Part 6

***Digital Design and Computer Architecture*, 2nd Edition**

David Money Harris and Sarah L. Harris

Sequential Logic

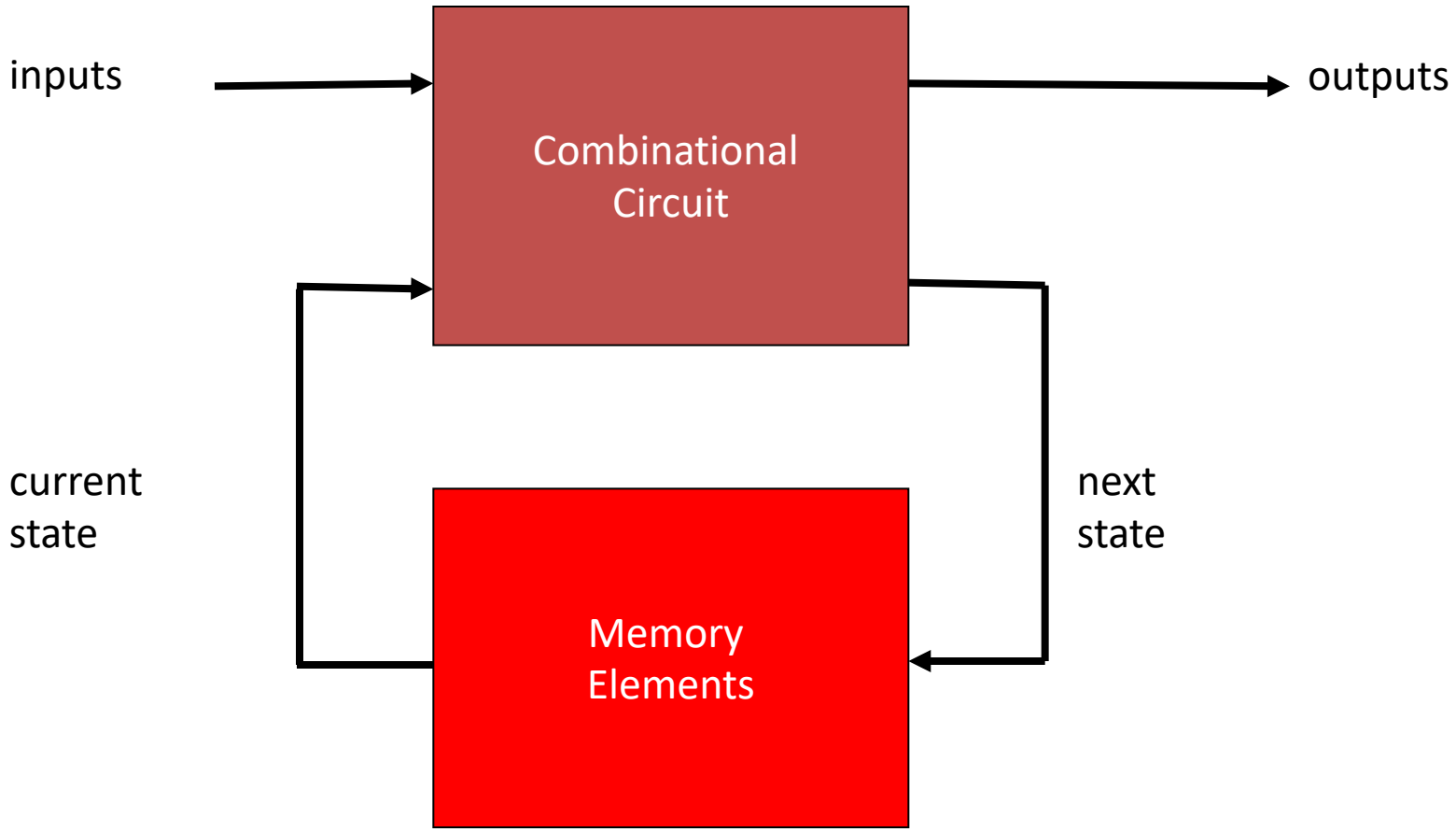
- Digital circuits we have learned, so far, have been **combinational**

  >> no memory,

  >> outputs are entirely defined by the "current" inputs

- However, many digital systems encountered in everyday life are **sequential** (i.e. **they have memory**)
  - **the memory elements remember past inputs**
  - **outputs of sequential circuits are not only dependent on the inputs but also the state of the memory elements.**

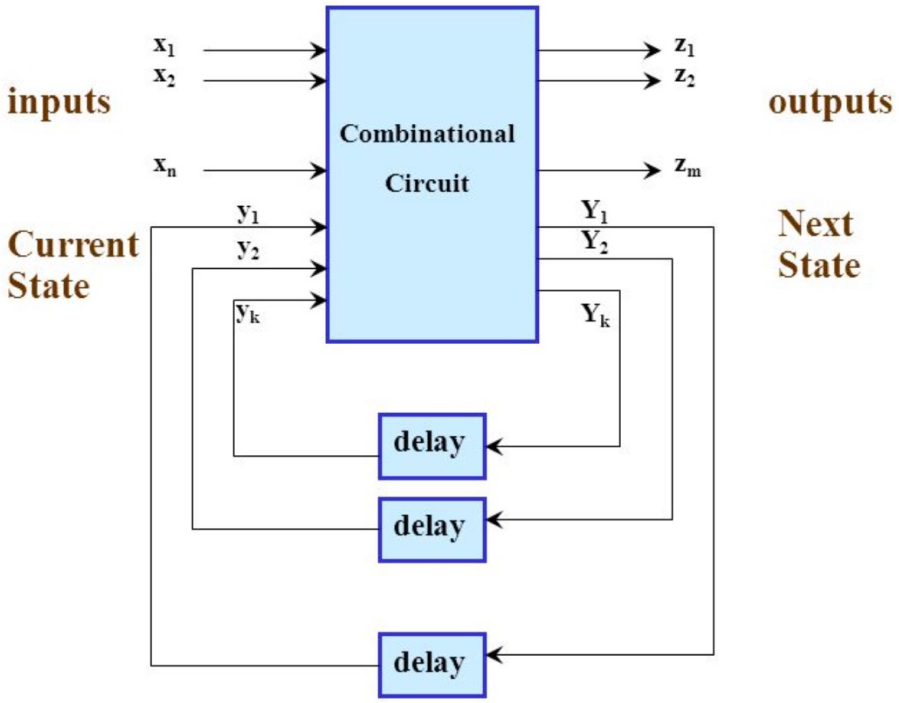
Sequential Circuits Model



current state is a function of **past inputs + initial state**

Asynchronous Sequential Circuits

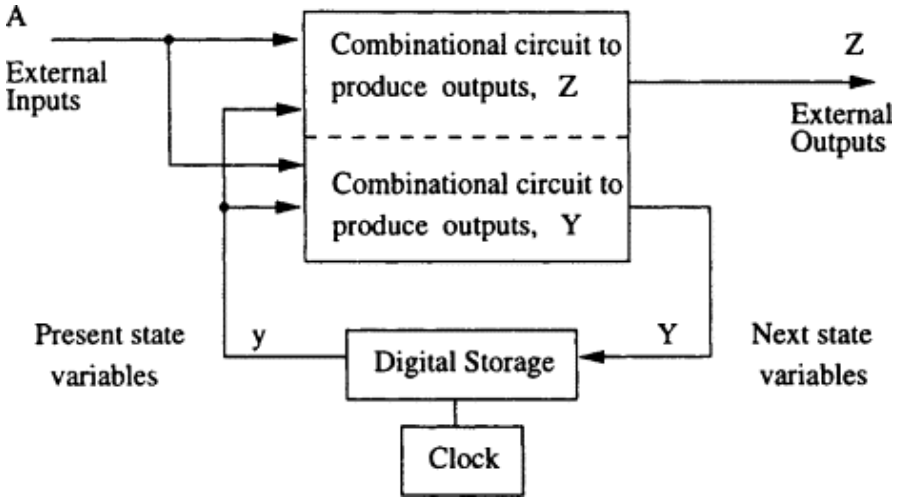
- Circuit behavior depends upon the input signals at any instant of time and the order in which the inputs change.
- Memory elements in asynchronous circuits are regarded as time-delay elements
- No clock



Synchronous (Clocked) Sequential Circuits

Signals affect the memory elements at discrete instants of time.

- requires synchronization.
- usually achieved through the use of a common clock.
- clock generator is a device that generates a periodic train of pulses:



Storage Elements

- Binary storage device capable of storing one bit
- Latch = *level-sensitive* device
  - State changes with input when enabled (e.g., when clock = 1)
  - Holds last input value when disabled (when clock = 0)
- Flip-flop = *edge-triggered* device
  - State of flip-flop can only change during clock *transition*
  - Example: Flip-flops change on rising/falling edge of clock

- Why change on an edge?
  - Couldn't we change state while clock is 1?
  - That would be a latch!
- Edge is moment in time, state is duration
  - Feedback would continue during clock being 1, causing possible race conditions

Level-sensitive vs Edge-triggered

- Latches are level-sensitive

(a) Response to positive level

- Flip-flops are edge-sensitive

(b) Positive-edge response

(c) Negative-edge response

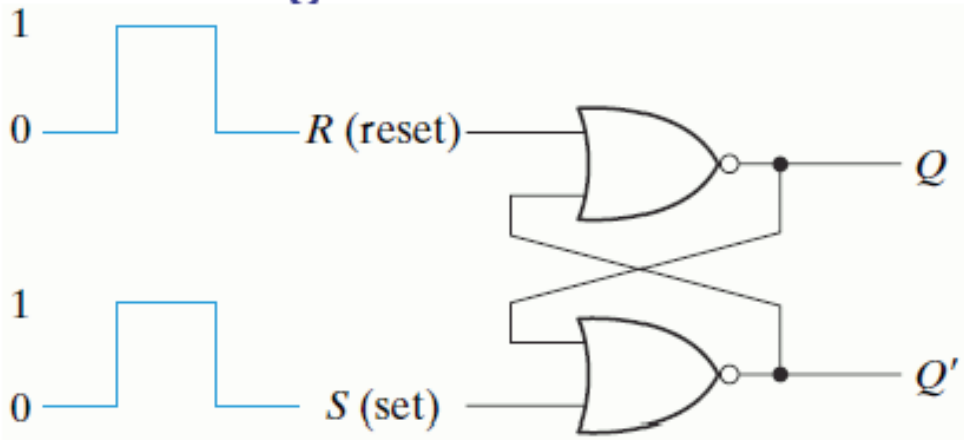
Latches

- Characteristics
  - Can store one bit of binary information
  - Level-sensitive devices, asynchronous

- SR Latch
  - Named after functionality: *S* = set, *R* = reset
  - Specification:
    - » Inputs: *S* and *R*
    - » Outputs: *Q* and *Q'*
  - Operation:
    - » *Q=1* and *Q'=0* when in set state
    - » *Q=0* and *Q'=1* when in reset state
    - » Inputs should be 0 unless pulse on *S* or *R* sets or resets latch

SR Latch

- Circuit diagram:

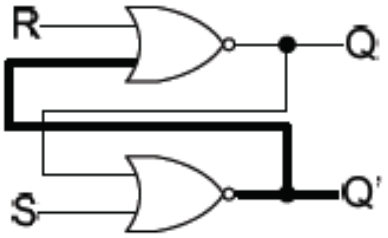


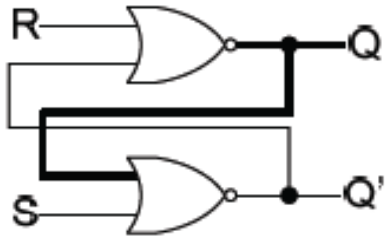
(a) Logic diagram

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	

(b) Function table

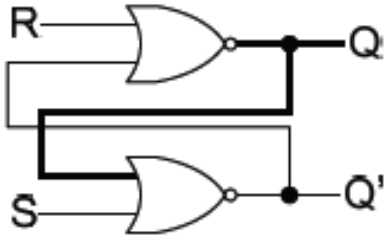
- *Set* and *Reset* are stable states
  - If *S=0* and *R=0*, then state will not change by itself





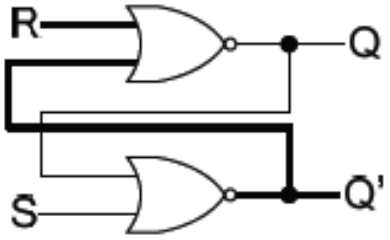
SR Latch - Operation

- Operation:



set state

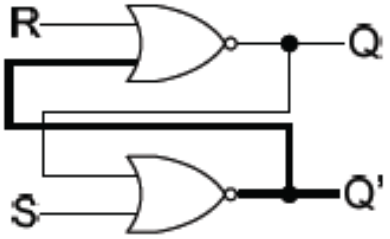
R,S=0
Q=1
Q′=0



reset active

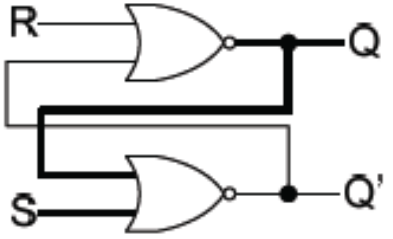
R=1
Q=0
Q′=1

- What happens if both *S*,*R* = 1?
  - Both NOR outputs become 0
  - Unstable state after releasing *S* and *R*



reset state

R,S=0
Q=0
Q′=1



set active

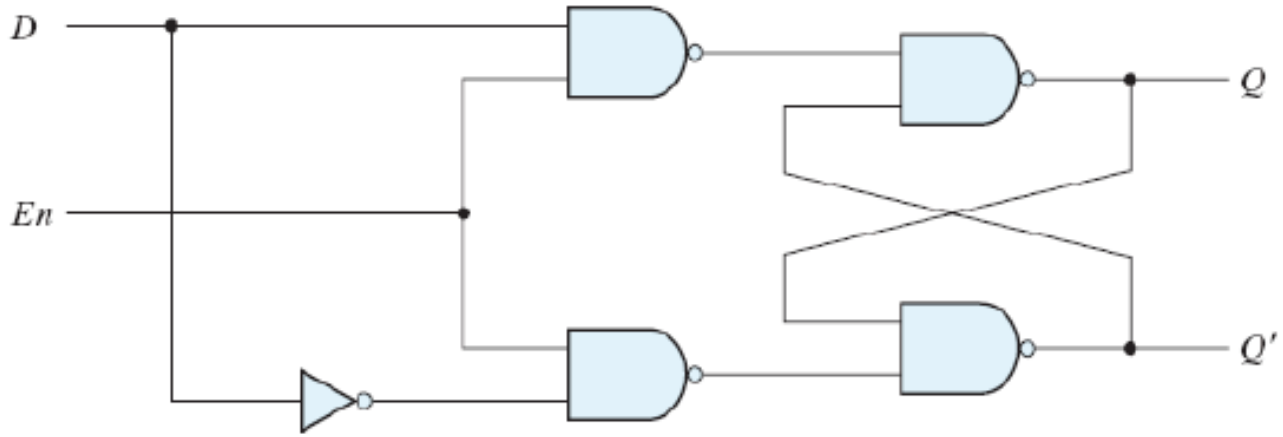
S=1
Q=1
Q′=0

D Latch

- How to remove state for *S=1*, *R=1* ?
- Solution
  - Just use one input pin *D* to indicate set or reset
  - Enable bit (*En*) ensures that latch is only set when intended

En	*D*	Next state of *Q*
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

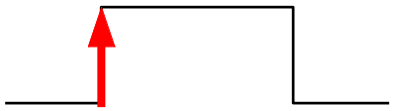
- D latch
  - Inputs:
    D (data)
    En (enable)
  - Circuit:



Edge-Triggered Flip-Flops

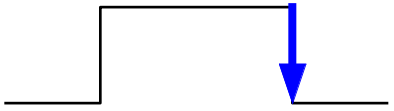
These circuits respond to their inputs on either the **rising** or **falling** edge of the clock — a precise point in time rather than an interval.

Positive edge triggered



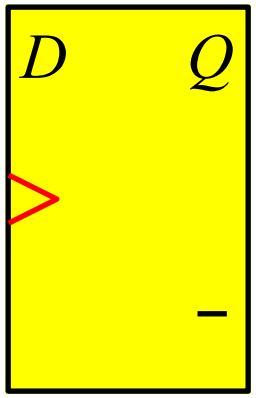
rising edge of clock

Negative edge triggered

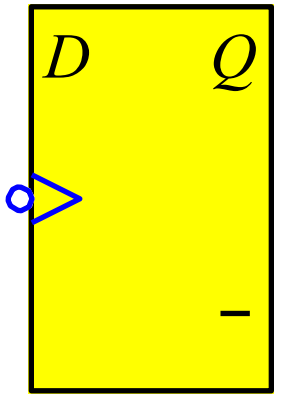


falling edge of clock

wedge shows positive edge triggering

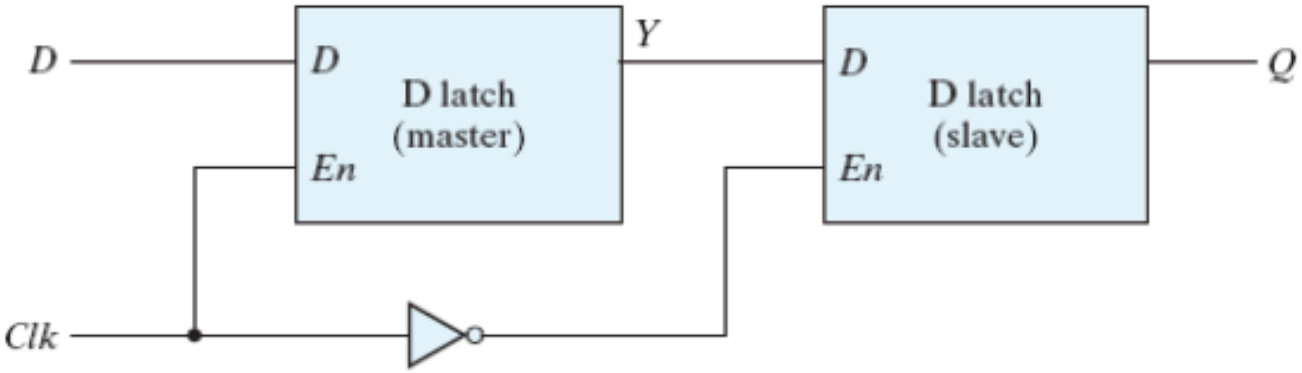


additional of a circle means that there is negative edge triggering



Edge-triggered D Flip-Flop

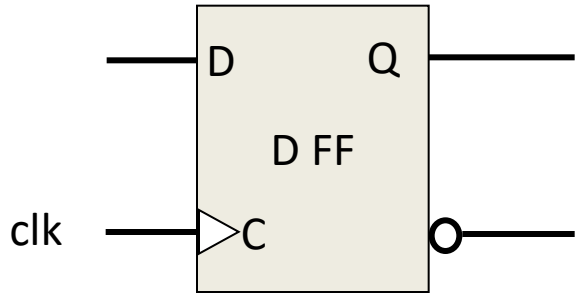
- Construct D flip-flop from two latches:



- Primary latch:
  - Reads value of *D* while *CLK* is high
  - Is disabled when clock is low

- Secondary latch:
  - Is disabled when *CLK* is high (i.e., holds previous value)
  - Takes value from master on negative edge of clock

D Flip-Flop



Positive edge-triggered
D Flip-Flop

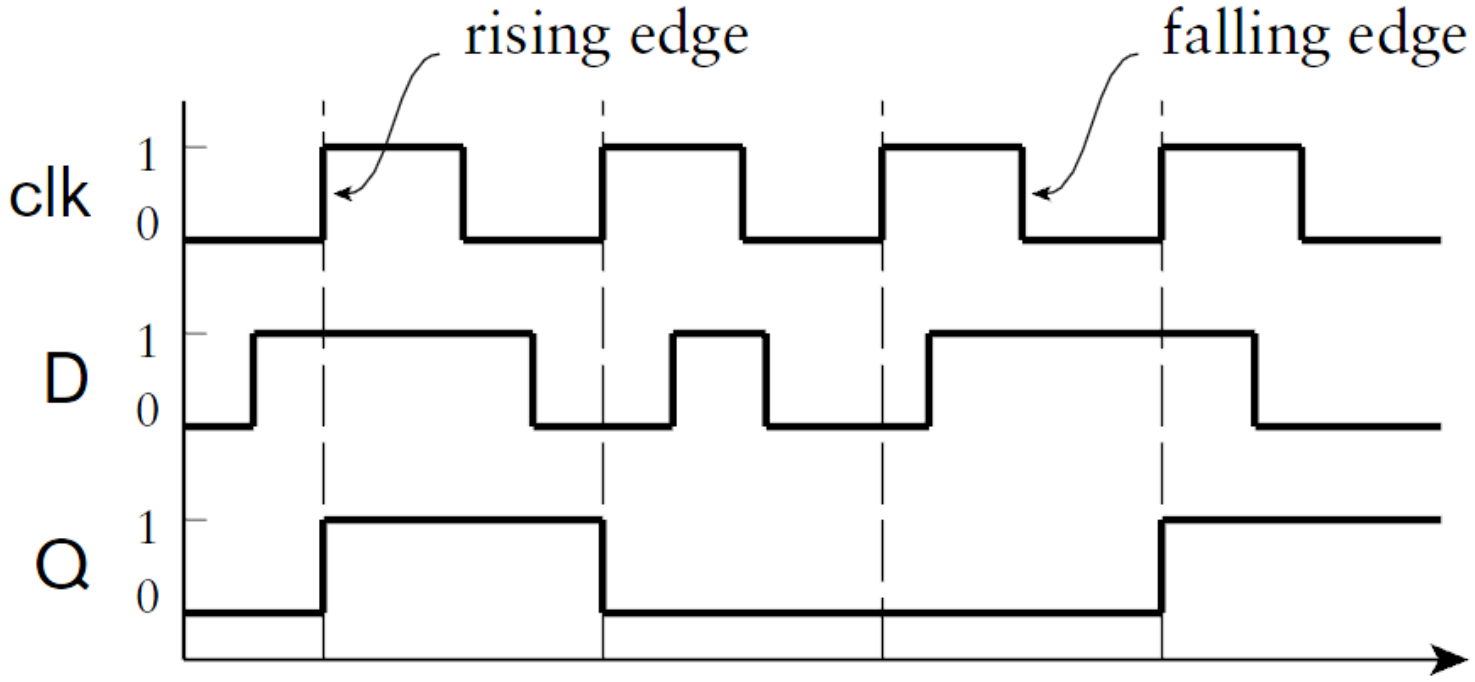
- **Characteristic equation**

  Q(t+1) = D

D	Q(t+1)
0	0
1	1

Characteristic Table

Timing Diagram of D Flip-Flop

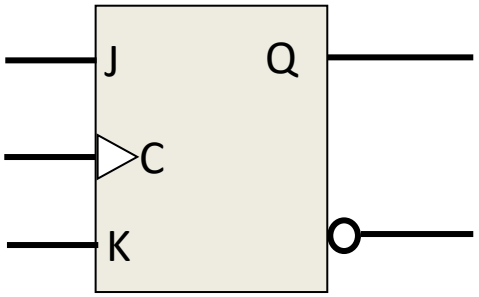


Other Flip-Flops

- D flip-flop is the most common
  - since it requires the fewest number of gates to construct

- Two other widely used flip-flops
  - JK flip-flops
  - T flip-flops

JK Flip-Flops

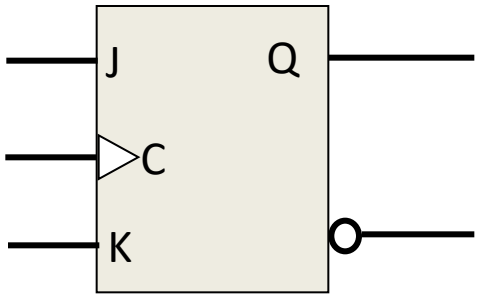


J	K	Q(t+1)	next state
0	0	Q(t)	no change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

Characteristic Table

Characteristic Equation of JK flip-flop?

