	1	X	X

B_{cur}

$$D_A = \bar{A}\bar{B}\bar{I} + BI$$

$$D_B = A\bar{B}\bar{I} + \bar{A}\bar{B}I$$

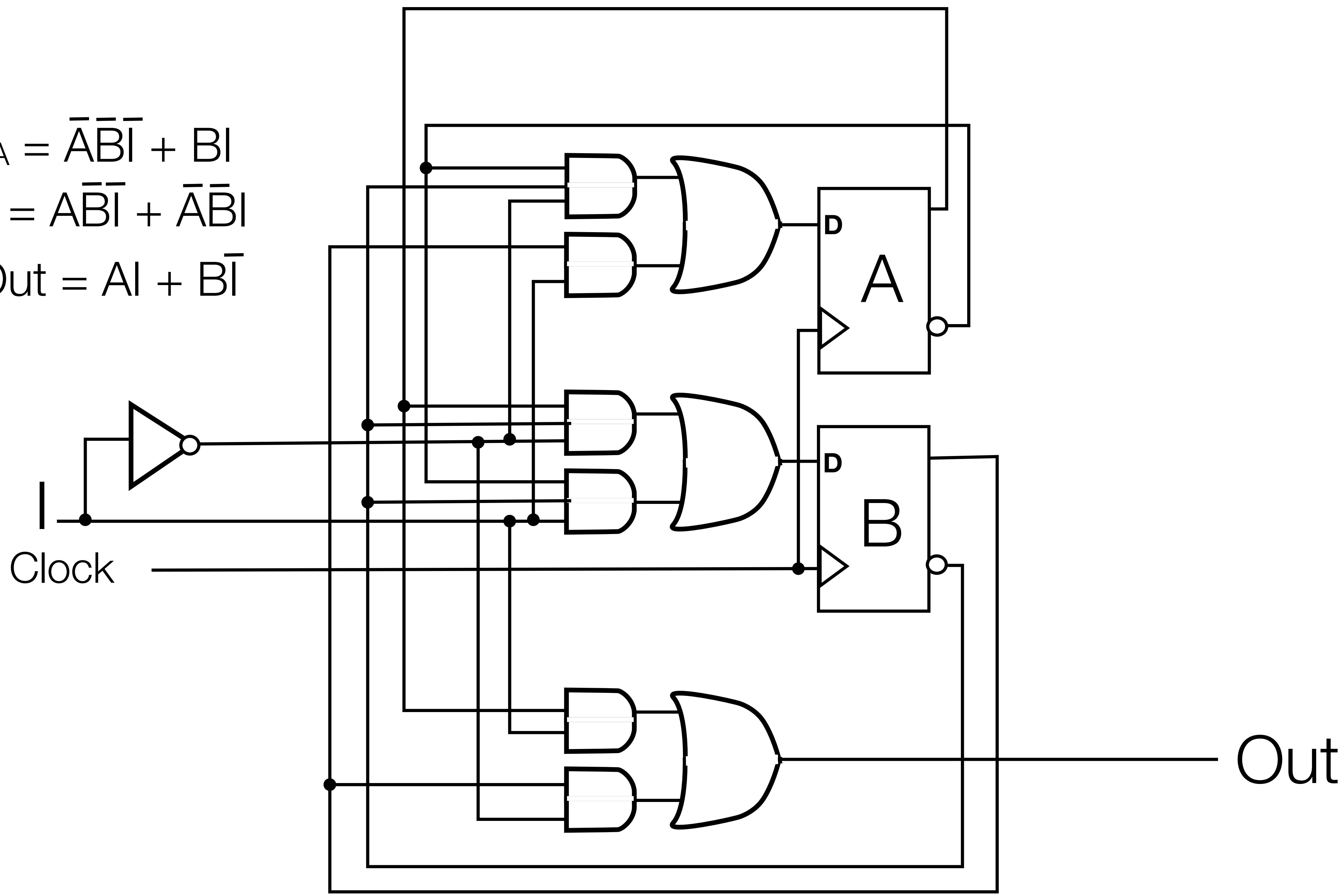
$$Out = AI + B\bar{I}$$

Design with D FF's cont'd

$D_A = \bar{A}\bar{B}\bar{I} + BI$

$D_B = A\bar{B}\bar{I} + \bar{A}\bar{B}I$

$Out = AI + B\bar{I}$



Design with T Flip Flops

- Value fed into T FF is 0 if FF should maintain value, 1 if it should flop

A_{cur}	B_{cur}	In	A_{next}	B_{next}	Out
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	X	X	X
1	1	1	X	X	X