JK Flip-Flops

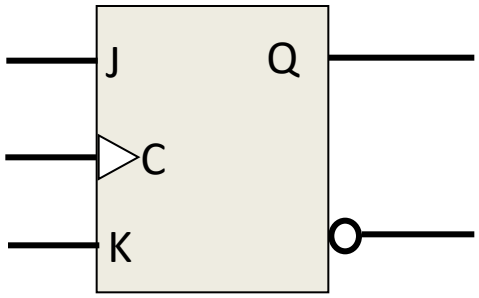


J	K	Q(t+1)	next state
0	0	Q(t)	no change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

Characteristic Table

J	K	Q(t)	Q(t+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	

JK Flip-Flops



J	K	Q(t+1)	next state
0	0	Q(t)	no change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

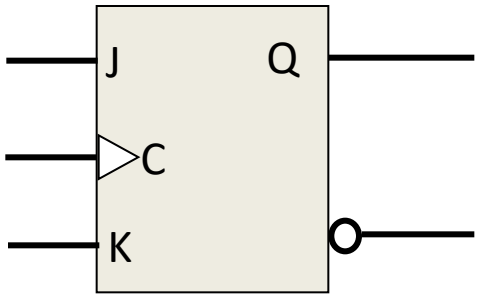
J	K	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Characteristic Table

J \ K,Q(t)	00	01	11	10
0	0	1	0	0
1	1	1	0	1

Q(t+1) = ?

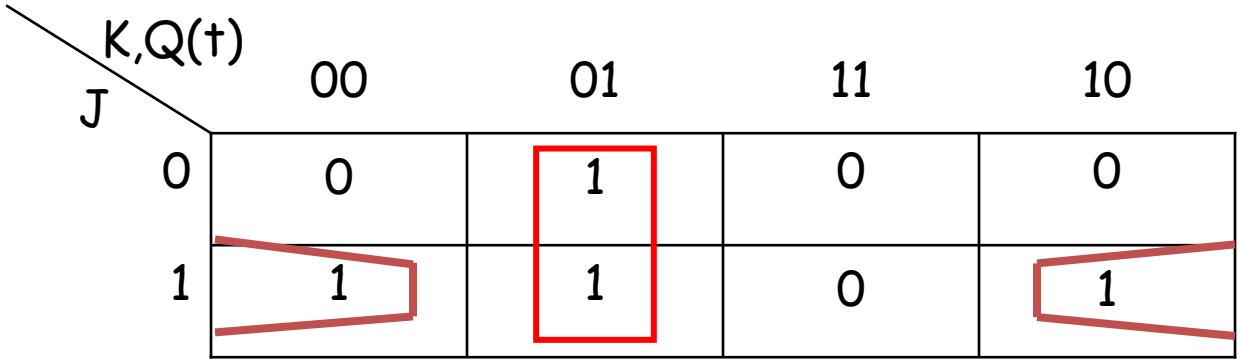
JK Flip-Flops



J	K	Q(t+1)	next state
0	0	Q(t)	no change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

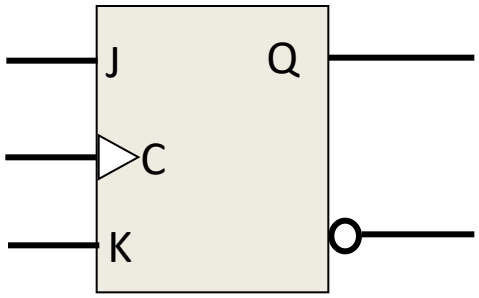
Characteristic Table

J	K	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



Q(t+1) = ?

JK Flip-Flops



J	K	Q(t+1)	next state
0	0	Q(t)	no change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

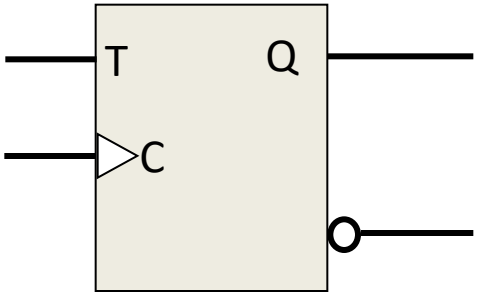
Characteristic Table

- Characteristic equation

  $Q(t+1) = JQ'(t) + K'Q(t)$

T (Toggle) Flip-Flop

- Complementing flip-flop

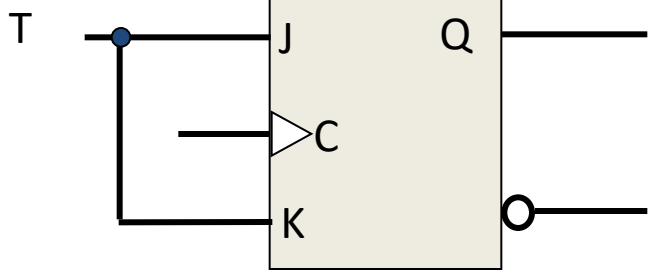


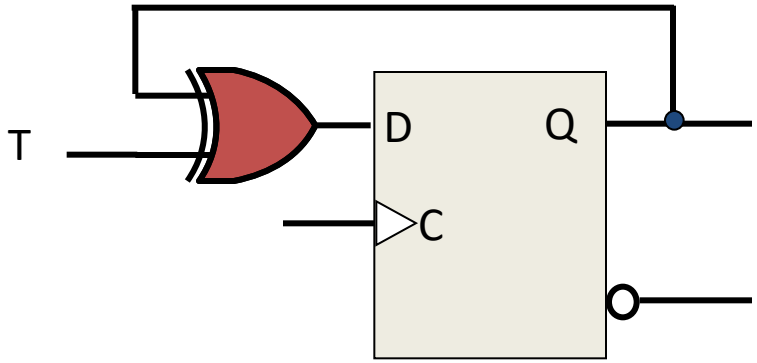
T	Q(t+1)	next state
0	Q(t)	no change
1	Q'(t)	Complement

Characteristic Table

- Characteristic equation

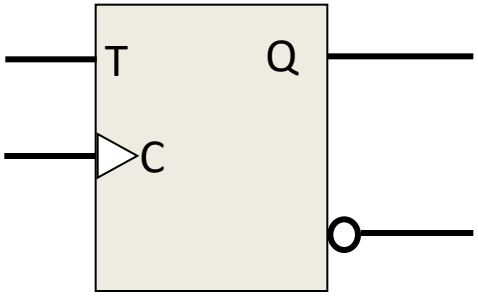
Q(t+1) = ?





T (Toggle) Flip-Flop

- Complementing flip-flop

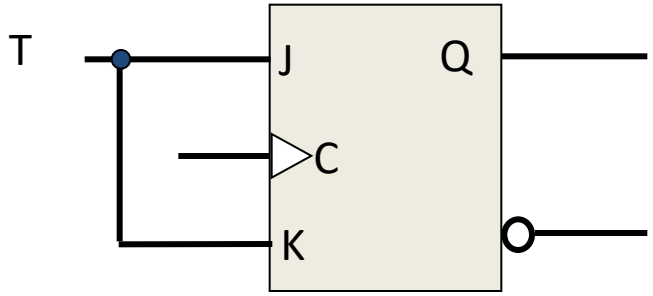


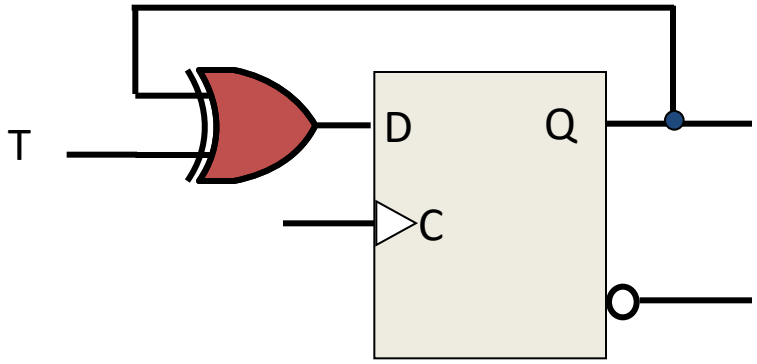
T	Q(t+1)	next state
0	Q(t)	no change
1	Q'(t)	Complement

Characteristic Table

- Characteristic equation

$$Q(t+1) = Q(t) \oplus T$$





Characteristic Equations

The logical properties of a flip-flop can be expressed algebraically using characteristic equations :

- D flip-flop

  $Q(t+1) = D$

- JK flip-flop

  $Q(t+1) = JQ'(t) + K'Q(t)$

- T flip-flop

  $Q(t+1) = Q(t) \oplus T$

What if we have Q(t+1) and Q(t), and looking for J and K values?

Q(t)Q(t+1)	00	01	11	10
J,K	0,X	1,X	X,0	X,1

J= { Q(t+1) if Q(t)=0; X if Q(t)=1

K= { Q(t+1)' if Q(t)=1; X if Q(t)=0

What if we have Q(t+1) and Q(t), and looking for D value?

Q(t)Q(t+1)	00	01	11	10
D	0	1	1	0

D= Q(t+1)

What if we have Q(t+1) and Q(t), and looking for T value?

Q(t)Q(t+1)	00	01	11	10
T	0	1	0	1

$T = Q(t) \oplus Q(t+1)$

Machine

Machine (

Inputs {X},

States {D},

Outputs {Z},

Output Function {F : $X \times D \rightarrow Z$},

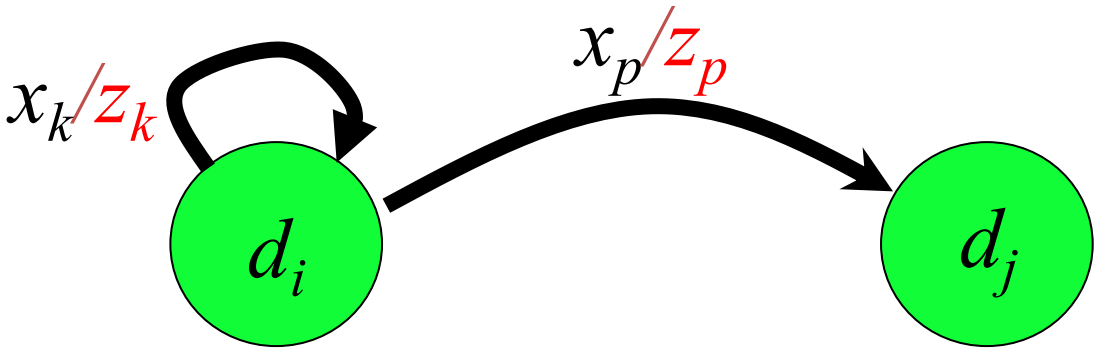
Next State Function {G : $X \times D \rightarrow D$}

)

Representation with **State Diagram**

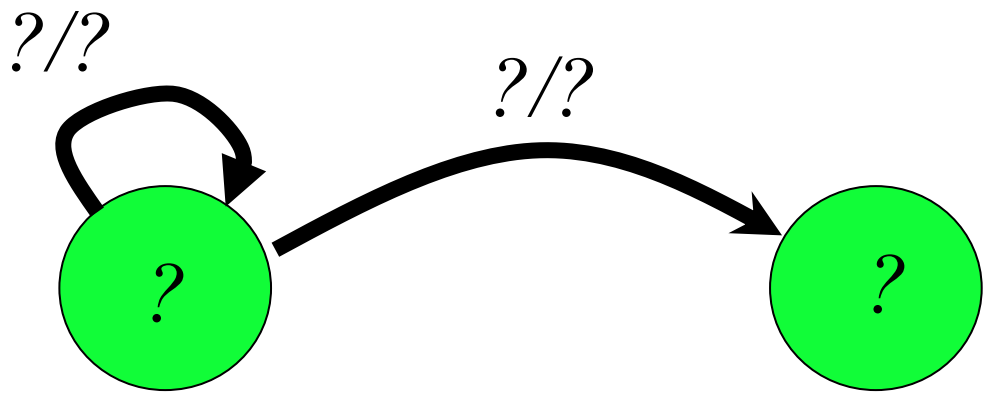
	x_1	x_2	…	x_i	…	x_l
d_1						
d_2						
⋮						
d_i				d_m, z		
⋮						
d_{r-1}						
d_r						

Assign a Node to Each State



- Machine is at state d_i, input x_k comes, the next state will be again d_i and the output is z_k
- Machine is at state d_i, input x_p comes, the next state will be d_j and the output is z_p

Assign a Node to Each State



- Machine is at state d_m , input x_t comes, the next state will be again d_m and the output is z_t
- Machine is at state d_m , input x_u comes, the next state will be d_n and the output is z_u

Notation

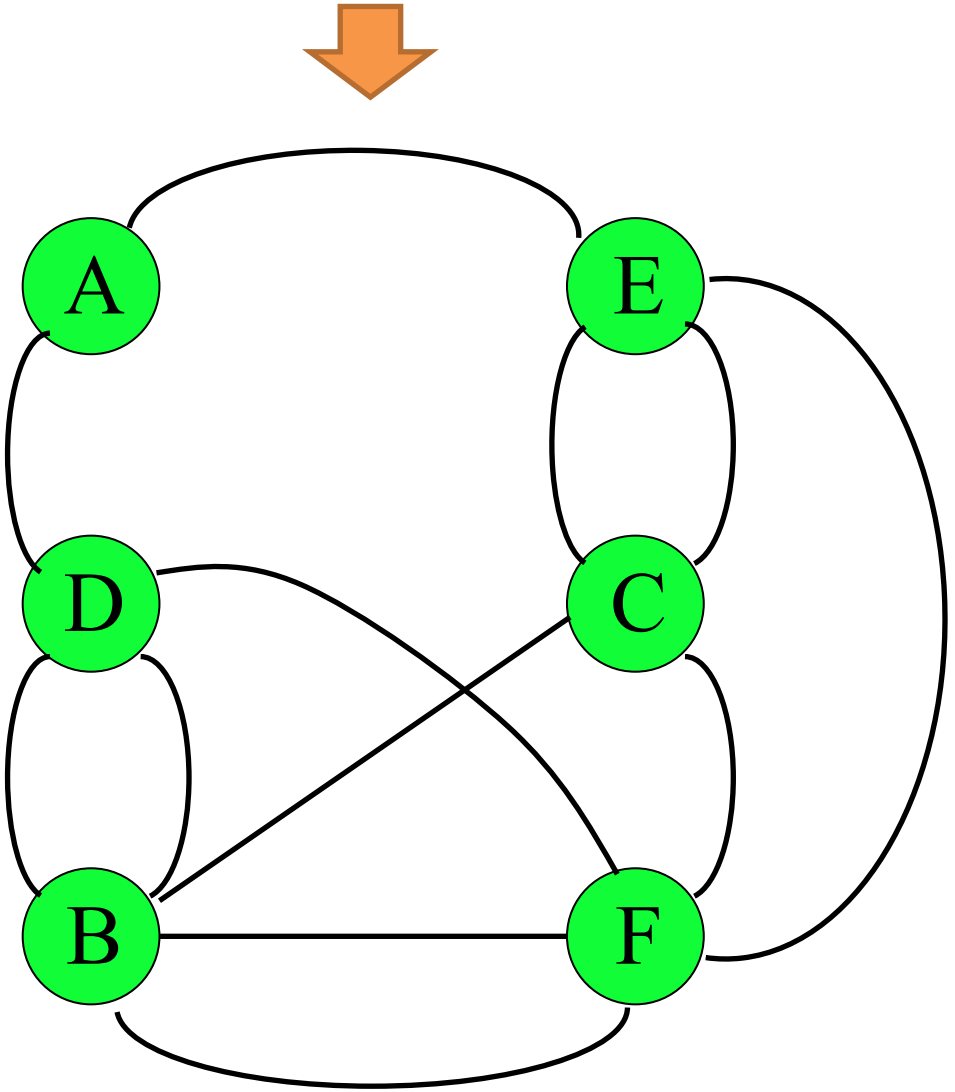
- Let $\mathbf{I_k}$ be an **input sequence** with length k,

i.e. $I_k = x_1x_2...x_k$

- $\mathbf{f(I_k,d_i)} = z_1z_2...z_k$ is an **output sequence**
- $\mathbf{g(I_k,d_i)} = di_1di_2...di_k$ is a **state sequence**
- $d_i \xrightarrow{I_k} \mathbf{di_k}$: **Follower of d_i** after the input sequence I_k

Example - Fill out the rest

	0	1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0



Example

	0	1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

- Let I_5=10110, find state sequence g(I_5,C) and output sequence f(I_5,C)

I_5						
g(I_5,C)						
f(I_5,C)						

- I_5 follower of C : ?

Example

	0	1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

- Let I_5=10110, find state sequence g(I_5,C) and output sequence f(I_5,C)

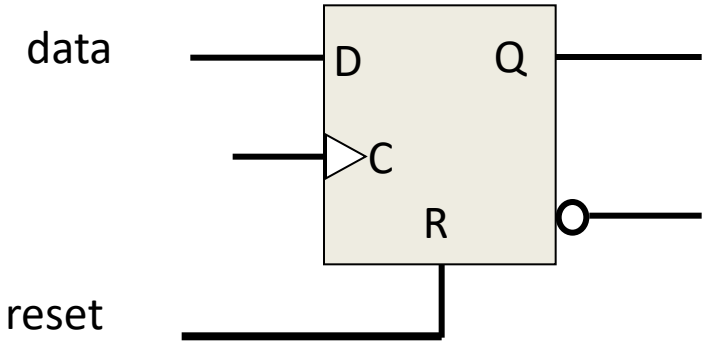
I_5		1	0	1	1	0
g(I_5,C)	C	B	F	C	B	F
f(I_5,C)		1	0	0	1	0

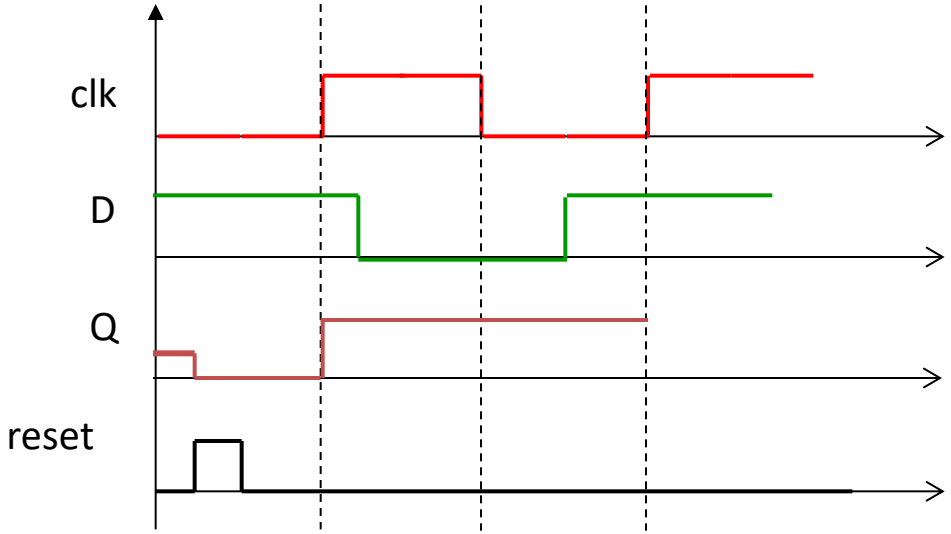
- I_5 follower of C is F

Asynchronous Inputs of Flip-Flops

- They are used to force the flip-flop to a particular state independent of clock
  - "Preset" (direct set) set FF state to 1
  - "Clear" (direct reset) set FF state to 0
- They are especially useful at startup.
  - In digital circuits when the power is turned on, the state of flip-flops are unknown.
  - Asynchronous inputs are used to bring all flip-flops to a known "starting" state prior to clock operation.

Asynchronous Inputs



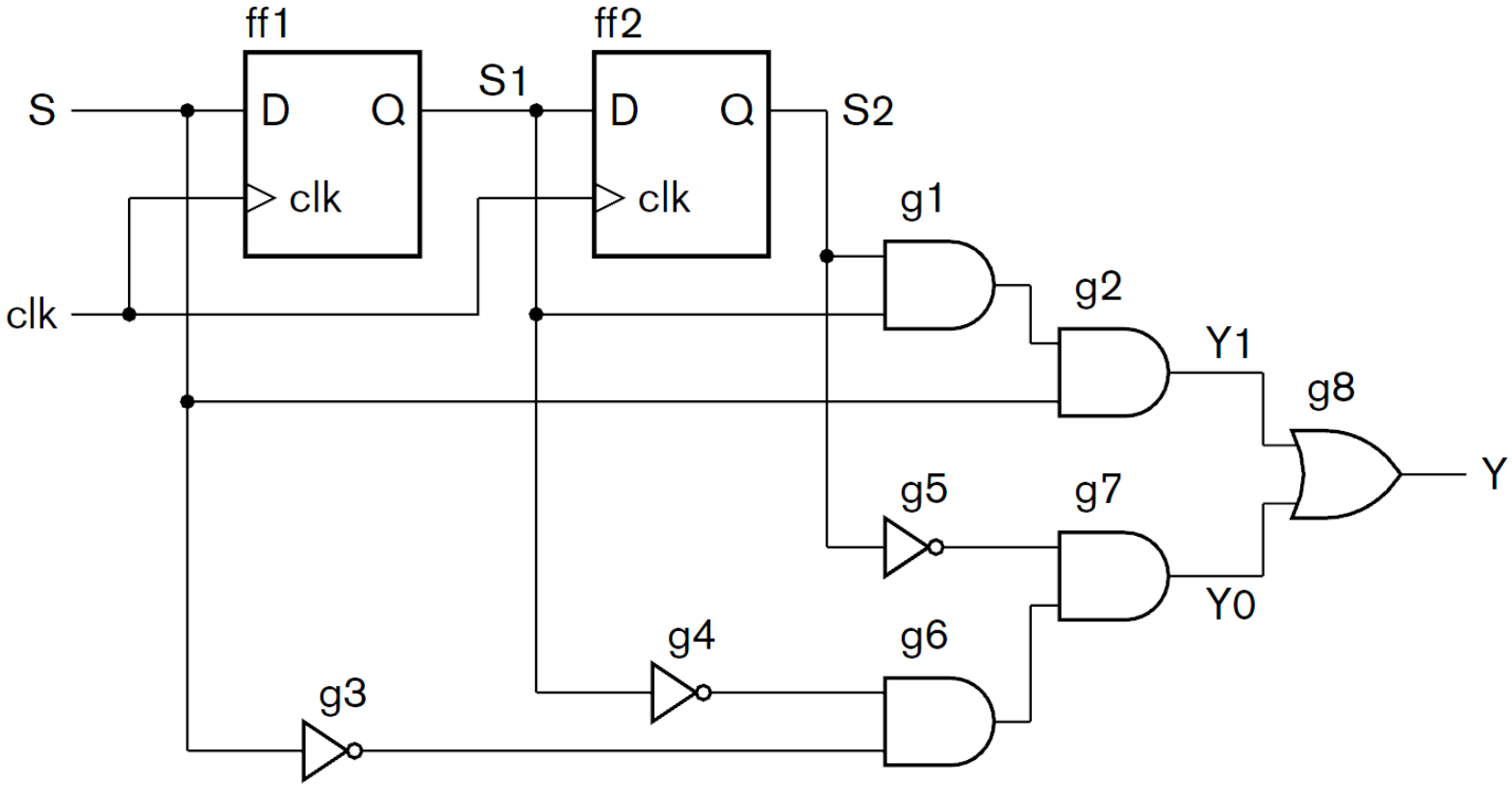


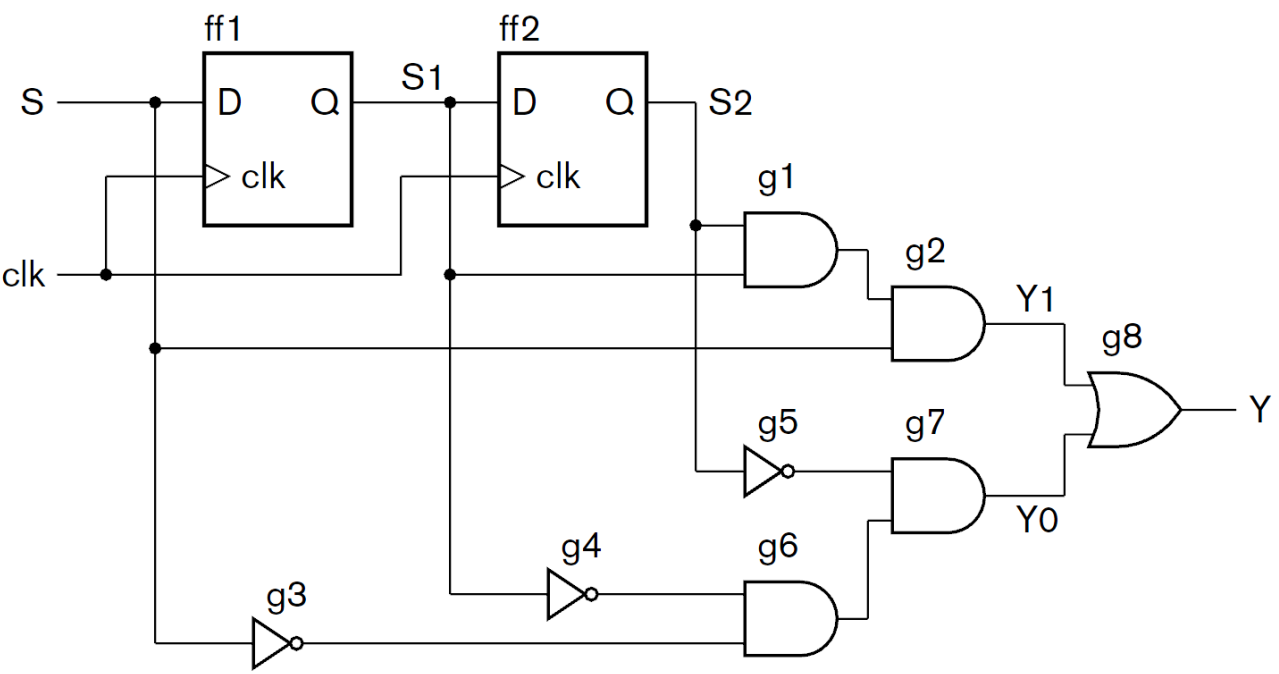
reset	C	D	Q	Q′	
1	X	X	0	1	Starting State
0	↑	0	0	1	
0	↑	1	1	0	

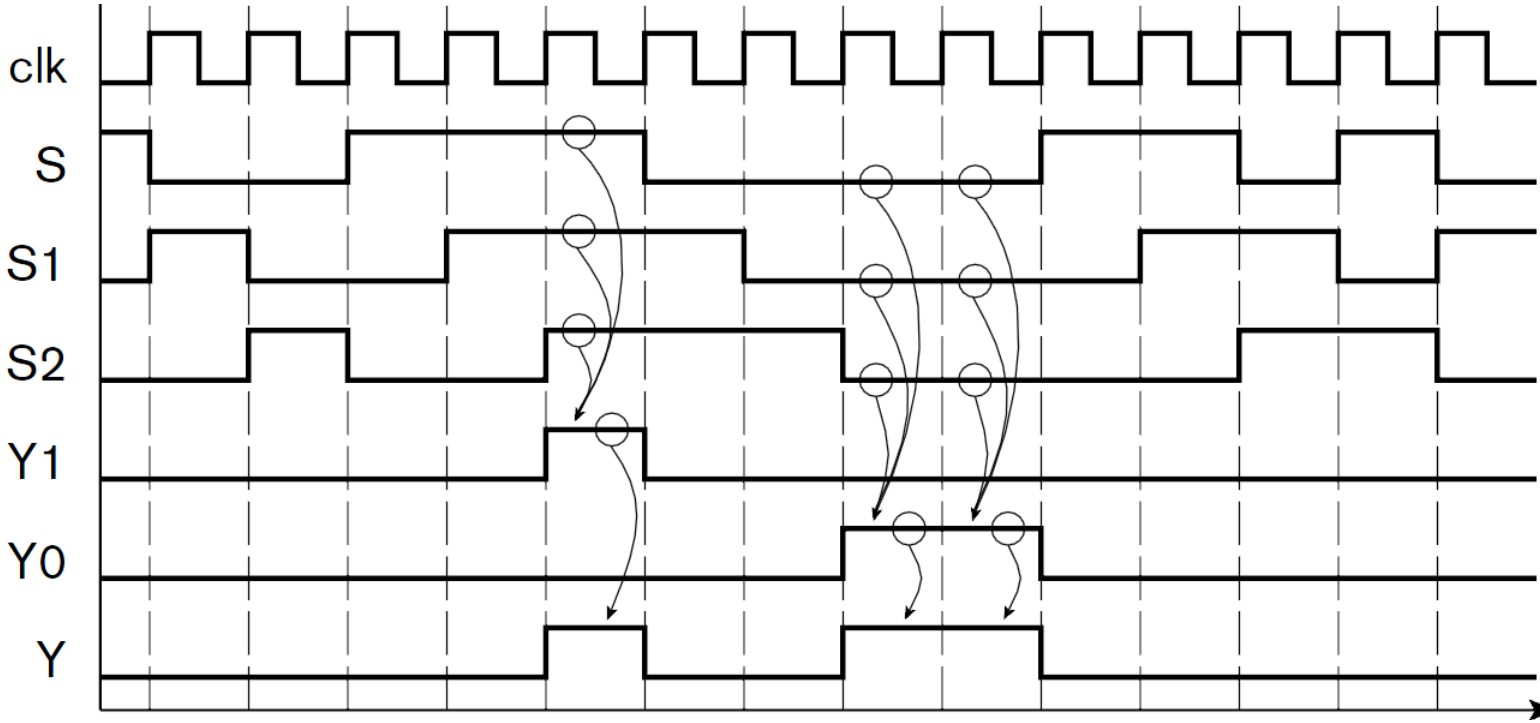
Analysis of Clocked Sequential Circuits

- Goal:
  - to determine the behavior of clocked sequential circuits
  - "Behavior" is determined from
    - Inputs
    - Outputs
    - State of the flip-flops
  - We have to obtain
    - Boolean expressions for output and next state
      - output & state equations
    - (state) table
    - (state) diagram

Analyze the circuit

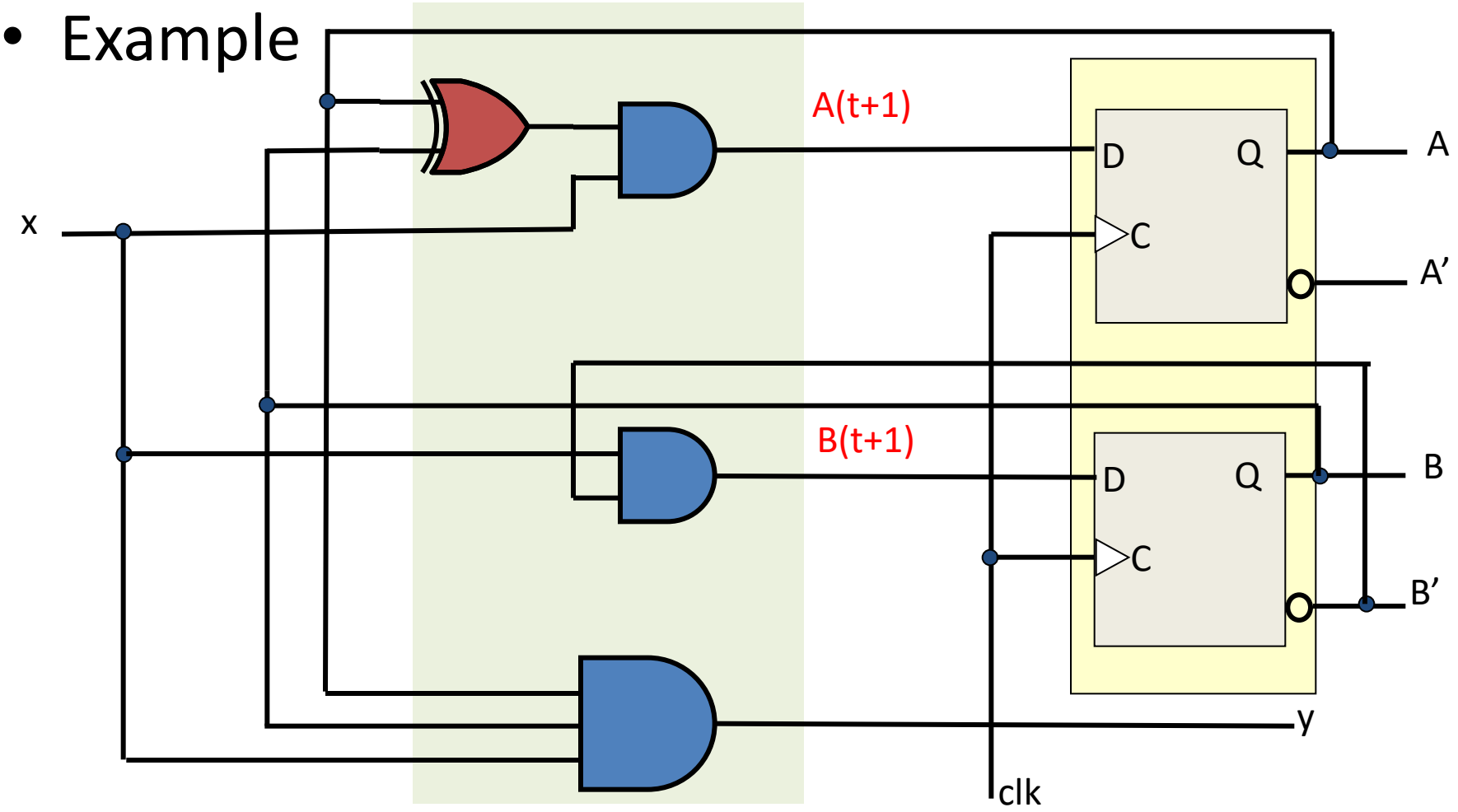



ff1
ff2
S
D
Q
clk
S1
S2
clk
g1
g2
Y1
g8
Y
g5
g7
Y0
g4
g6
g3


clk
S
S1
S2
Y1
Y0
Y

State Equations

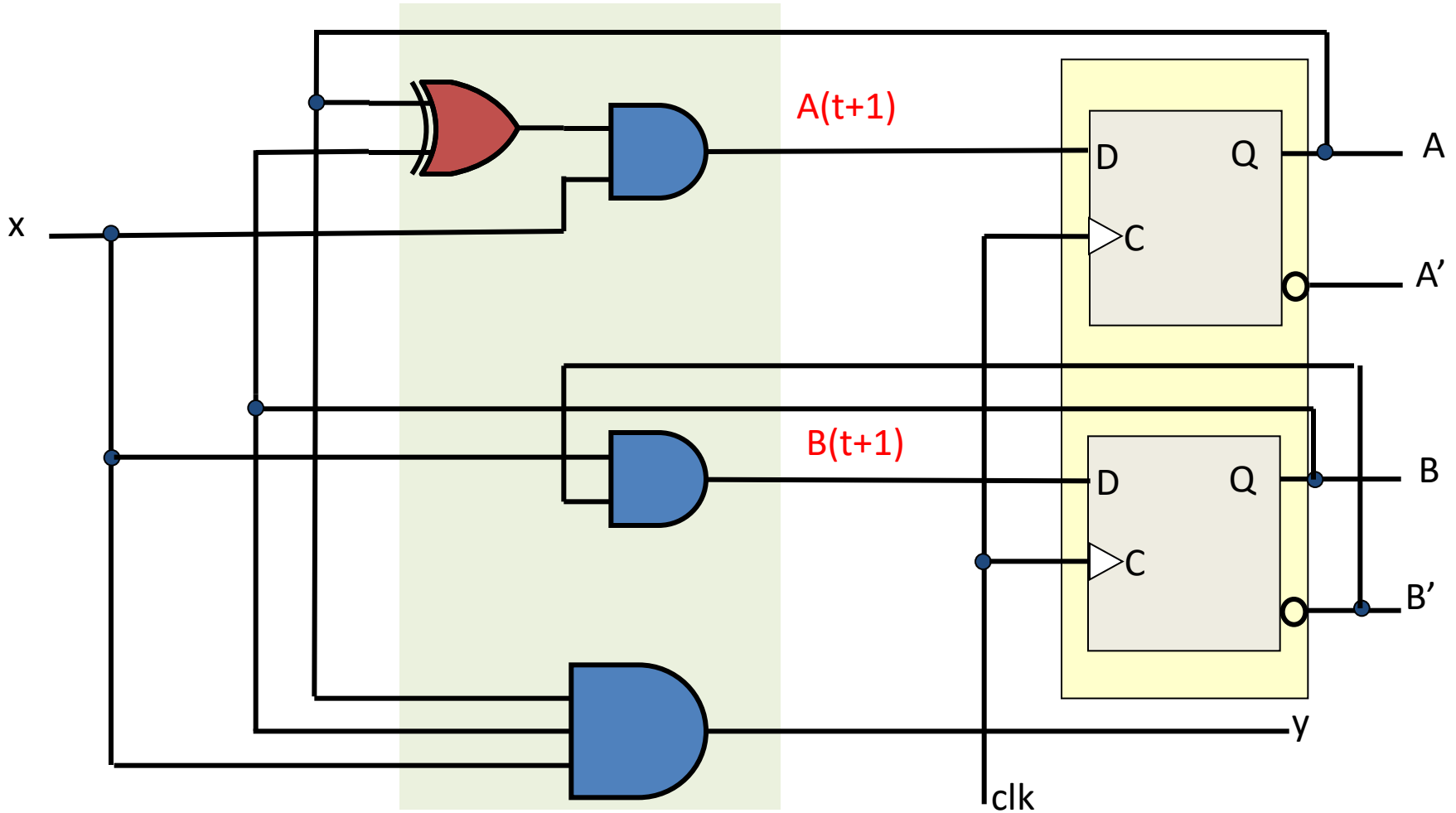
- Also known as “transition equations”
  - specify the next state as a function of the present state and inputs
- Example



Output and State Equations

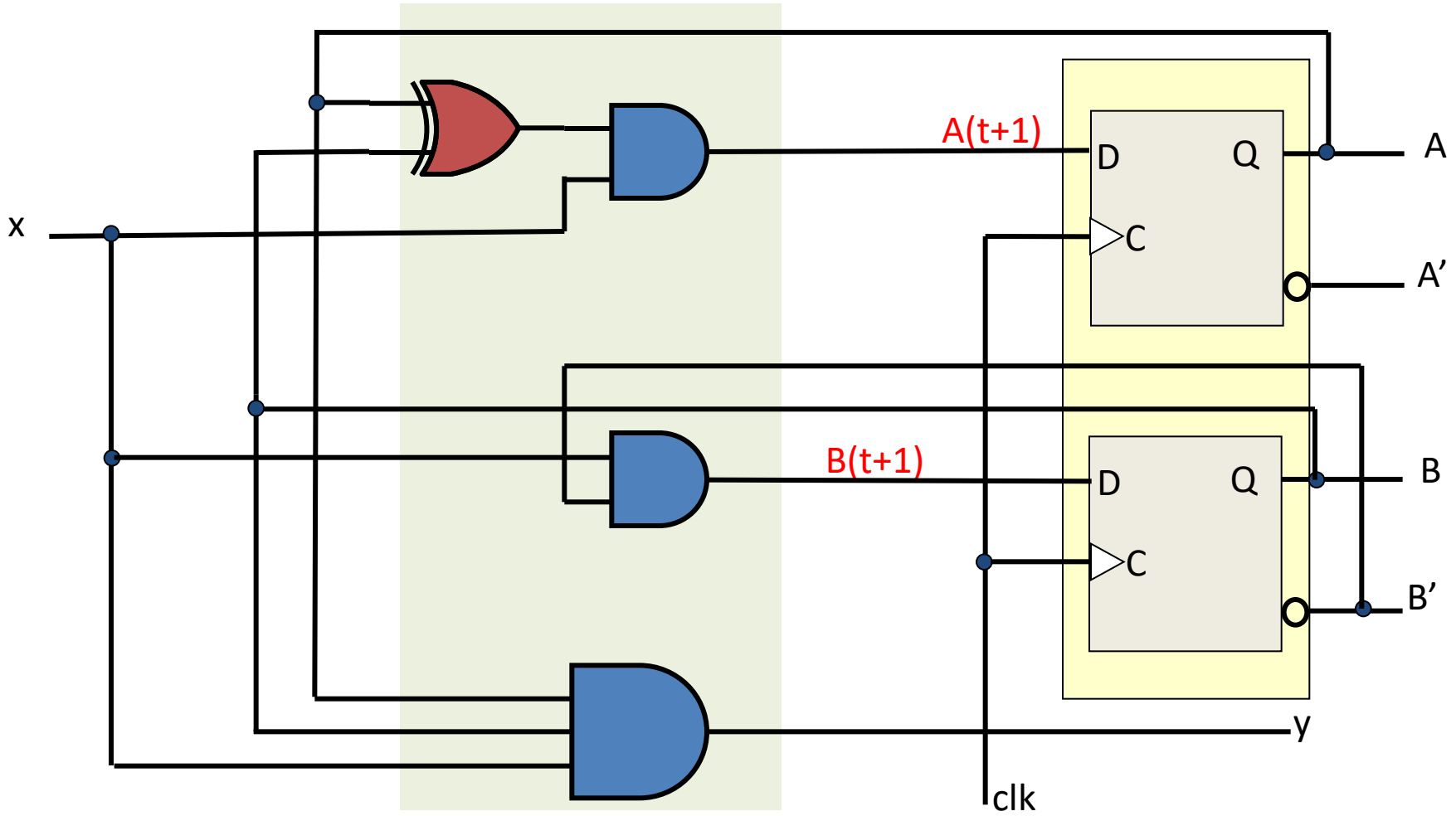
- A(t+1) =
- B(t+1) =
- y =

input of a flip-flop determines the value of the next state (i.e., the state reached after the clock transition)



Output and State Equations

- $A(t+1) = (A(t) \oplus B(t))\, x$
- $B(t+1) = (B(t))'\, x$
- $y = A(t)B(t)\, x$



Flip Flop Input Equations

- Flip-Flop input (excitation) equations
- Same as the state equations in D flip-flops

Example: State (Transition) Table

$A(t+1) = (AB' + A'B) x$ $B(t+1) = B'x$ $y = ABx$

Present state		input	Next state		output
A(t)	B(t)	x	A(t+1)	B(t+1)	y
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

A sequential circuit with **m** FFs and **n** inputs needs $\mathbf{2^{m+n}}$ rows in the transition table

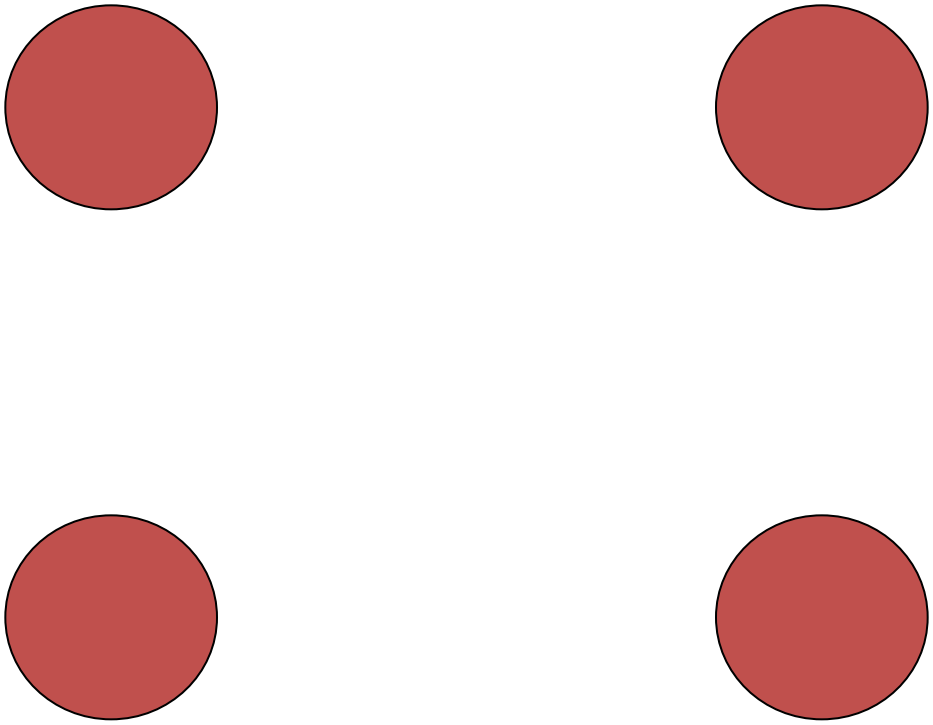
Example: State (Transition) Table

$A(t+1) = (AB' + A'B) x$ $B(t+1) = B'x$ $y = ABx$

Present state		input	Next state		output
A(t)	B(t)	x	A(t+1)	B(t+1)	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	1

A sequential circuit with **m** FFs and **n** inputs needs $\mathbf{2^{m+n}}$ rows in the transition table

Example: State Diagram

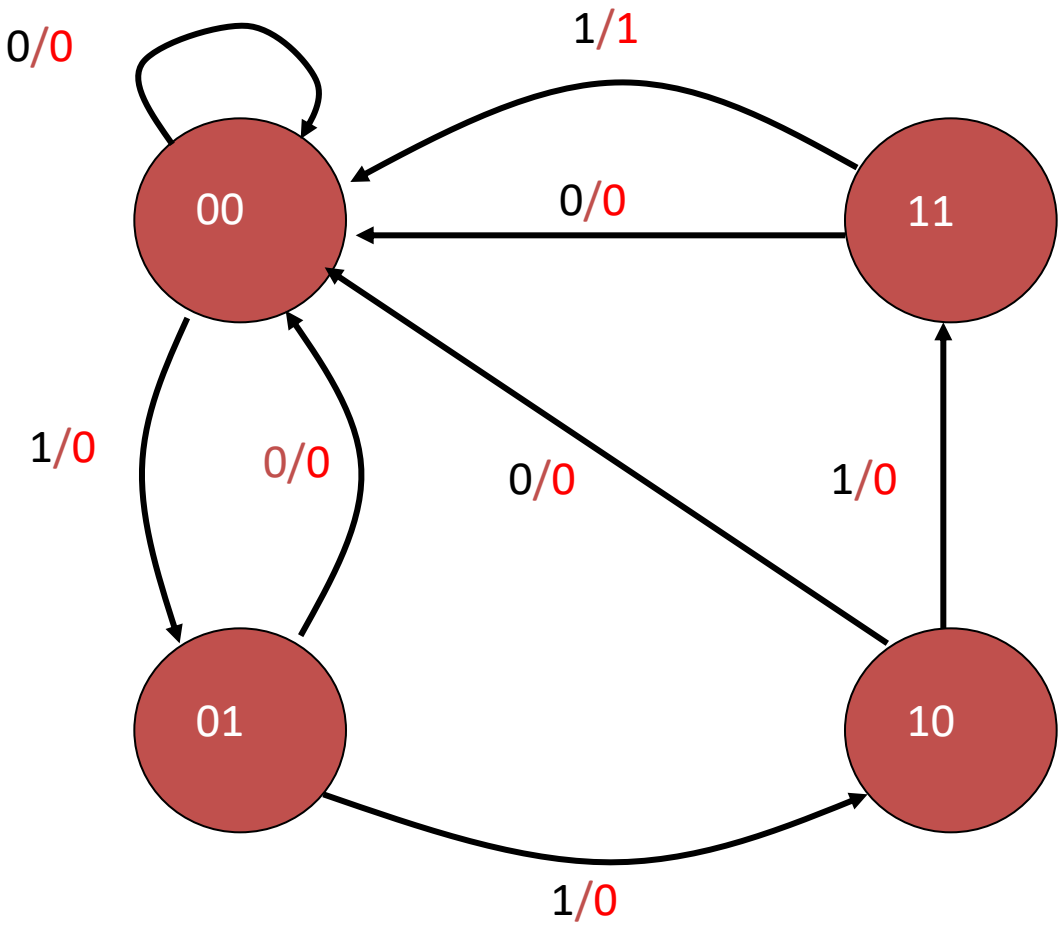


Present state		input	Next state		output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	1

What is this circuit doing?