A_{cur}	B_{cur}	In	A_{next}	T_A	B_{next}	T_B	Out
0	0	0	1	1	0		0
0	0	1	0	0	1		0
0	1	0	0	0	0		1
0	1	1	1	1	0		0
1	0	0	0	1	1		0
1	0	1	0	1	0		1
1	1	0	X	X	X		X
1	1	1	X	X	X		X

T_A:

In

	1	0	1	0
A_{cur}	1	1	X	X

B_{cur}

$$T_A = A + BI + \bar{B}\bar{I}$$

Design with T Flip Flops

- Value fed into T FF is 0 if FF should maintain value, 1 if it should flop

A_{cur}	B_{cur}	In	A_{next}	B_{next}	Out
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	X	X	X
1	1	1	X	X	X

A_{cur}	B_{cur}	In	A_{next}	T_A	B_{next}	T_B	Out
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	0
0	1	0	0	0	0	1	1
0	1	1	1	1	0	1	0
1	0	0	0	1	1	1	0
1	0	1	0	1	0	0	1
1	1	0	X	X	X	X	X
1	1	1	X	X	X	X	X

T_B:

In

	0	1	0	0
A_{cur}	1	0	X	X

B_{cur}

$$T_B = A\bar{B}\bar{I} + \bar{A}\bar{B}I$$

Design with JK Flip-Flops

- Note: to change A_{cur} to the correct A_{next} value, two possible input pairs can be fed into the J,K inputs of a JK Flip-Flop

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

A_{cur}	A_{next}	J,K
0	0	0,0 or 0,1 (0,X)
0	1	1,0 or 1,1 (1,X)
1	0	0,1 or 1,1 (X,1)
1	1	0,0 or 1,0 (X,0)

Design with JK Flip-Flop

A_{cur}	A_{next}	J,K
0	0	0,0 or 0,1 (0,X)
0	1	1,0 or 1,1 (1,X)
1	0	0,1 or 1,1 (X,1)
1	1	0,0 or 1,0 (X,0)

A_{cur}	B_{cur}	In	A_{next}	B_{next}	Out
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	X	X	X
1	1	1	X	X	X

A_{cur}	B_{cur}	In	A_{next}	J_A	K_A	B_{next}	J_B	K_B	Out
0	0	0	1	1	X	0			0
0	0	1	0	0	X	1			0
0	1	0	0	0	X	0			1
0	1	1	1	1	X	0			0
1	0	0	0	X	1	1			0
1	0	1	0	X	1	0			1
1	1	0	X	X	X	X			X
1	1	1	X	X	X	X			X

Design with JK Flip-Flop

A_{cur}	A_{next}	J,K
0	0	0,0 or 0,1 (0,X)
0	1	1,0 or 1,1 (1,X)
1	0	0,1 or 1,1 (X,1)
1	1	0,0 or 1,0 (X,0)

A_{cur}	B_{cur}	In	A_{next}	B_{next}	Out
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	X	X	X
1	1	1	X	X	X

A_{cur}	B_{cur}	In	A_{next}	J_A	K_A	B_{next}	J_B	K_B	Out
0	0	0	1	1	X	0	0	X	0
0	0	1	0	0	X	1	1	X	0
0	1	0	0	0	X	0	X	1	1
0	1	1	1	1	X	0	X	1	0
1	0	0	0	X	1	1	1	X	0
1	0	1	0	X	1	0	0	X	1
1	1	0	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X

Design with JK Flip-Flop

A_{cur}	B_{cur}	In	A_{next}	J_A	K_A	B_{next}	J_B	K_B	Out
0	0	0	1	1	X	0	0	X	0
0	0	1	0	0	X	1	1	X	0
0	1	0	0	0	X	0	X	1	1
0	1	1	1	1	X	0	X	1	0
1	0	0	0	X	1	1	1	X	0
1	0	1	0	X	1	0	0	X	1
1	1	0	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X

J_A:

In

	1	0	1	0
A_{cur}	X	X	X	X

B_{cur}

K_A:

In

	X	X	X	X
A_{cur}	1	1	X	X

B_{cur}

J_B:

In

	0	1	X	X
A_{cur}	1	0	X	X

B_{cur}

K_B:

In

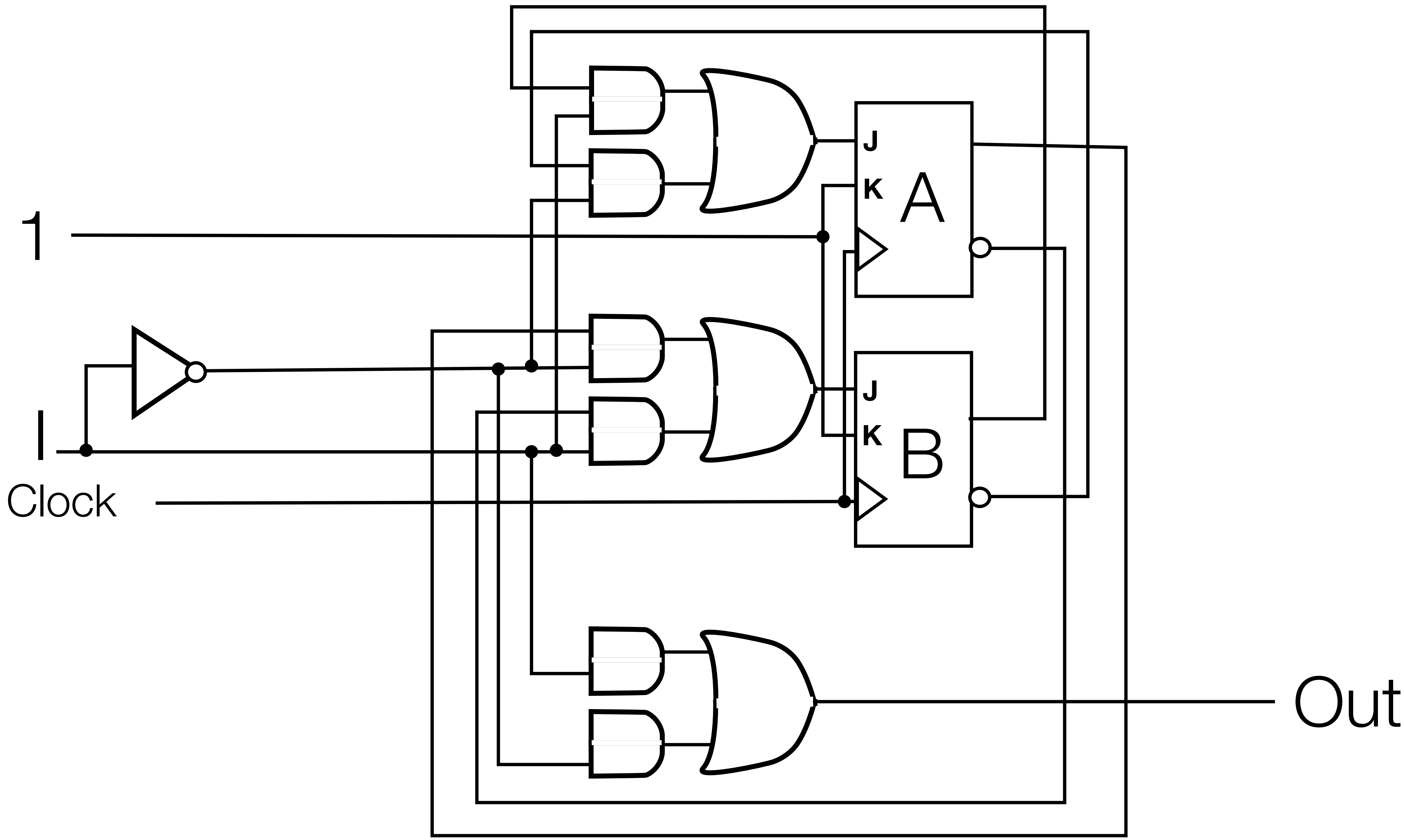
	X	X	1	1
A_{cur}	X	X	X	X

B_{cur}

$J_A = BI + \overline{B}\overline{I}$ $\quad K_A = 1$ $\quad J_B = \overline{A}I + A\overline{I}$ $\quad K_B = 1$