State diagram provides the same information as state table

Example: State Diagram



Present state		input	Next state		output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	1

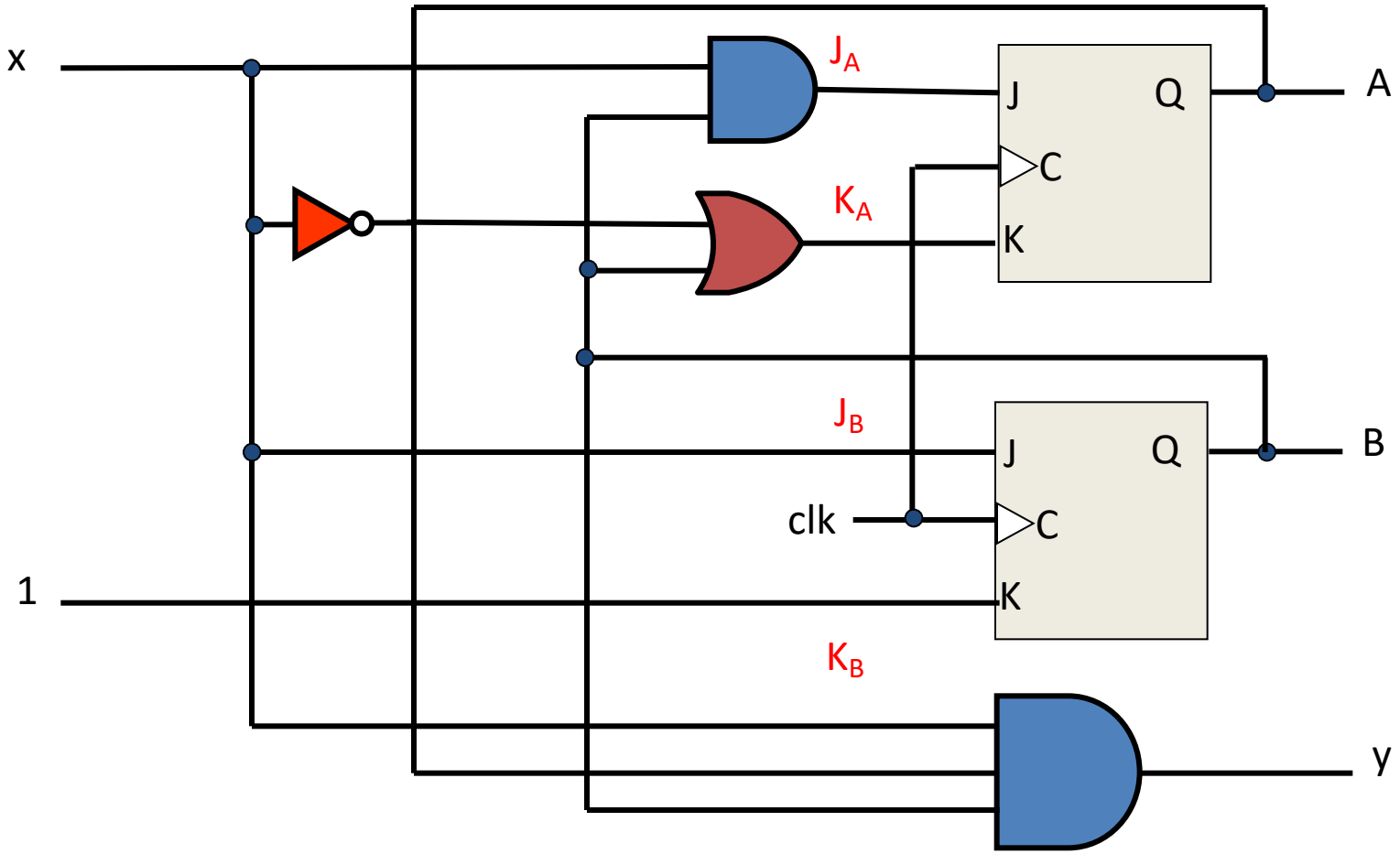
What is this circuit doing?

State diagram provides the same information as state table

Analysis with JK Flip-Flops

- For a D flip-flop, the state equation is the same as the flip-flop input equation
  - $Q(t+1) = D$
- For JK flip-flops, situation is different
  - Goal is to find state equations
  - Method
    1. Determine flip-flop input equations
    2. List the binary values of each input equation
    3. Use the corresponding flip-flop characteristic table to determine the next state values in the state table

Example: Analysis with JK FFs



- Flip-flop input equations

  J_A = and K_A =

  J_B = and K_B =

Example: Analysis with JK FFs

- $J_A = Bx$ and $K_A = x'+B$
- $J_B = x$ and $K_B = 1$

present State		input	next state		FF inputs			
A(t)	B(t)	x	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0						
0	0	1						
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1						

Example: Analysis with JK FFs

- $J_A = Bx$ and $K_A = x'+B$
- $J_B = x$ and $K_B = 1$

present State		input	next state		FF inputs			
A(t)	B(t)	x	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0			0	1	0	1
0	0	1			0	0	1	1
0	1	0			0	1	0	1
0	1	1			1	1	1	1
1	0	0			0	1	0	1
1	0	1			0	0	1	1
1	1	0			0	1	0	1
1	1	1			1	1	1	1

Example: Analysis with JK FFs

- $J_A = Bx$ and $K_A = x'+B$
- $J_B = x$ and $K_B = 1$

present State		input	next state		FF inputs			
A(t)	B(t)	x	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	0	0	0	1	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	0	0	1	0	1
0	1	1	1	0	1	1	1	1
1	0	0	0	0	0	1	0	1
1	0	1	1	1	0	0	1	1
1	1	0	0	0	0	1	0	1
1	1	1	0	0	1	1	1	1

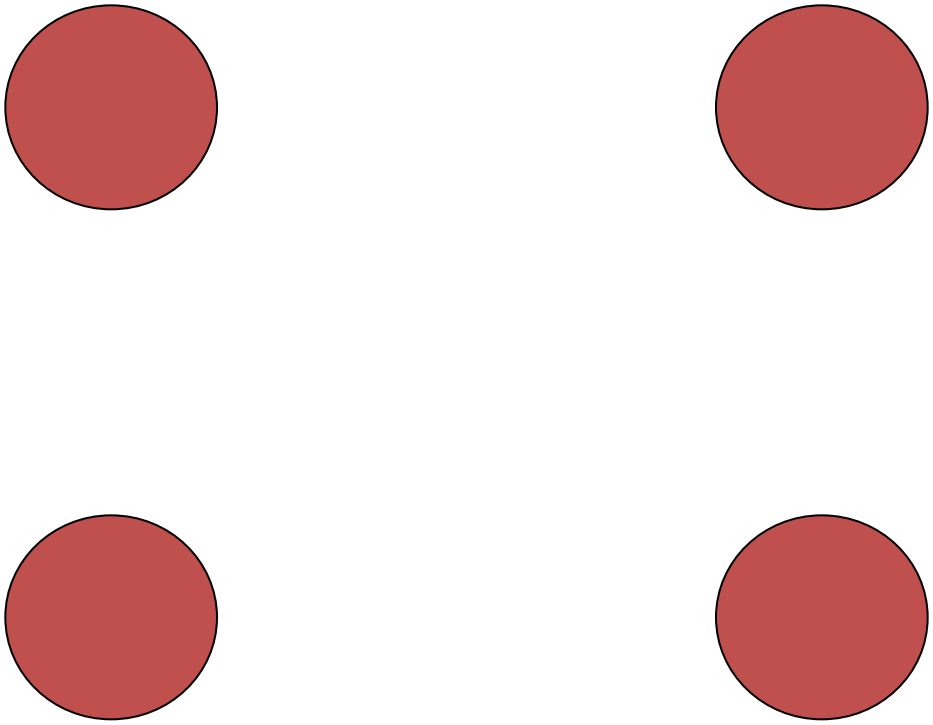
Example: Analysis with JK FFs

- Characteristic equations
  - $A(t+1) = J_A A' + K'_A A$
  - $B(t+1) = J_B B' + K'_B B$
  - $y = ABx$
- Input equations
  - $J_A = Bx$ and $K_A = x'+B$
  - $J_B = x$ and $K_B = 1$
- State equations
  - $A(t+1) =$
    $=$
  - $B(t+1) =$

Example: Analysis with JK FFs

- Characteristic equations
  - $A(t+1) = J_A A' + K'_A A$
  - $B(t+1) = J_B B' + K'_B B$
  - $y = ABx$
- Input equations
  - $J_A = Bx$ and $K_A = x'+B$
  - $J_B = x$ and $K_B = 1$
- State equations
  - $A(t+1) = A'Bx + (x'+B)' A$
    $= A'Bx + AB'x = (A \oplus B)\, x$
  - $B(t+1) = B'x$

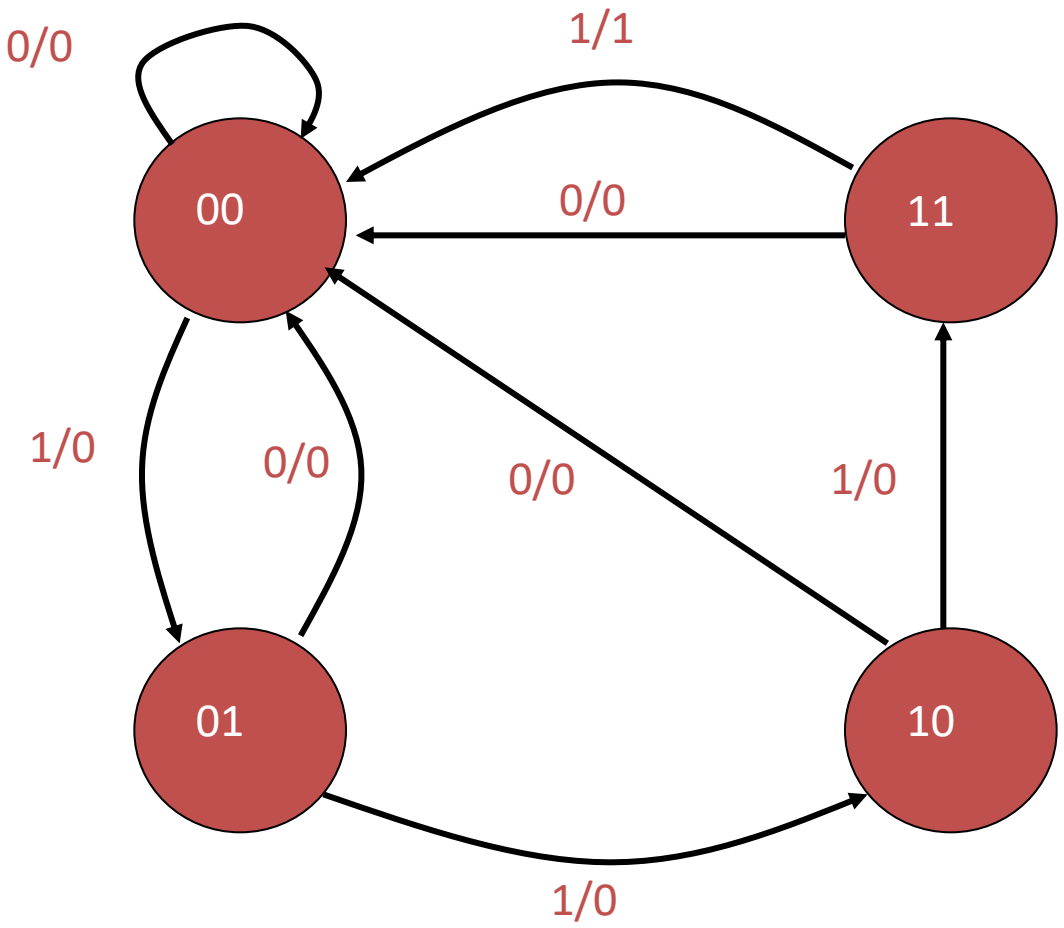
State Diagram



Present state		input	Next state		output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	1

What is the circuit doing?

State Diagram



Present state		input	Next state		output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	0	0	1

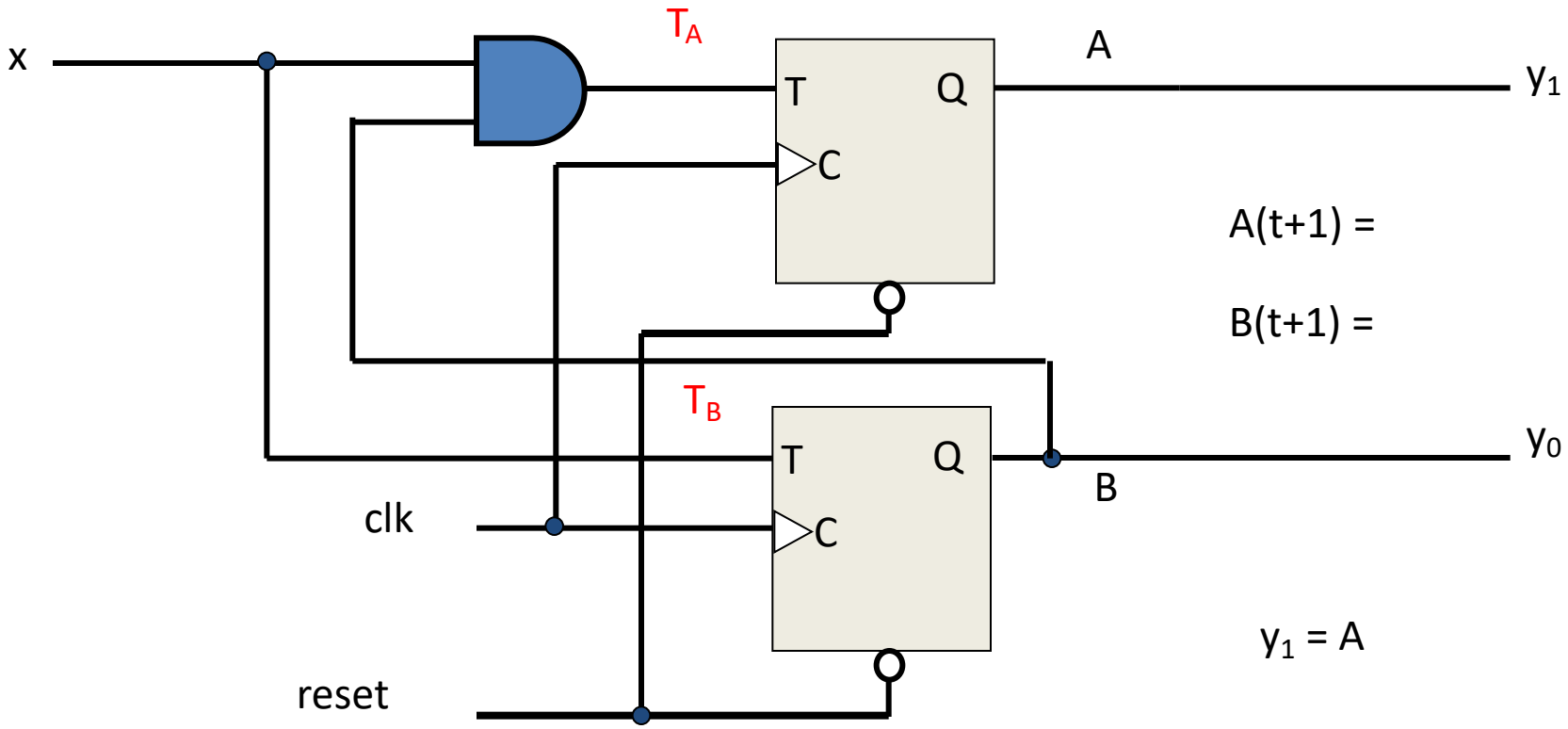
What is the circuit doing?

Analysis with T Flip-Flops

- Method is the same
- Example

$T_A =$

$T_B =$



$A(t+1) =$

$B(t+1) =$

$y_1 = A$

$y_0 = B$

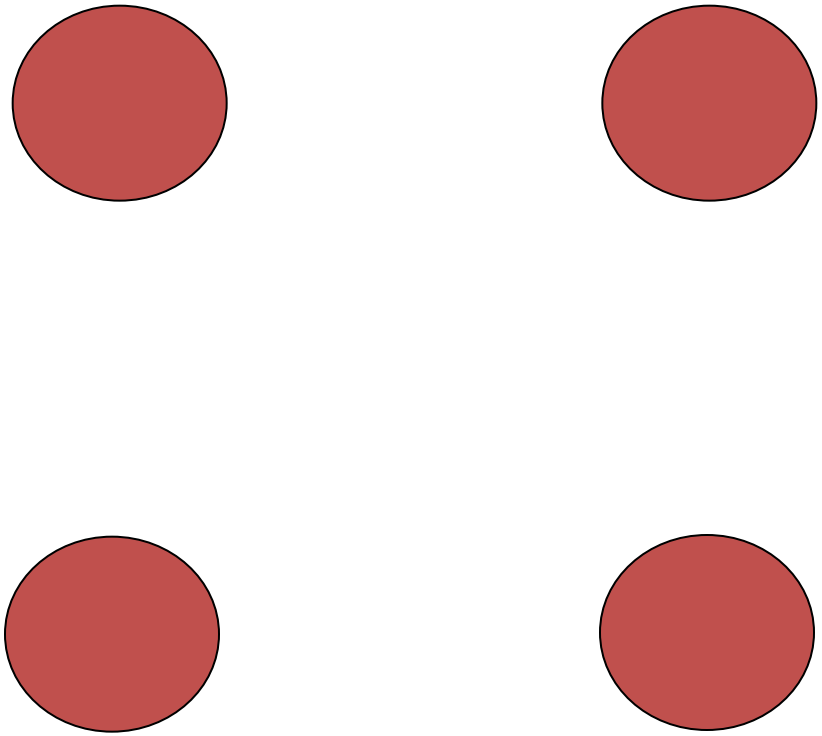
Example: Analysis with T Flip-Flops

- Characteristic equation
  - $A(t+1) = T_A \oplus A$
  - $B(t+1) = T_B \oplus B$
- Input equations
  - $T_A = Bx$
  - $T_B = x$
- Output equations
  - $y_1 = A$
  - $y_0 = B$
- State equations
  - $A(t+1) =$
  - $B(t+1) =$

State Table & Diagram

- $A(t+1) = Bx \oplus A$
- $B(t+1) = x \oplus B$
- $y_1 = A$; $y_0 = B$

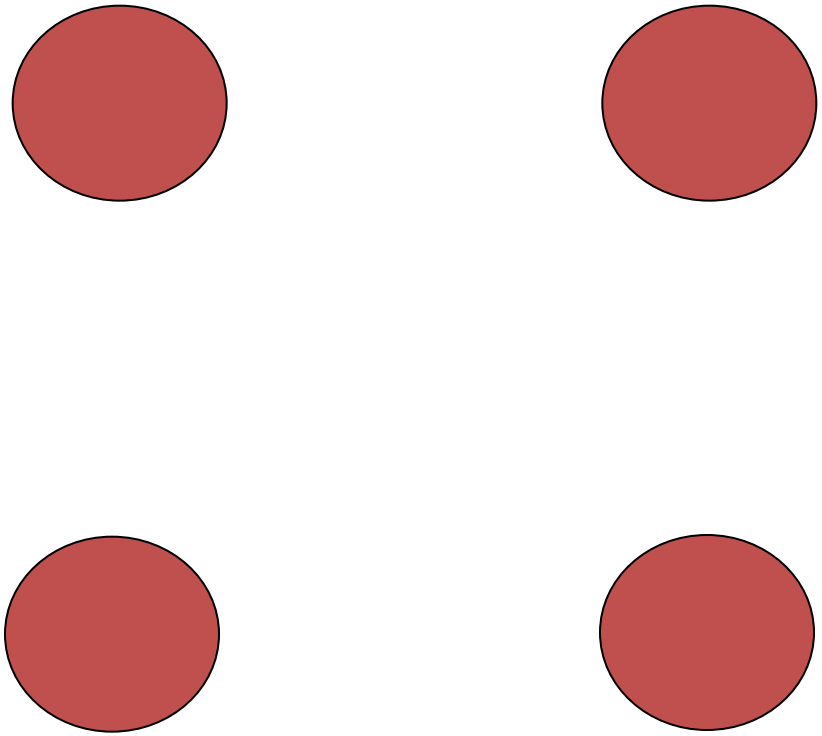
Present state		input	Next state		output	
A	B	x	A	B	y_1	y_0
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				



State Table & Diagram

- $A(t+1) = xB \oplus A$
- $B(t+1) = x \oplus B$
- $y_1 = A$; $y_0 = B$

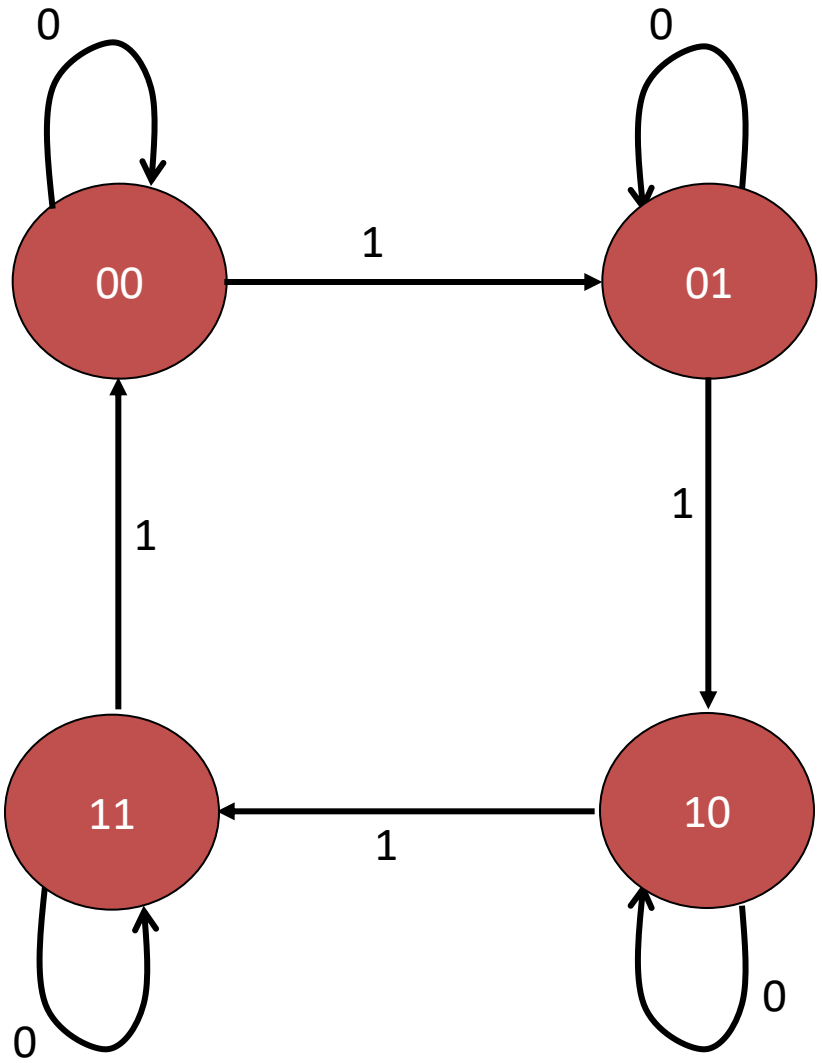
Present state		input	Next state		output	
A	B	x	A	B	y_1	y_0
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	0	1	0	1
0	1	1	1	0	0	1
1	0	0	1	0	1	0
1	0	1	1	1	1	0
1	1	0	1	1	1	1
1	1	1	0	0	1	1



State Table & Diagram

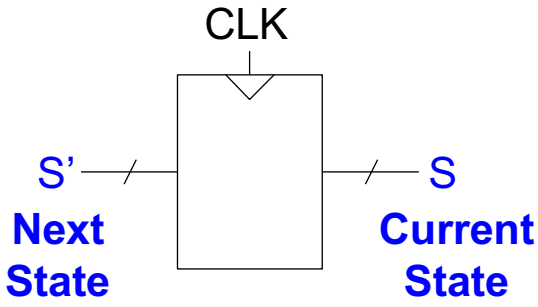
- $A(t+1) = xB \oplus A$
- $B(t+1) = x \oplus B$
- $y_1 = A$; $y_0 = B$

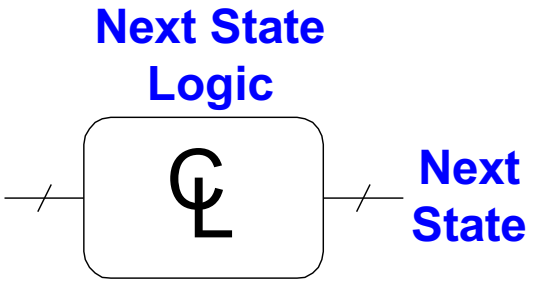
Present state		input	Next state		output	
A	B	x	A	B	y_1	y_0
0	0	0	0	0	0	0
0	0	1	0	1	0	0
0	1	0	0	1	0	1
0	1	1	1	0	0	1
1	0	0	1	0	1	0
1	0	1	1	1	1	0
1	1	0	1	1	1	1
1	1	1	0	0	1	1

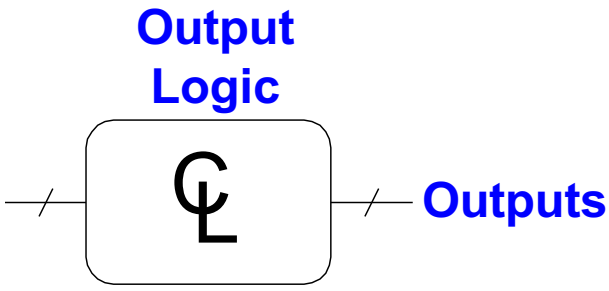


Finite State Machine (FSM)

- Consists of:
  - **State register**
    - Stores current state
    - Loads next state at clock edge
  - **Combinational logic**
    - Computes the next state
    - Computes the outputs