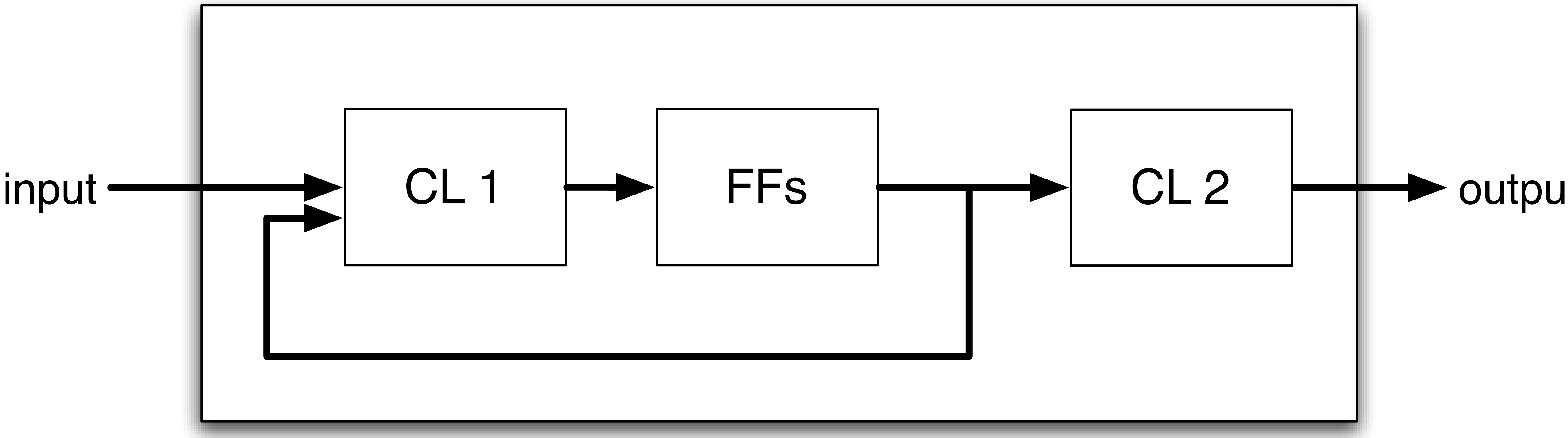
Design with JK FF's cont'd

$J_A = BI + \bar{B}\bar{I}$ $K_A = 1$ $J_B = \bar{A}I + A\bar{I}$ $K_B = 1$



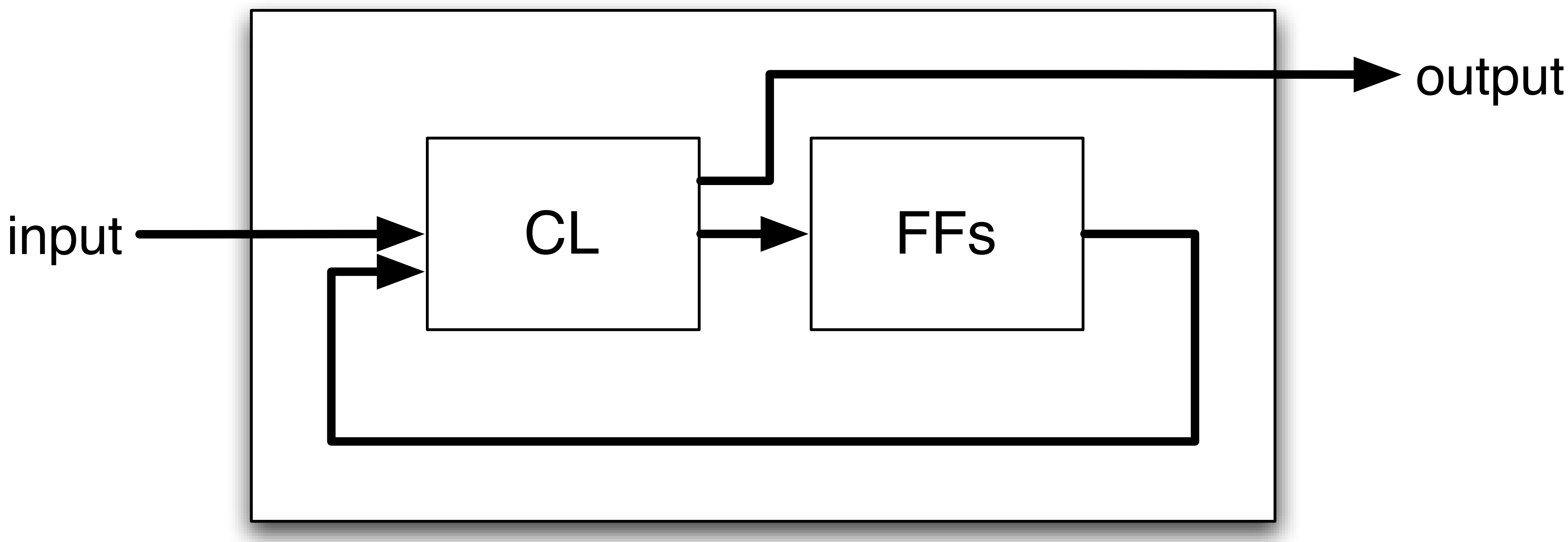
In class exercise: design a 3-bit counter