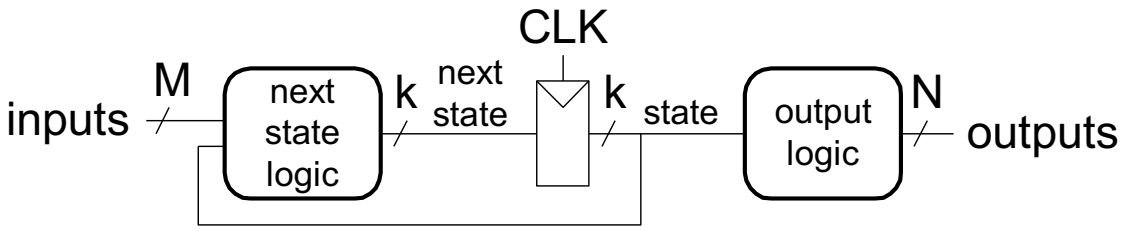




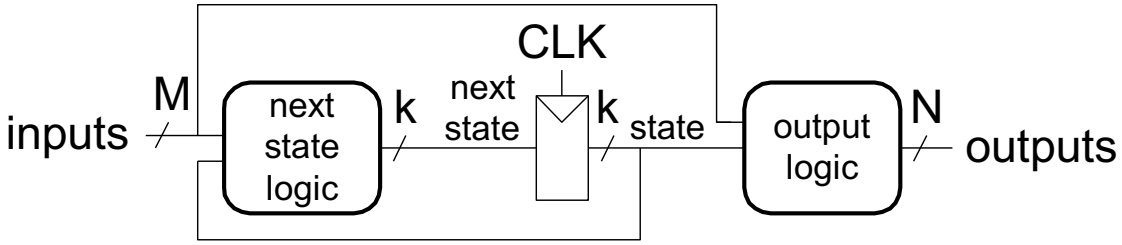
Finite State Machines (FSMs)

- Next state determined by current state and inputs
- Two types of finite state machines differ in output logic:
  - **Moore FSM:** outputs depend only on current state
  - **Mealy FSM:** outputs depend on current state *and* inputs

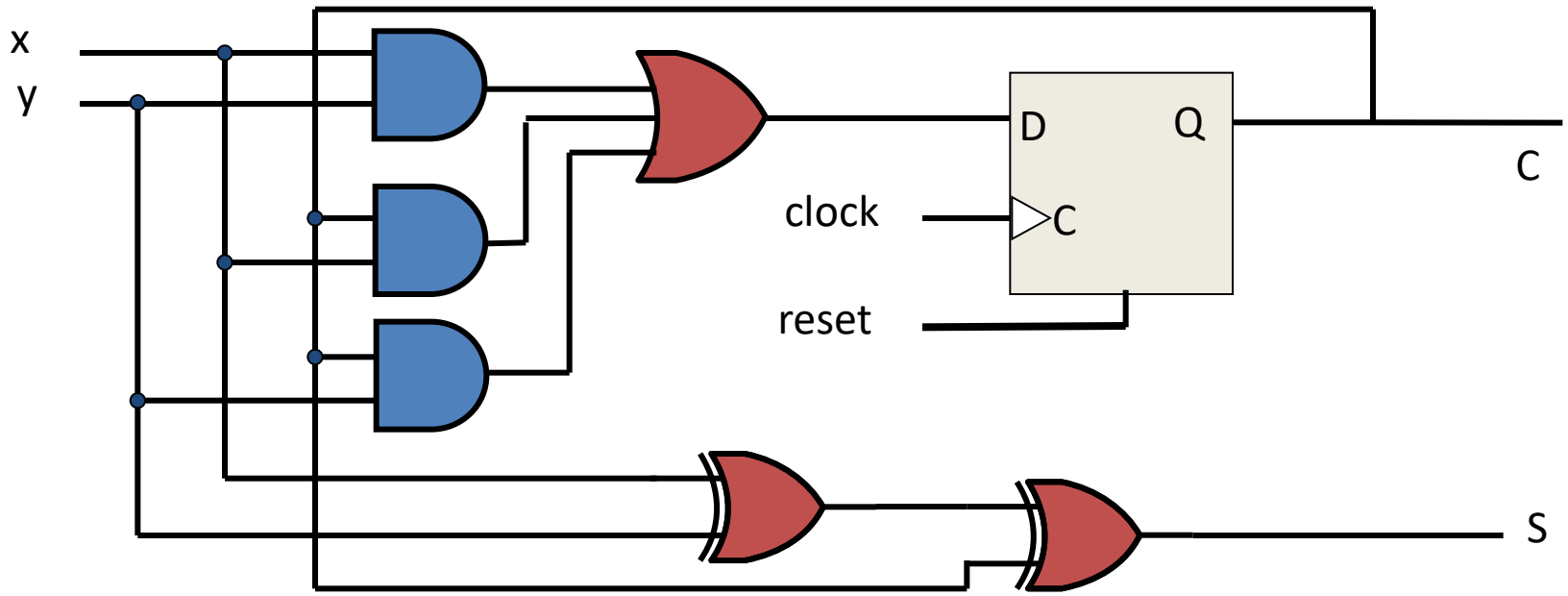
Moore FSM



Mealy FSM



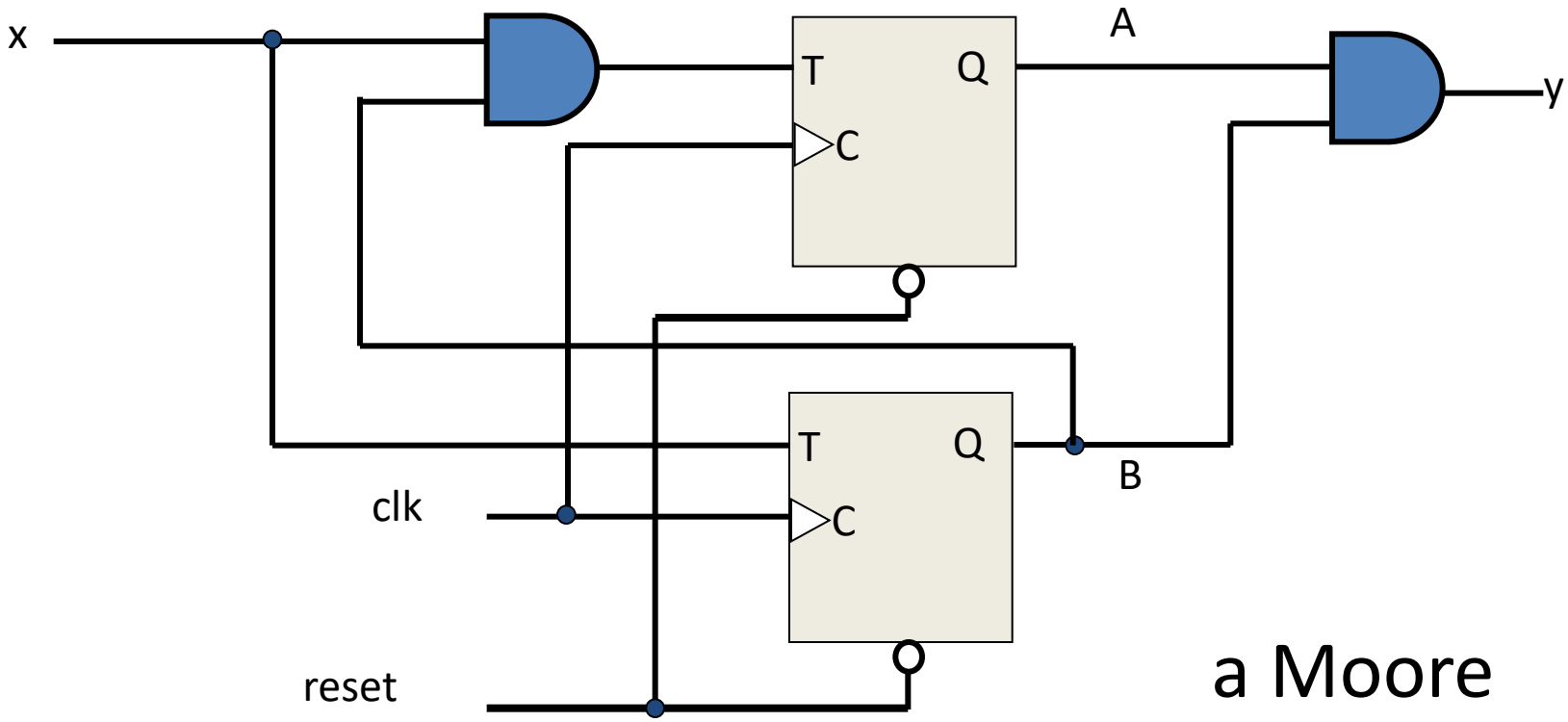
What kind is this?



a Mealy machine

- External inputs x and y are asynchronous
- Thus, outputs may have momentary (incorrect) values
- Inputs must be synchronized with clocks
- Outputs must be sampled only during clock edges

What kind is this?

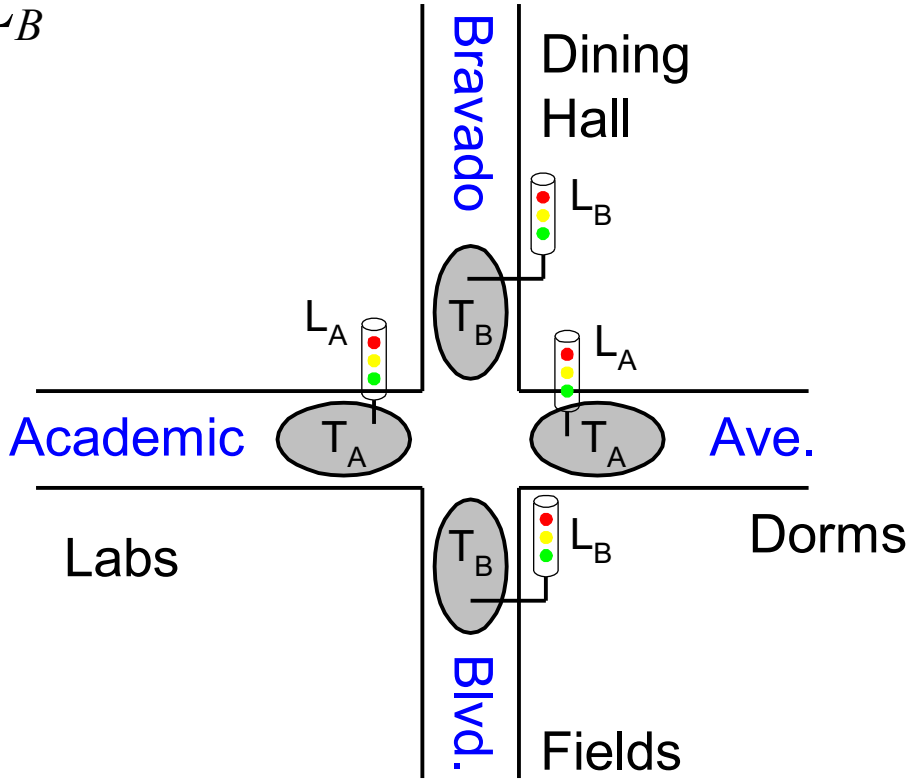


a Moore machine

- Outputs are already synchronized with clock.
- They change synchronously with the clock edge.

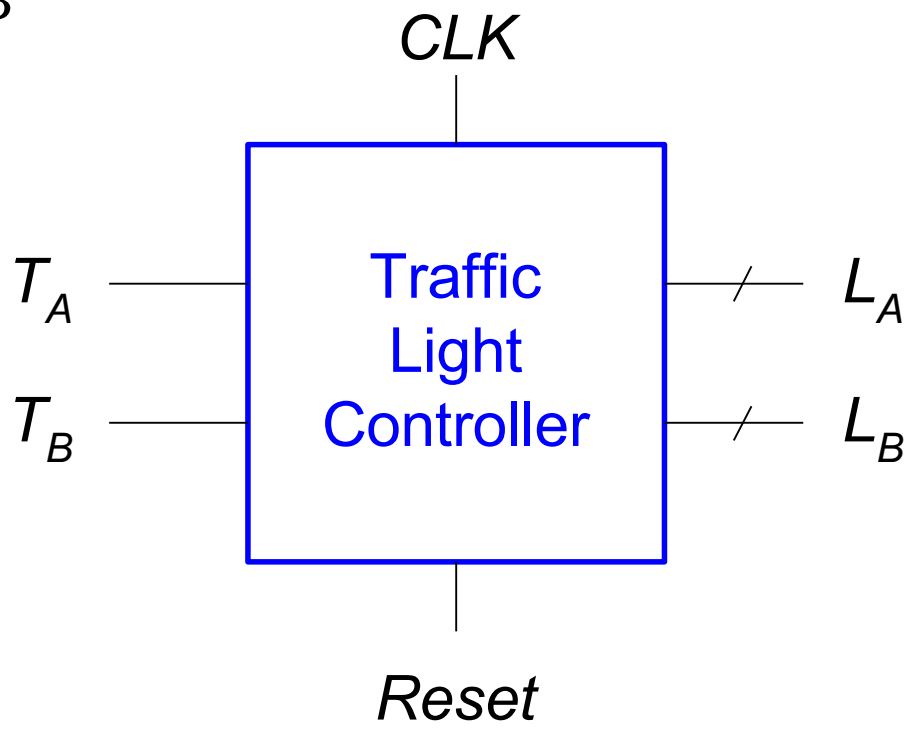
FSM Example

- Traffic light controller
  - Traffic sensors: T_A, T_B (TRUE when there's traffic)
  - Lights: L_A, L_B



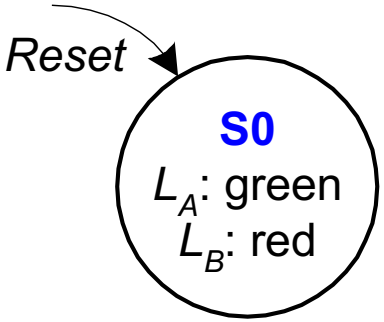
FSM Black Box

- Inputs: *CLK*, *Reset*, T_A, T_B
- Outputs: L_A, L_B



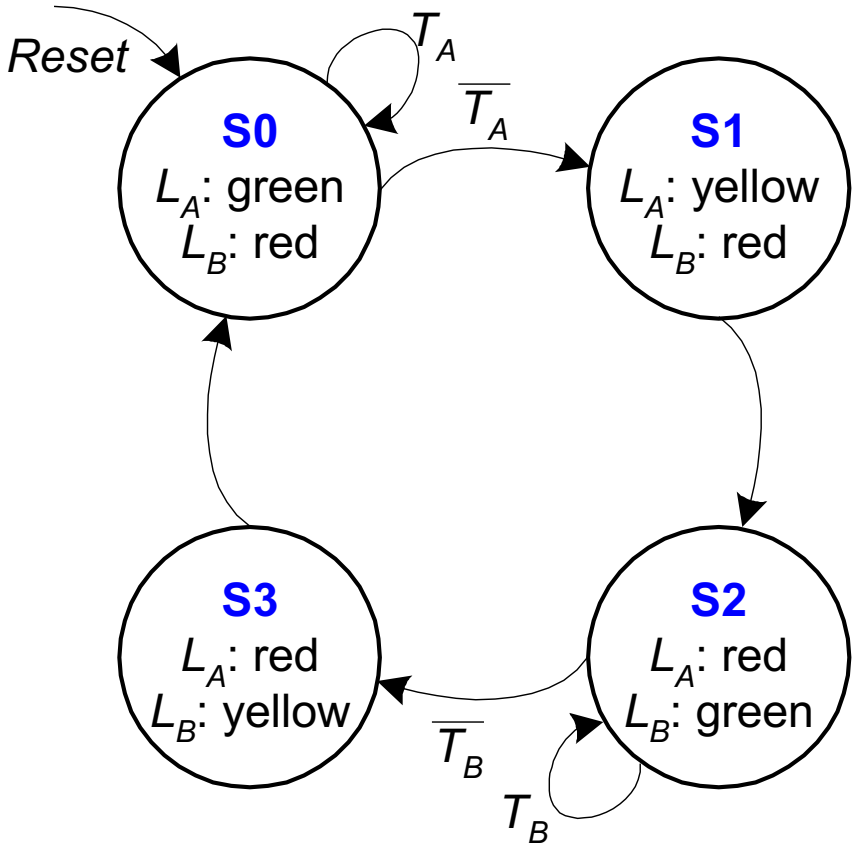
FSM State Transition Diagram

- **Moore FSM:** outputs labeled in each state
- **States:** Circles
- **Transitions:** Arcs



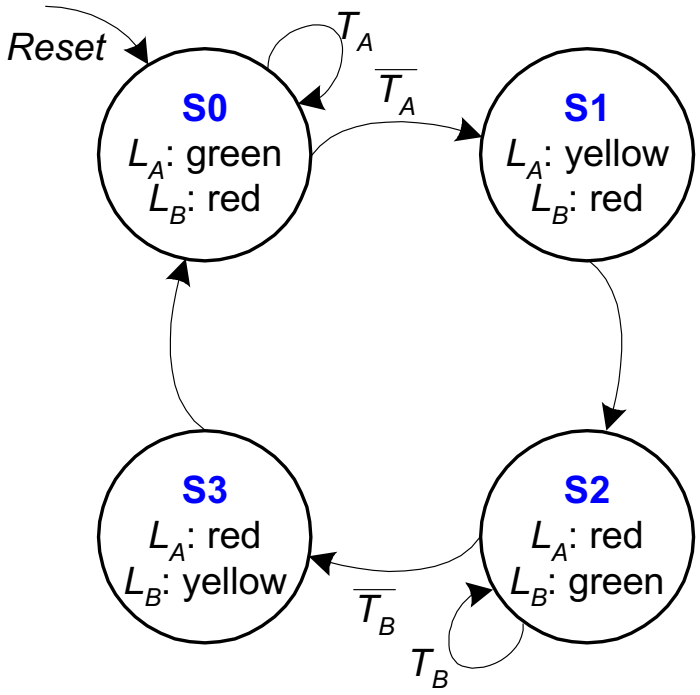
FSM State Transition Diagram

- **Moore FSM:** outputs labeled in each state
- **States:** Circles
- **Transitions:** Arcs



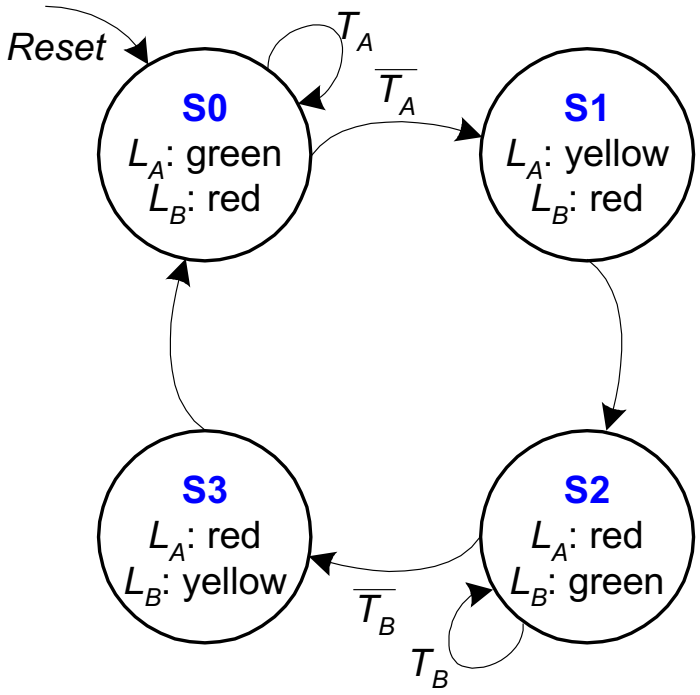
FSM State Transition Table

Current State S	Inputs T_A	T_B	Next State S'
S0	0	X	
S0	1	X	
S1	X	X	
S2	X	0	
S2	X	1	
S3	X	X	



FSM State Transition Table

Current State S	Inputs T_A	T_B	Next State S'
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0



FSM Encoded State Transition Table

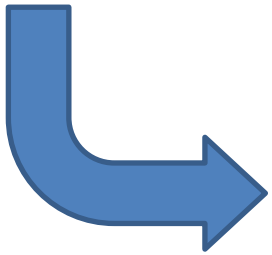
Current State S	Inputs T_A	T_B	Next State S'
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0

State	Encoding
S0	00
S1	01
S2	10
S3	11

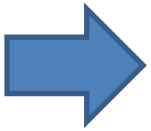
Current State S_1	S_0	Inputs T_A	T_B	Next State S'_1	S'_0
0	0	0	X		
0	0	1	X		
0	1	X	X		
1	0	X	0		
1	0	X	1		
1	1	X	X		

FSM Encoded State Transition Table

Current State S	Inputs T_A	T_B	Next State S'
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0



State	Encoding
S0	00
S1	01
S2	10
S3	11



Current State S_1	S_0	Inputs T_A	T_B	Next State S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

FSM Encoded State Transition Table

Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

FSM Encoded State Transition Table

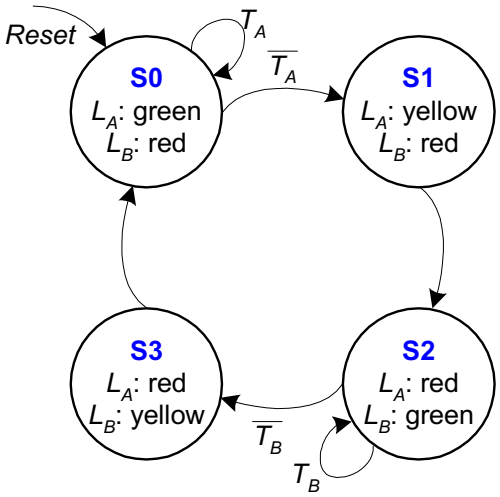
Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

$$S'_1 = S_1 \oplus S_0$$
$$S'_0 = \overline{S_1}\,\overline{S_0}\,\overline{T_A} + S_1\overline{S_0}\,\overline{T_B}$$

FSM Output Table

Current State		Outputs			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0				
0	1				
1	0				
1	1				

Output	Encoding
green	00
yellow	01
red	10



FSM Output Table

Current State		Outputs			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

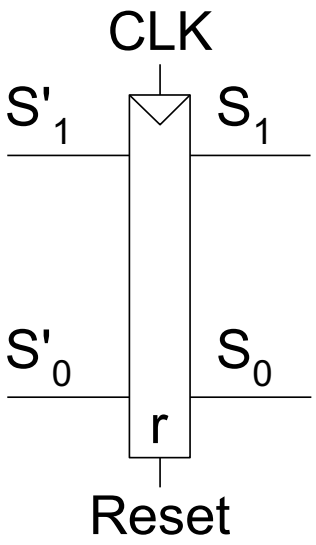
FSM Output Table

Current State		Outputs			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

Output	Encoding
green	00
yellow	01
red	10

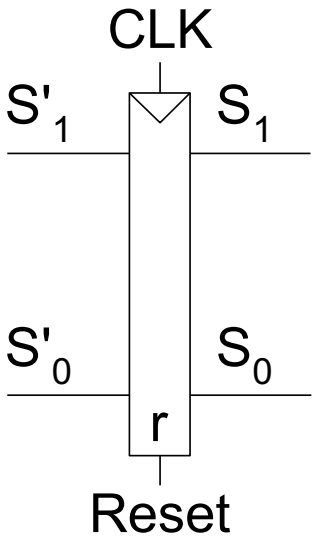
$$L_{A1} = S_1$$
$$L_{A0} = \overline{S_1} S_0$$
$$L_{B1} = \overline{S_1}$$
$$L_{B0} = S_1 S_0$$