Moore machine



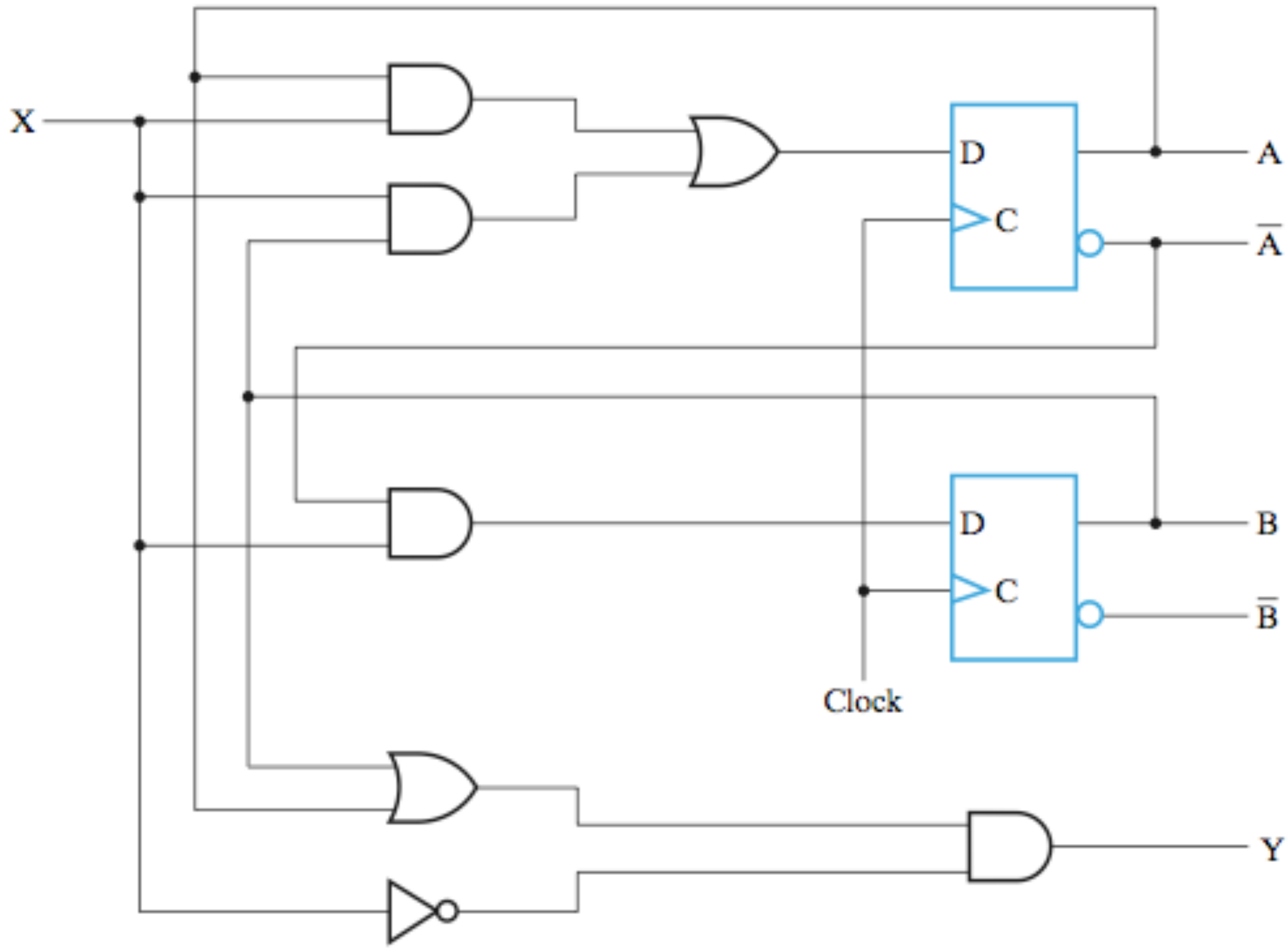
a circuit in which the output depends only on the current state