FSM Schematic: State Register



state register

FSM Schematic: State Register

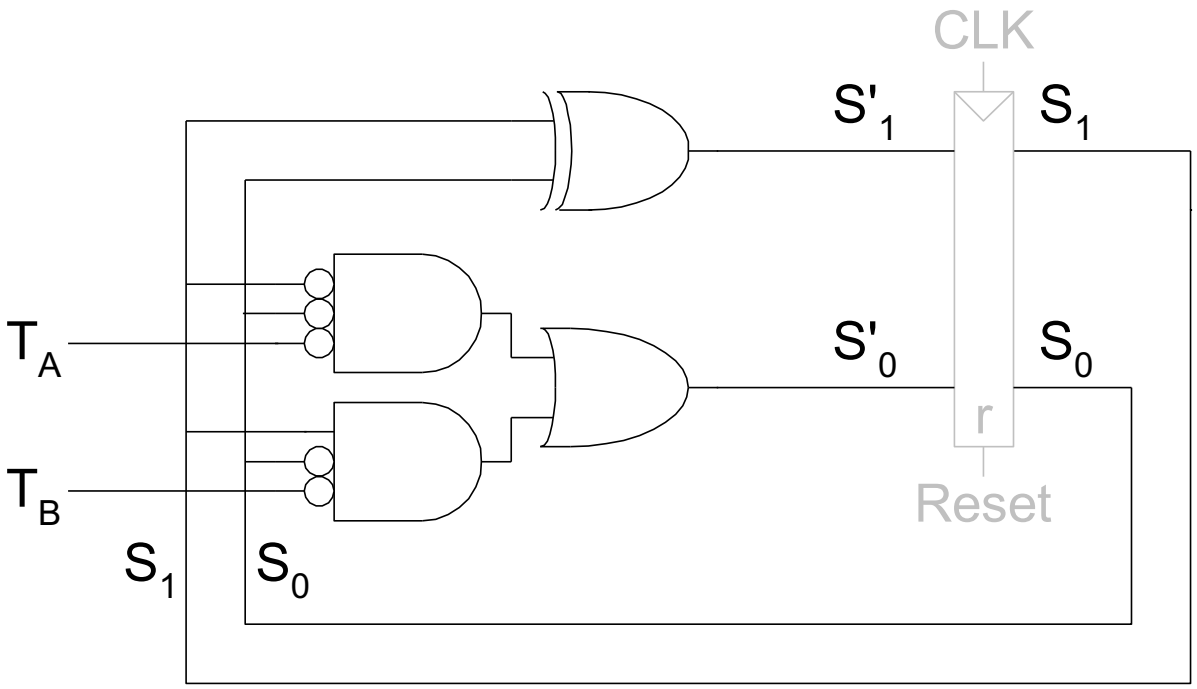


state register

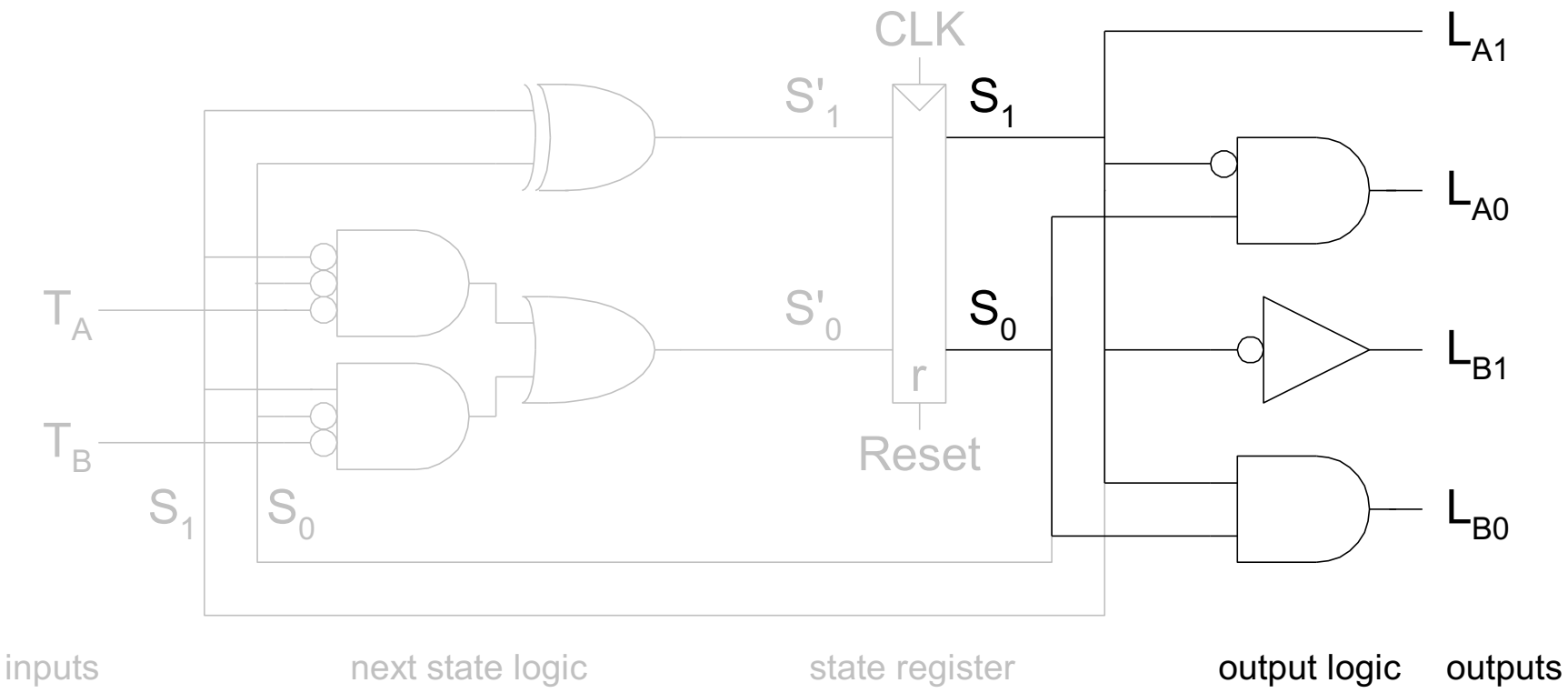
$$S'_1 = S_1 \oplus S_0$$
$$S'_0 = \overline{S_1}\,\overline{S_0}\,\overline{T_A} + S_1\overline{S_0}\,\overline{T_B}$$

$$L_{A1} = S_1$$
$$L_{A0} = \overline{S_1}S_0$$
$$L_{B1} = \overline{S_1}$$
$$L_{B0} = S_1S_0$$

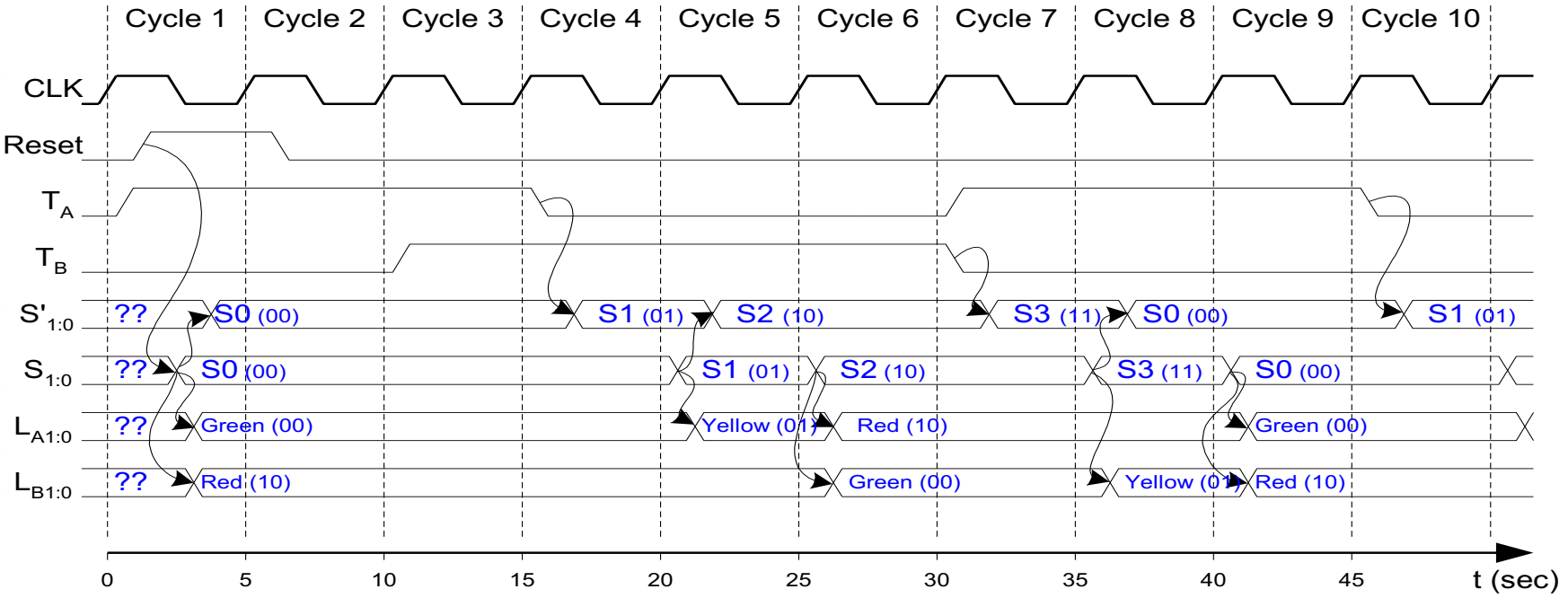
FSM Schematic: Next State Logic

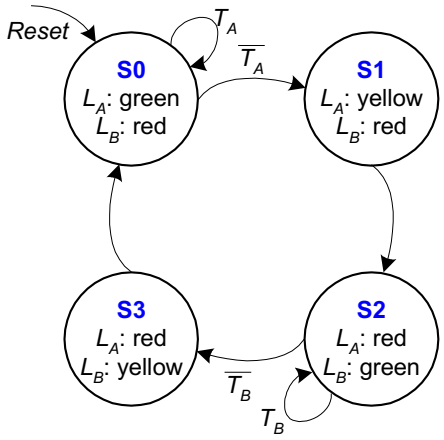


FSM Schematic: Output Logic



FSM Timing Diagram



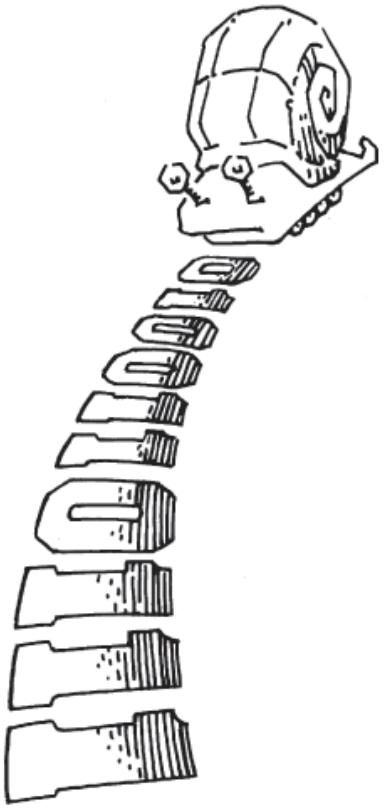


FSM State Encoding

- **Binary** encoding:
  - i.e., for four states, 00, 01, 10, 11
- **One-hot** encoding
  - One state bit per state
  - Only one state bit HIGH at once
  - i.e., for 4 states, 0001, 0010, 0100, 1000
  - Requires more flip-flops
  - Often next state and output logic is simpler

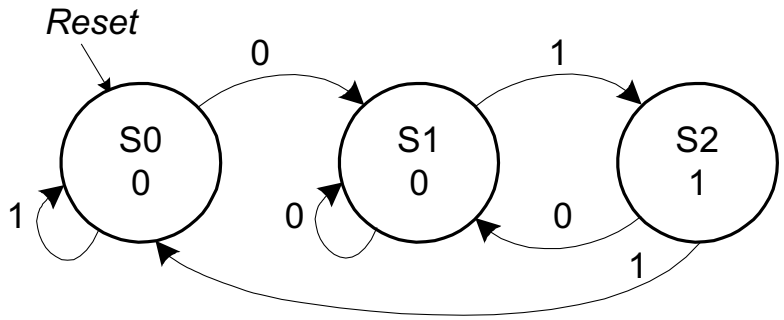
Moore vs. Mealy FSM

- Alyssa P. Hacker has a snail that crawls down a paper tape with 1's and 0's on it. The snail smiles whenever the last two digits it has crawled over are 01. Design Moore and Mealy FSMs of the snail's brain.



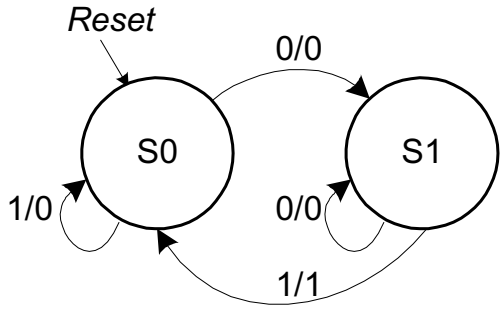
State Transition Diagrams

Moore FSM



Moore FSM: arcs indicate inputs,
outputs are labeled inside circles (each state)

Mealy FSM



these are just 2 alternatives.
other designs possible too

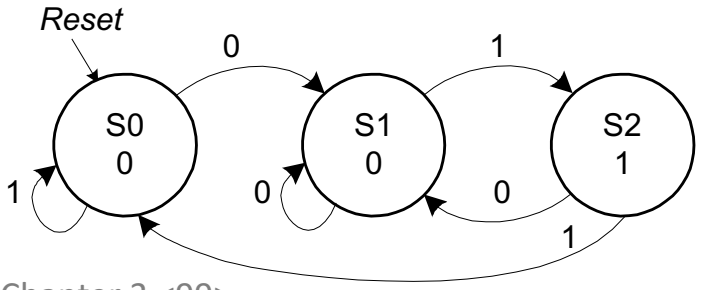
Mealy FSM: arcs indicate input/output

Moore FSM State Transition Table

Current State		Inputs	Next State	
S_1	S_0	A	S'_1	S'_0
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		

State	Encoding
S0	00
S1	01
S2	10

Moore FSM



Moore FSM State Transition Table

Current State		Inputs	Next State	
S_1	S_0	A	S'_1	S'_0
0	0	0	0	1
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0

Moore FSM State Transition Table

Current State		Inputs	Next State	
S_1	S_0	A	S'_1	S'_0
0	0	0	0	1
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	X	X
1	1	1	X	X

Moore FSM State Transition Table

Current State		Inputs	Next State	
S_1	S_0	A	S'_1	S'_0
0	0	0	0	1
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0

State	Encoding
S0	00
S1	01
S2	10

$$S_1' = S_0 A$$
$$S_0' = \overline{A}$$

Moore FSM Output Table

Current State		Output
S_1	S_0	Y
0	0	
0	1	
1	0	

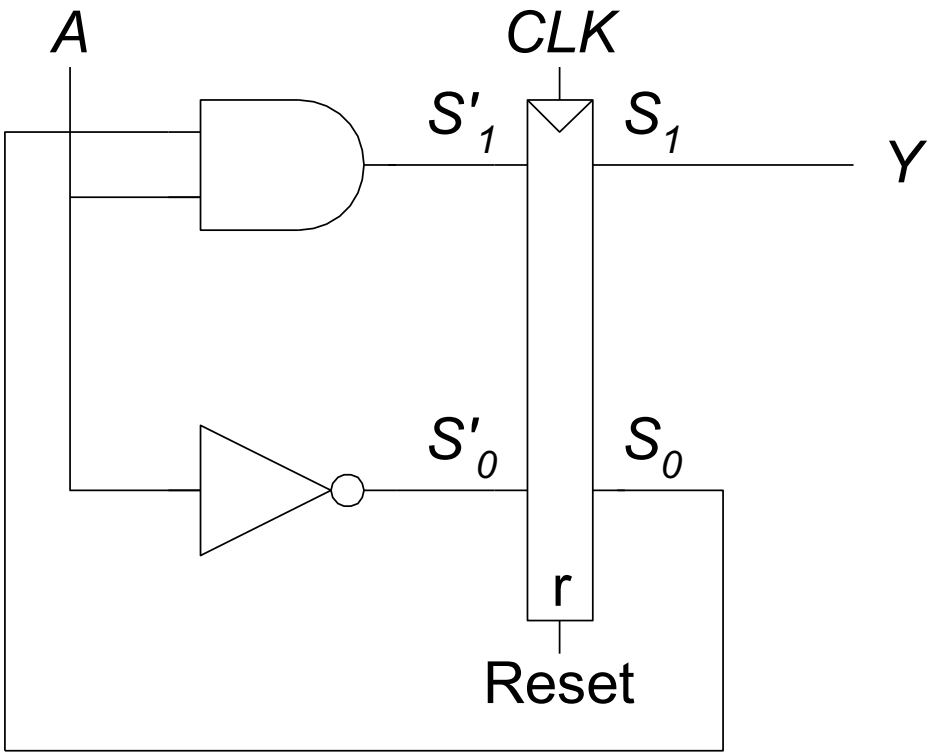
$$Y = S_1$$

Moore FSM Output Table

Current State		Output
S_1	S_0	Y
0	0	0
0	1	0
1	0	1

$$Y = S_1$$

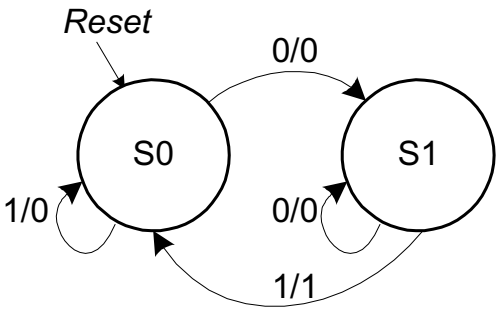
Moore FSM Schematic



Mealy FSM State Transition & Output Table

Current State	Input	Next State	Output
S_0	A	S'_0	Y
0	0		
0	1		
1	0		
1	1		

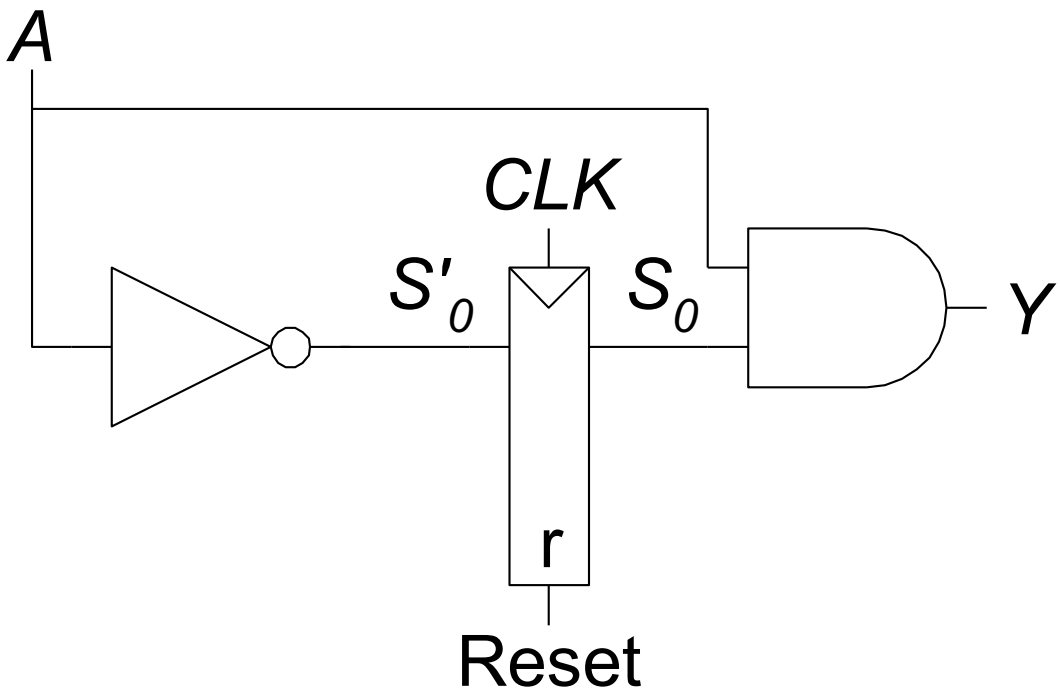
Mealy FSM



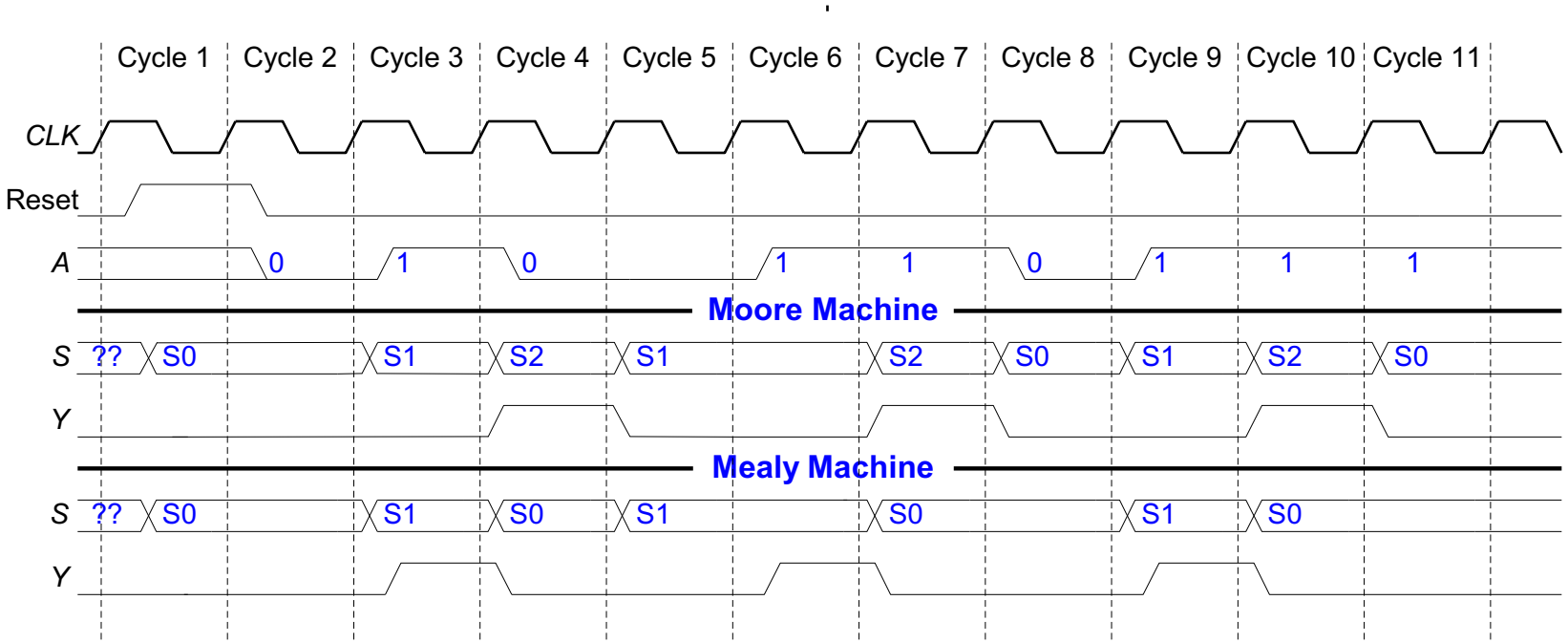
Mealy FSM State Transition & Output Table

Current State	Input	Next State	Output
S_0	A	S'_0	Y
0	0	1	0
0	1	0	0
1	0	1	0
1	1	0	1

Mealy FSM Schematic

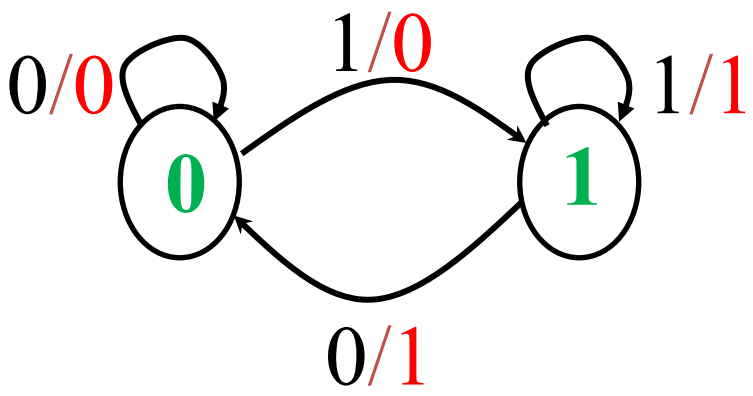


Moore & Mealy Timing Diagram



Design Example - 1

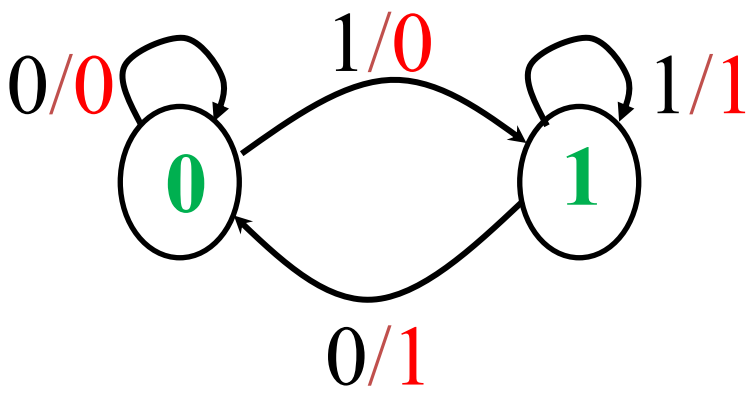
- Implement the following state diagram with D FFs.



x	Q(t)	Q(t+1)	D	Z

Design Example - 1

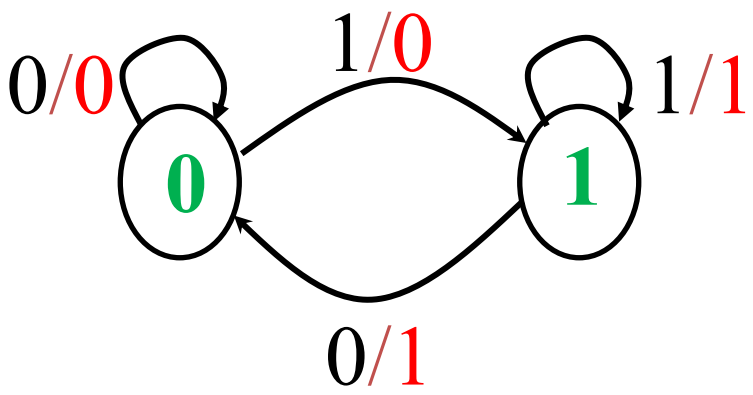
- Implement the following state diagram with D FFs.



x	Q(t)	Q(t+1)	D	Z
0	0			
0	1			
1	0			
1	1			

Design Example - 1

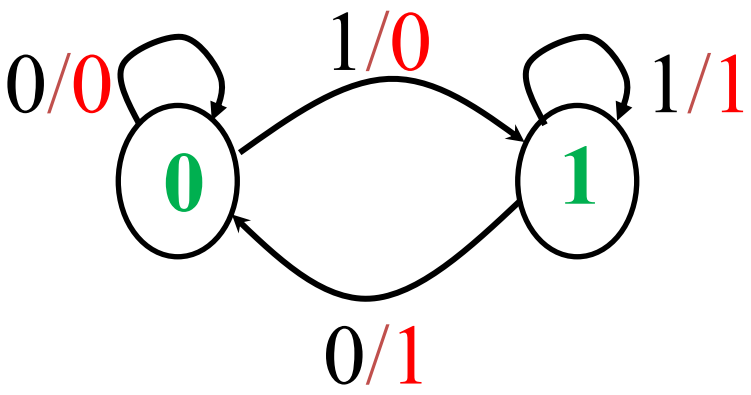
- Implement the following state diagram with D FFs.



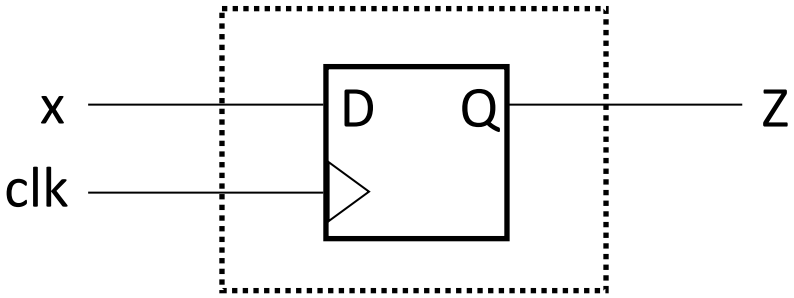
x	Q(t)	Q(t+1)	D	Z
0	0	0		0
0	1	0		1
1	0	1		0
1	1	1		1

Design Example - 1

- Implement the following state diagram with D FFs.



x	Q(t)	Q(t+1)	D	Z
0	0	0	0	0
0	1	0	0	1
1	0	1	1	0
1	1	1	1	1

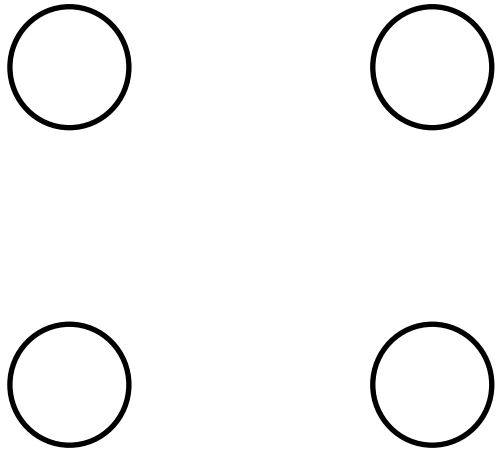


Design Example - 2

- Design a sequential circuit that counts up (00, 01, 10, 11, 00,…) when x=1, and counts down (00,11,10,01,00,…) when x=0. Use JK FFs.

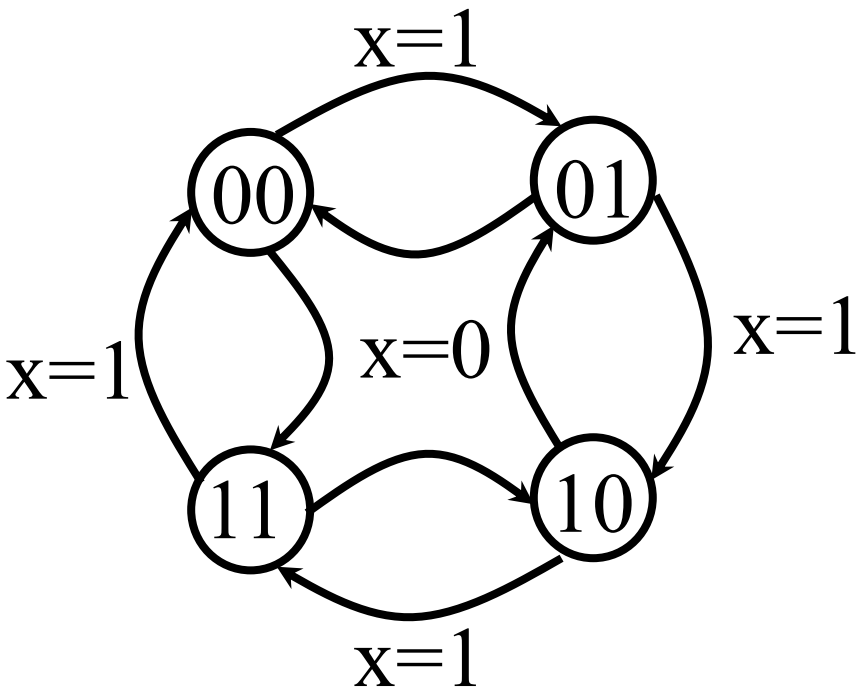
Design Example - 2

- Design a sequential circuit
  - that counts up (00, 01, 10, 11,00,…) when x=1,
  - and counts down (00,11,10,01,00,…) when x=0.
  - Use JK FFs.



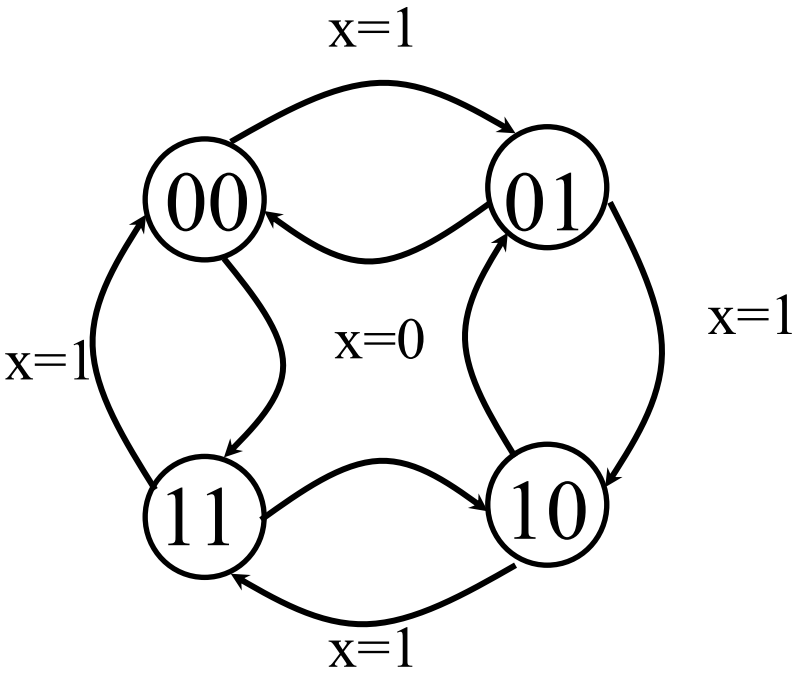
Design Example - 2

- Design a sequential circuit
  - that counts up (00, 01, 10, 11,00,…) when x=1,
  - and counts down (00,11,10,01,00,…) when x=0.
  - Use JK FFs.



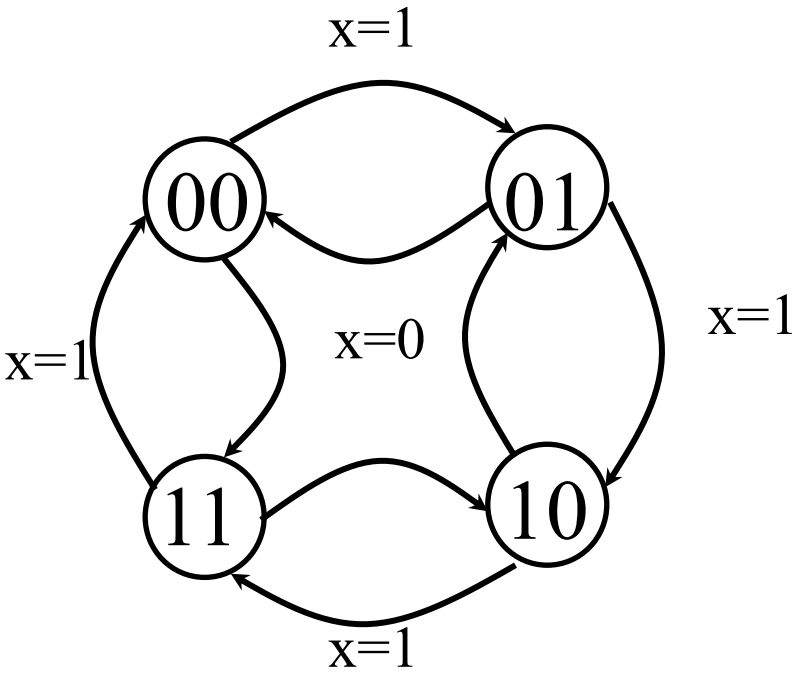
Design Example - 2

x	A	B	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B



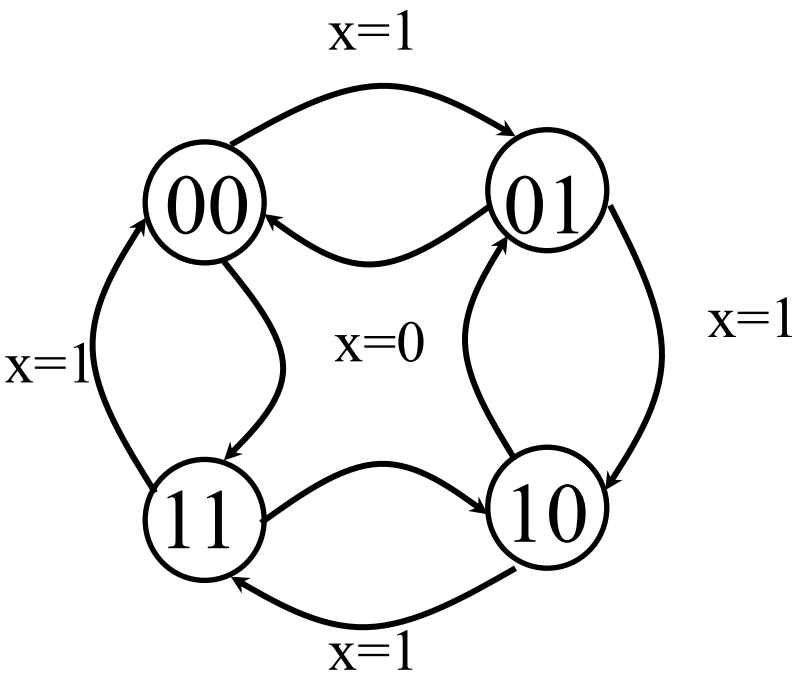
Design Example - 2

x	A	B	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0						
0	0	1						
0	1	0						
0	1	1						
1	0	0						
1	0	1						
1	1	0						
1	1	1						



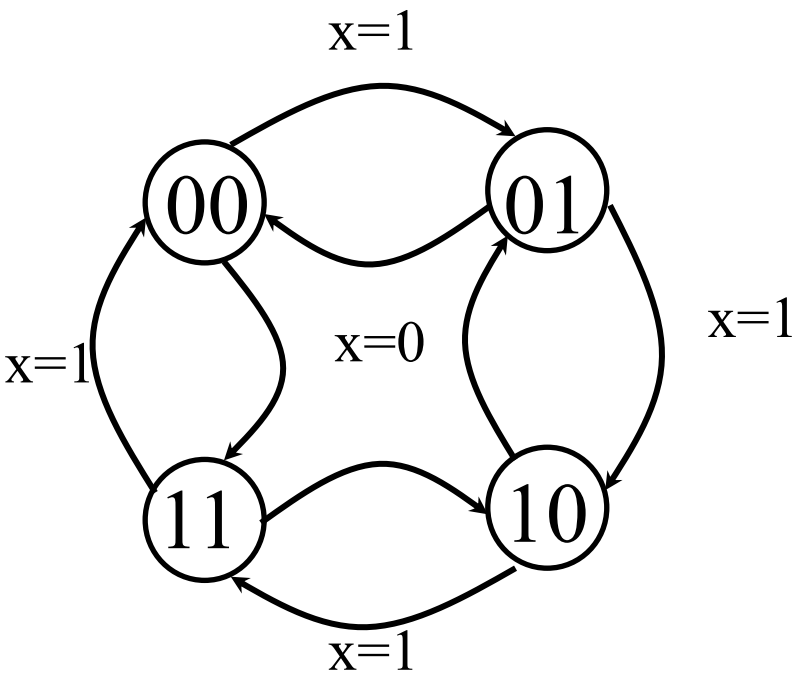
Design Example - 2

x	A	B	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	1	1				
0	0	1	0	0				
0	1	0	0	1				
0	1	1	1	0				
1	0	0	0	1				
1	0	1	1	0				
1	1	0	1	1				
1	1	1	0	0				



Design Example - 2

x	A	B	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	1	1				
0	0	1	0	0				
0	1	0	0	1				
0	1	1	1	0				
1	0	0	0	1				
1	0	1	1	0				
1	1	0	1	1				
1	1	1	0	0				



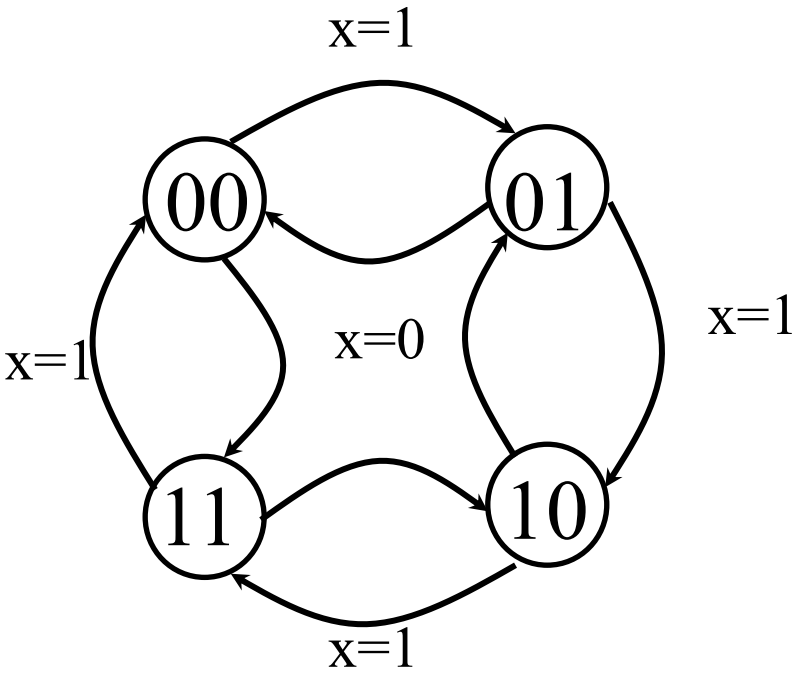
J= { Q(t+1) Q(t)=0 ; X Q(t)=1

K= { Q(t+1)' Q(t)=1 ; X Q(t)=0

Q(t)Q(t+1)	00	01	11	10
J,K	0,X	1,X	X,0	X,1

Design Example - 2

x	A	B	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	1	1	1			
0	0	1	0	0	0			
0	1	0	0	1	X			
0	1	1	1	0	X			
1	0	0	0	1	0			
1	0	1	1	0	1			
1	1	0	1	1	X			
1	1	1	0	0	X			

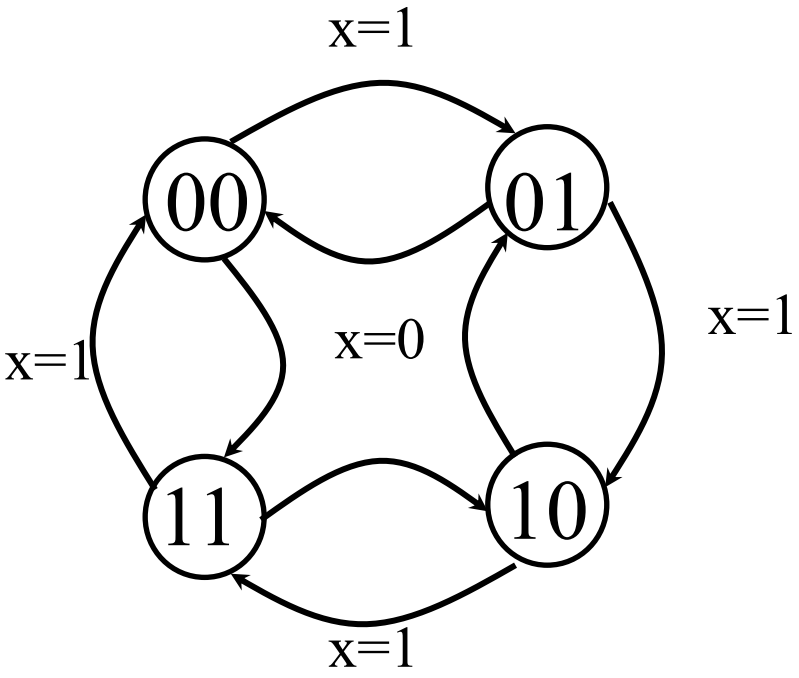


J= { Q(t+1) Q(t)=0 ; X Q(t)=1

K= { Q(t+1)' Q(t)=1 ; X Q(t)=0

Design Example - 2

x	A	B	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	1	1	1	X		
0	0	1	0	0	0	X		
0	1	0	0	1	X	1		
0	1	1	1	0	X	0		
1	0	0	0	1	0	X		
1	0	1	1	0	1	X		
1	1	0	1	1	X	0		
1	1	1	0	0	X	1		

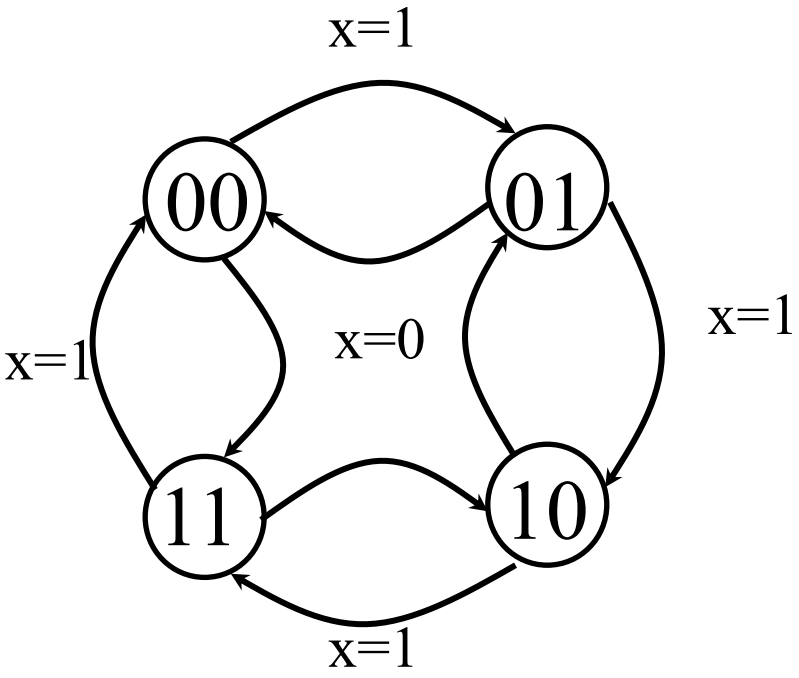


J= { Q(t+1) Q(t)=0 ; X Q(t)=1

K= { Q(t+1)' Q(t)=1 ; X Q(t)=0

Design Example - 2

x	A	B	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	1	1	1	X	1	
0	0	1	0	0	0	X	X	
0	1	0	0	1	X	1	1	
0	1	1	1	0	X	0	X	
1	0	0	0	1	0	X	1	
1	0	1	1	0	1	X	X	
1	1	0	1	1	X	0	1	
1	1	1	0	0	X	1	X	

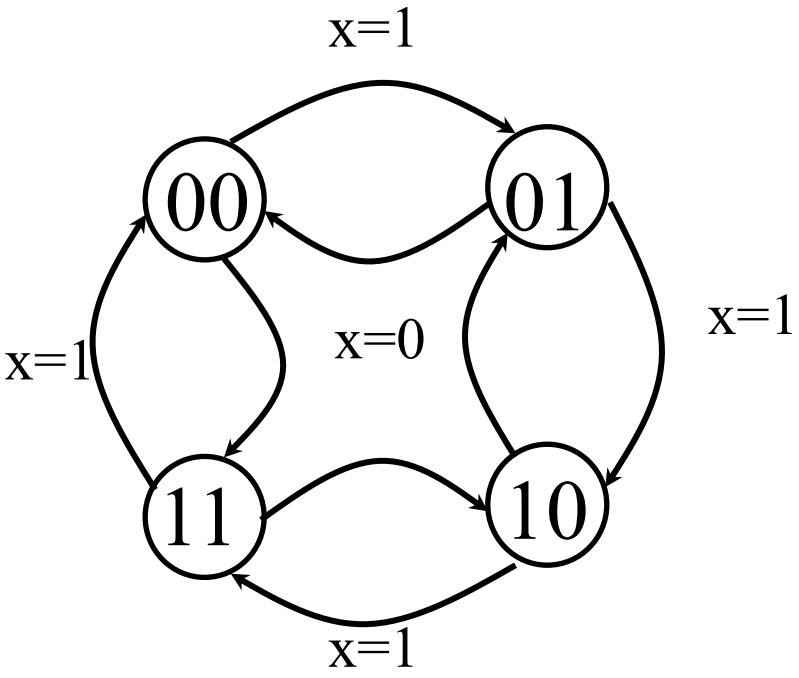


J= { Q(t+1) Q(t)=0 ; X Q(t)=1

K= { Q(t+1)' Q(t)=1 ; X Q(t)=0

Design Example - 2

x	A	B	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	1	1	1	X	1	X
0	0	1	0	0	0	X	X	1
0	1	0	0	1	X	1	1	X
0	1	1	1	0	X	0	X	1
1	0	0	0	1	0	X	1	X
1	0	1	1	0	1	X	X	1
1	1	0	1	1	X	0	1	X
1	1	1	0	0	X	1	X	1

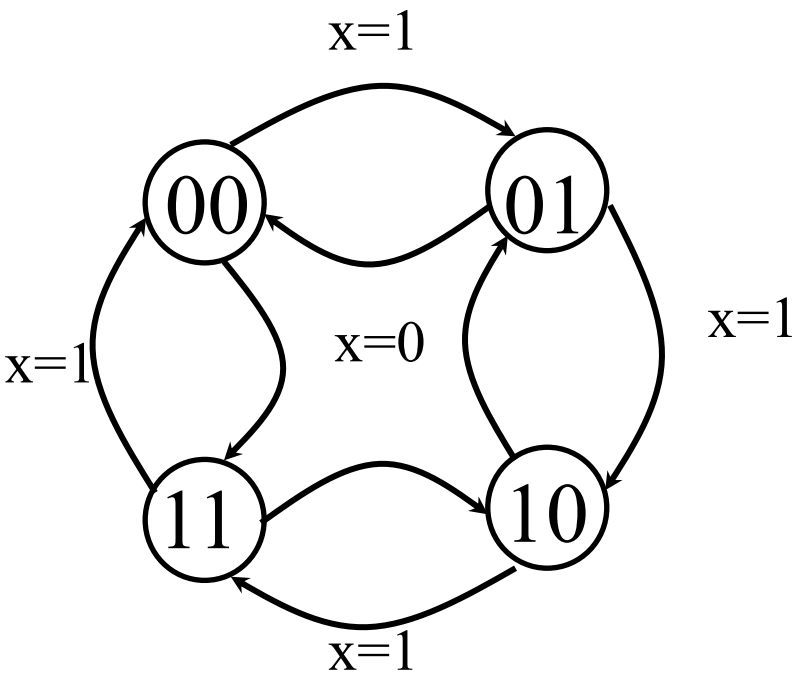


J= { Q(t+1) Q(t)=0 ; X Q(t)=1

K= { Q(t+1)' Q(t)=1 ; X Q(t)=0

Design Example - 2

x	A	B	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	1	1	1	X	1	X
0	0	1	0	0	0	X	X	1
0	1	0	0	1	X	1	1	X
0	1	1	1	0	X	0	X	1
1	0	0	0	1	0	X	1	X
1	0	1	1	0	1	X	X	1
1	1	0	1	1	X	0	1	X
1	1	1	0	0	X	1	X	1

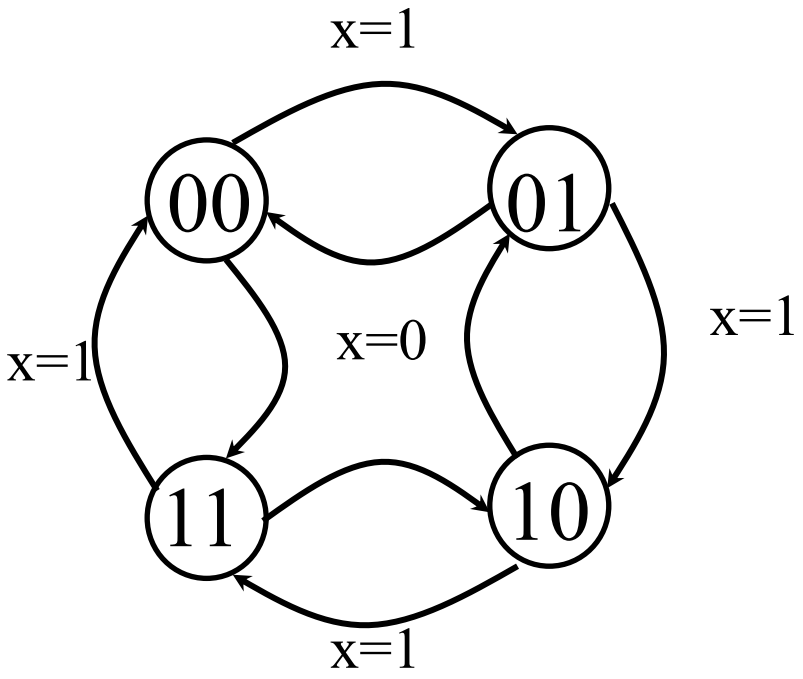


J_A = ?
J_B = ?