Mealy machine



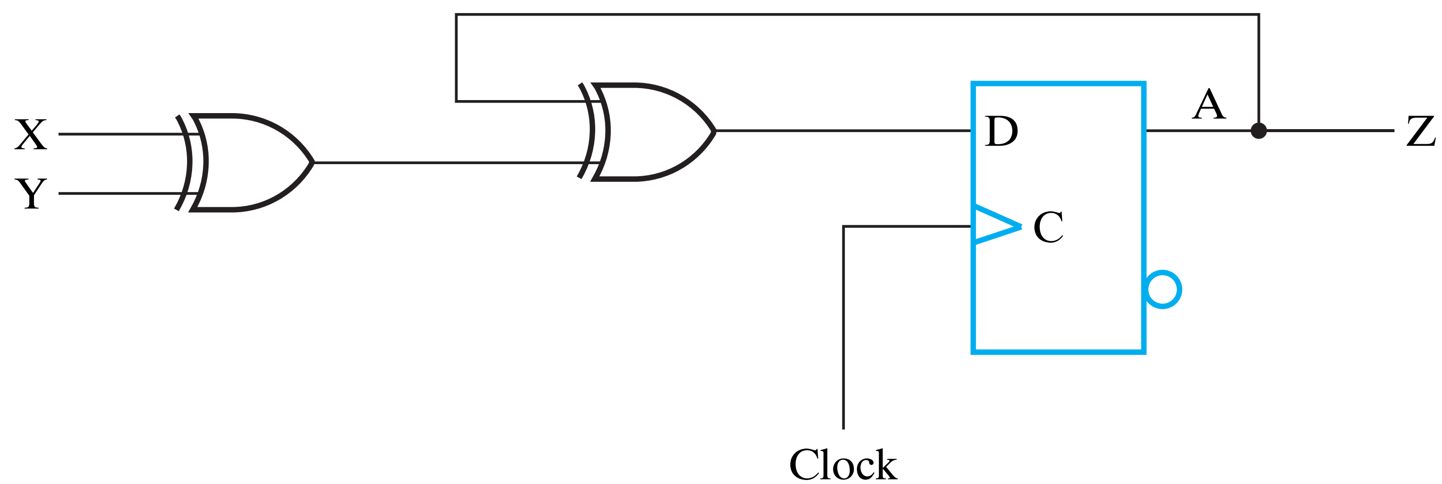
a circuit in which the outputs depend on the inputs as well as the current state

A Meały or Moore circuit?



An example Moore circuit

5-16



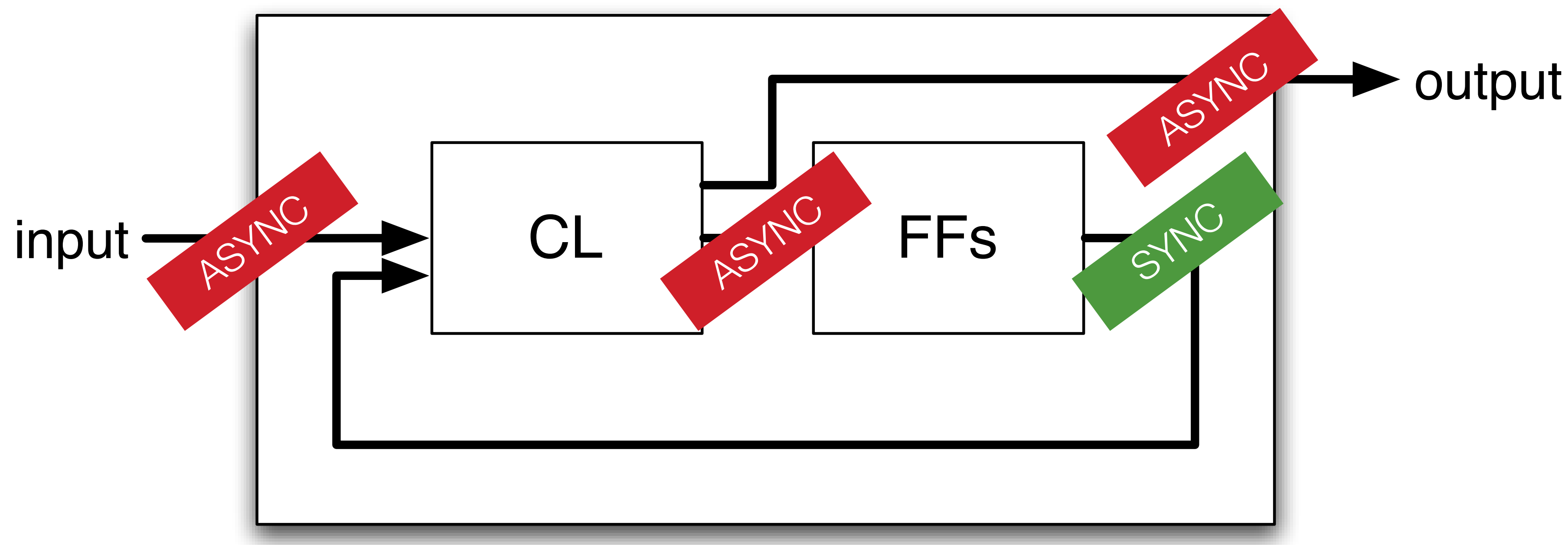
(a)

Present state	Inputs		Next state	Output
A	X	Y	A	Z
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

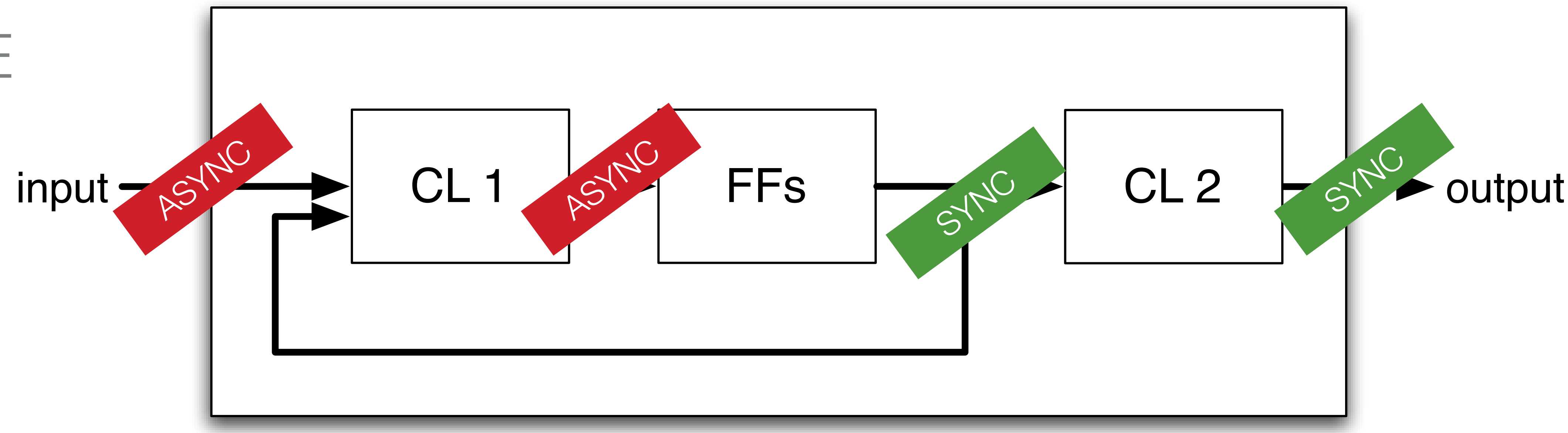
(b) State table

FSM timing characteristics

MEALY



MOORE



Advanced FSM design and implementation

Unused states: *extra state encodings (e.g., using 3 FFs to represent 6 states leaves 2 unused states) can be treated as "don't care" values and used to simplify the combinational logic*

State minimization: *two states are equivalent if they transition to the same or equivalent states on the same inputs (while producing the same outputs in the case of a Mealy machine)*