K_A = ?
K_B = ?

Design Example - 2

x	A	B	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	1	1	1	X	1	X
0	0	1	0	0	0	X	X	1
0	1	0	0	1	X	1	1	X
0	1	1	1	0	X	0	X	1
1	0	0	0	1	0	X	1	X
1	0	1	1	0	1	X	X	1
1	1	0	1	1	X	0	1	X
1	1	1	0	0	X	1	X	1



x \ AB	00	01	11	10
0	1	0	X	X
1	0	1	X	X

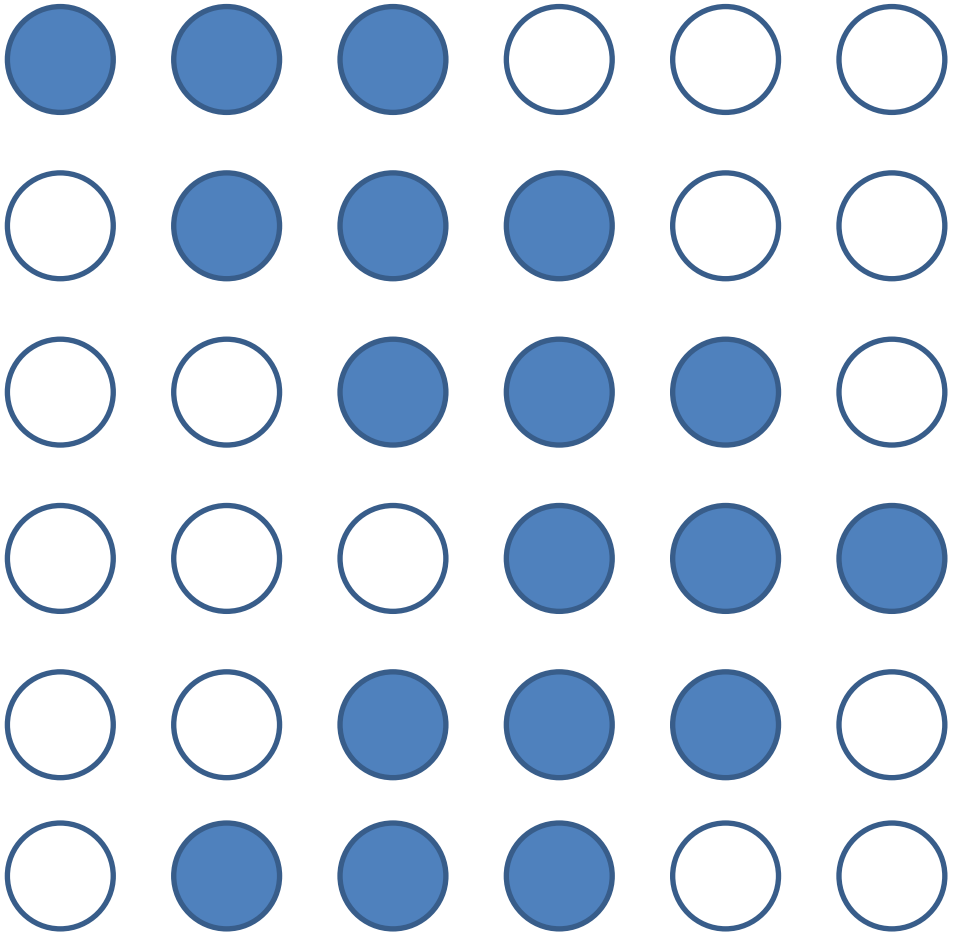
x \ AB	00	01	11	10
0	X	X	0	1
1	X	X	1	0

Design Example - 2

- $J_A = xB + x'B'$
- $K_A = xB + x'B'$
- $J_B = 1$
- $K_B = 1$
- **Let's draw the circuit**

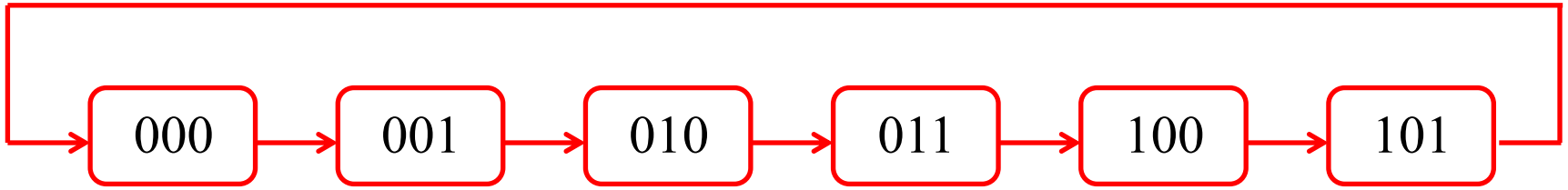
Design Example - 3

OUTPUTS:



design with JK flip-flops

STATE TRANSITIONS:



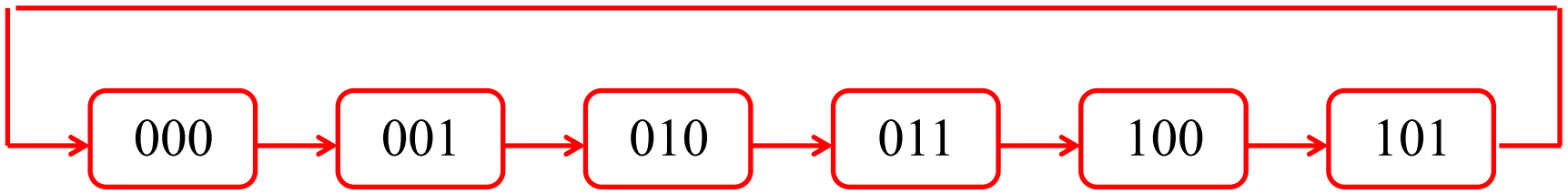
Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	JA	KA	JB	KB	JC	KC
						1	1	1	0	0	0						
						0	1	1	1	0	0						
						0	0	1	1	1	0						
						0	0	0	1	1	1						
						0	0	1	1	1	0						
						0	1	1	1	0	0						

Q(t)Q(t+1)	00	01	11	10
J,K	0,X	1,X	X,0	X,1

$$J = \begin{cases} Q(t+1) & Q(t)=0 \\ X & Q(t)=1 \end{cases} \qquad K = \begin{cases} Q(t+1)' & Q(t)=1 \\ X & Q(t)=0 \end{cases}$$

Mealy or Moore?

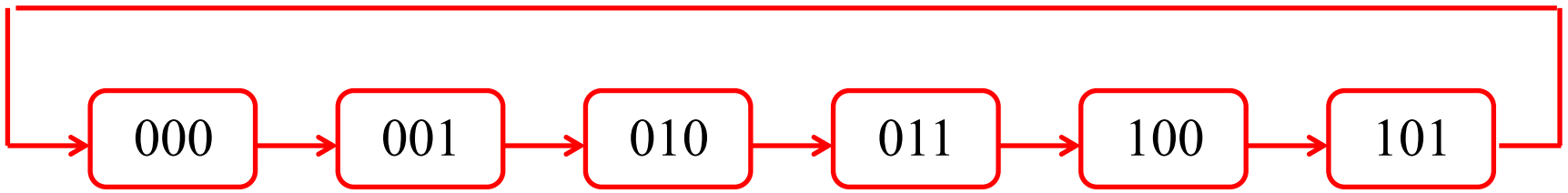


Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	JA	KA	JB	KB	JC	KC
0	0	0	0	0	1	1	1	1	0	0	0						
0	0	1	0	1	0	0	1	1	1	0	0						
0	1	0	0	1	1	0	0	1	1	1	0						
0	1	1	1	0	0	0	0	0	1	1	1						
1	0	0	1	0	1	0	0	1	1	1	0						
1	0	1	0	0	0	0	1	1	1	0	0						

Q(t)Q(t+1)	00	01	11	10
J,K	0,X	1,X	X,0	X,1

$$J = \begin{cases} Q(t+1) & Q(t)=0 \\ X & Q(t)=1 \end{cases} \qquad K = \begin{cases} Q(t+1)' & Q(t)=1 \\ X & Q(t)=0 \end{cases}$$

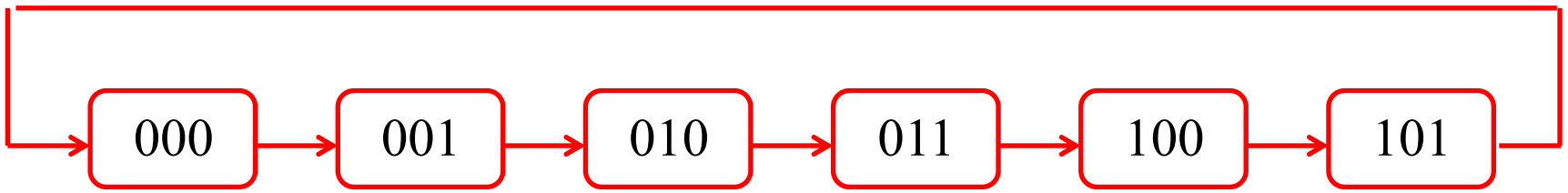


Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0		0		1	
0	0	1	0	1	0	0	1	1	1	0	0	0		1			
0	1	0	0	1	1	0	0	1	1	1	0	0				1	
0	1	1	1	0	0	0	0	0	1	1	1	1					
1	0	0	1	0	1	0	0	1	1	1	0			0		1	
1	0	1	0	0	0	0	1	1	1	0	0			0			

Q(t)Q(t+1)	00	01	11	10
J,K	0,X	1,X	X,0	X,1

$$J = \begin{cases} Q(t+1) & Q(t)=0 \\ X & Q(t)=1 \end{cases} \qquad K = \begin{cases} Q(t+1)' & Q(t)=1 \\ X & Q(t)=0 \end{cases}$$

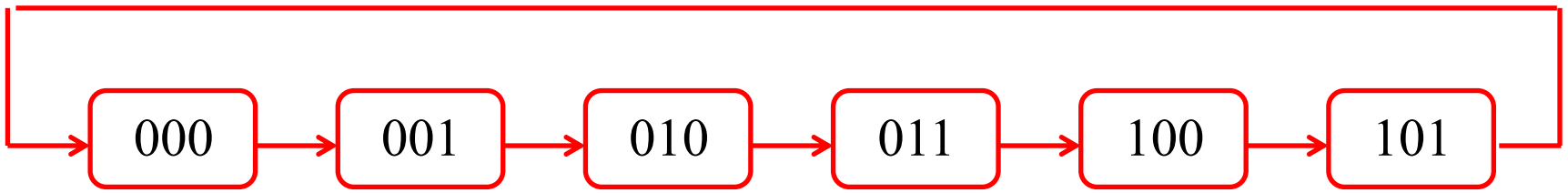


Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	JA	KA	JB	KB	JC	KC
0	0	0	0	0	1	1	1	1	0	0	0	0		0		1	
0	0	1	0	1	0	0	1	1	1	0	0	0		1		X	
0	1	0	0	1	1	0	0	1	1	1	0	0		X		1	
0	1	1	1	0	0	0	0	0	1	1	1	1		X		X	
1	0	0	1	0	1	0	0	1	1	1	0	X		0		1	
1	0	1	0	0	0	0	1	1	1	0	0	X		0		X	

Q(t)Q(t+1)	00	01	11	10
J,K	0,X	1,X	X,0	X,1

$$J = \begin{cases} Q(t+1) & Q(t)=0 \\ X & Q(t)=1 \end{cases} \qquad K = \begin{cases} Q(t+1)' & Q(t)=1 \\ X & Q(t)=0 \end{cases}$$

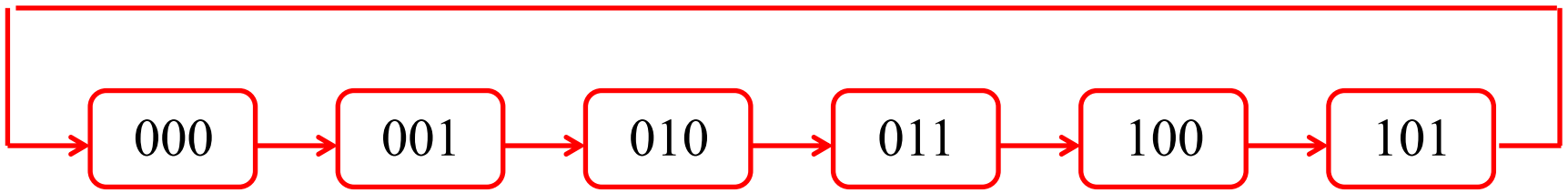


Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	JA	KA	JB	KB	JC	KC
0	0	0	0	0	1	1	1	1	0	0	0	0		0		1	
0	0	1	0	1	0	0	1	1	1	0	0	0		1		X	1
0	1	0	0	1	1	0	0	1	1	1	0	0		X	0	1	
0	1	1	1	0	0	0	0	0	1	1	1	1		X	1	X	1
1	0	0	1	0	1	0	0	1	1	1	0	X	0	0		1	
1	0	1	0	0	0	0	1	1	1	0	0	X	1	0		X	1

Q(t)Q(t+1)	00	01	11	10
J,K	0,X	1,X	X,0	X,1

$$J = \begin{cases} Q(t+1) & Q(t)=0 \\ X & Q(t)=1 \end{cases} \qquad K = \begin{cases} Q(t+1)' & Q(t)=1 \\ X & Q(t)=0 \end{cases}$$

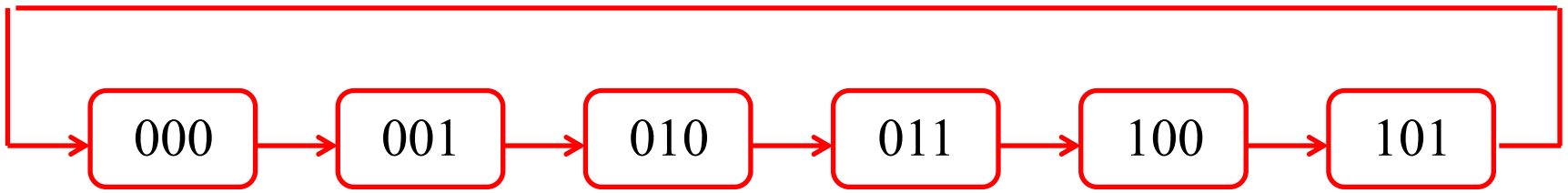


Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	X	0	X	1	X
0	0	1	0	1	0	0	1	1	1	0	0	0	X	1	X	X	1
0	1	0	0	1	1	0	0	1	1	1	0	0	X	X	0	1	X
0	1	1	1	0	0	0	0	0	1	1	1	1	X	X	1	X	1
1	0	0	1	0	1	0	0	1	1	1	0	X	0	0	X	1	X
1	0	1	0	0	0	0	1	1	1	0	0	X	1	0	X	X	1

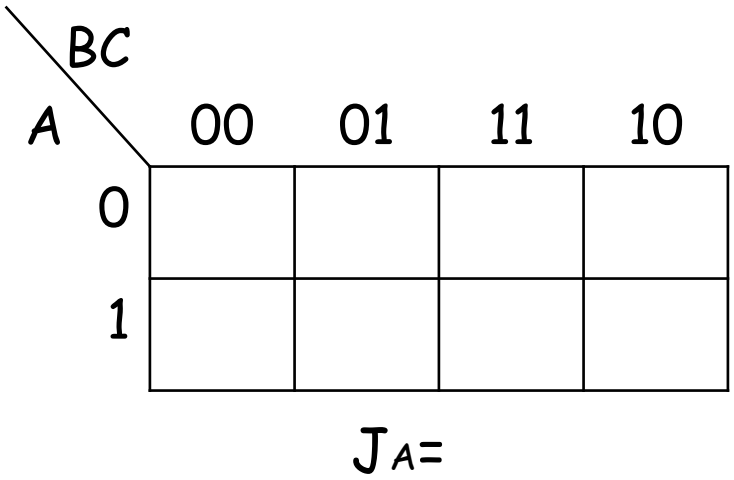
Q(t)Q(t+1)	00	01	11	10
J,K	0,X	1,X	X,0	X,1

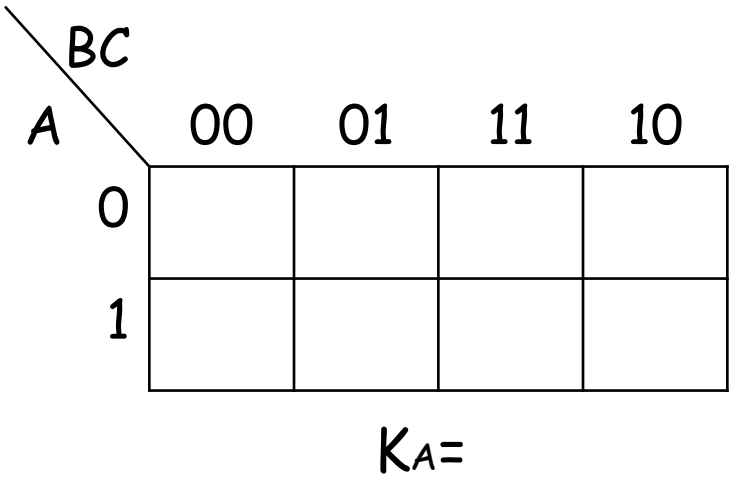
$$J = \begin{cases} Q(t+1) & Q(t)=0 \\ X & Q(t)=1 \end{cases} \qquad K = \begin{cases} Q(t+1)' & Q(t)=1 \\ X & Q(t)=0 \end{cases}$$



Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	X	0	X	1	X
0	0	1	0	1	0	0	1	1	1	0	0	0	X	1	X	X	1
0	1	0	0	1	1	0	0	1	1	1	0	0	X	X	0	1	X
0	1	1	1	0	0	0	0	0	1	1	1	1	X	X	1	X	1
1	0	0	1	0	1	0	0	1	1	1	0	X	0	0	X	1	X
1	0	1	0	0	0	0	1	1	1	0	0	X	1	0	X	X	1





Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	X	0	X	1	X
0	0	1	0	1	0	0	1	1	1	0	0	0	X	1	X	X	1
0	1	0	0	1	1	0	0	1	1	1	0	0	X	X	0	1	X
0	1	1	1	0	0	0	0	0	1	1	1	1	X	X	1	X	1
1	0	0	1	0	1	0	0	1	1	1	0	X	0	0	X	1	X
1	0	1	0	0	0	0	1	1	1	0	0	X	1	0	X	X	1
1	1	0										X	X	X	X	X	X
1	1	1										X	X	X	X	X	X

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

$J_A =$

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

$K_A =$

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	X	0	X	1	X
0	0	1	0	1	0	0	1	1	1	0	0	0	X	1	X	X	1
0	1	0	0	1	1	0	0	1	1	1	0	0	X	X	0	1	X
0	1	1	1	0	0	0	0	0	1	1	1	1	X	X	1	X	1
1	0	0	1	0	1	0	0	1	1	1	0	X	0	0	X	1	X
1	0	1	0	0	0	0	1	1	1	0	0	X	1	0	X	X	1
1	1	0										X	X	X	X	X	X
1	1	1										X	X	X	X	X	X

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

$J_A = BC$

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

$K_A = C$

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	X	0	X	1	X
0	0	1	0	1	0	0	1	1	1	0	0	0	X	1	X	X	1
0	1	0	0	1	1	0	0	1	1	1	0	0	X	X	0	1	X
0	1	1	1	0	0	0	0	0	1	1	1	1	X	X	1	X	1
1	0	0	1	0	1	0	0	1	1	1	0	X	0	0	X	1	X
1	0	1	0	0	0	0	1	1	1	0	0	X	1	0	X	X	1
1	1	0										X	X	X	X	X	X
1	1	1										X	X	X	X	X	X

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

J_B=

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

K_B=

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	X	0	X	1	X
0	0	1	0	1	0	0	1	1	1	0	0	0	X	1	X	X	1
0	1	0	0	1	1	0	0	1	1	1	0	0	X	X	0	1	X
0	1	1	1	0	0	0	0	0	1	1	1	1	X	X	1	X	1
1	0	0	1	0	1	0	0	1	1	1	0	X	0	0	X	1	X
1	0	1	0	0	0	0	1	1	1	0	0	X	1	0	X	X	1
1	1	0										X	X	X	X	X	X
1	1	1										X	X	X	X	X	X

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

$J_B = A'C$

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

$K_B = C$

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	X	0	X	1	X
0	0	1	0	1	0	0	1	1	1	0	0	0	X	1	X	X	1
0	1	0	0	1	1	0	0	1	1	1	0	0	X	X	0	1	X
0	1	1	1	0	0	0	0	0	1	1	1	1	X	X	1	X	1
1	0	0	1	0	1	0	0	1	1	1	0	X	0	0	X	1	X
1	0	1	0	0	0	0	1	1	1	0	0	X	1	0	X	X	1
1	1	0										X	X	X	X	X	X
1	1	1										X	X	X	X	X	X

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

J_C=

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

K_C=

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	X	0	X	1	X
0	0	1	0	1	0	0	1	1	1	0	0	0	X	1	X	X	1
0	1	0	0	1	1	0	0	1	1	1	0	0	X	X	0	1	X
0	1	1	1	0	0	0	0	0	1	1	1	1	X	X	1	X	1
1	0	0	1	0	1	0	0	1	1	1	0	X	0	0	X	1	X
1	0	1	0	0	0	0	1	1	1	0	0	X	1	0	X	X	1
1	1	0										X	X	X	X	X	X
1	1	1										X	X	X	X	X	X

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

$J_C=1$

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

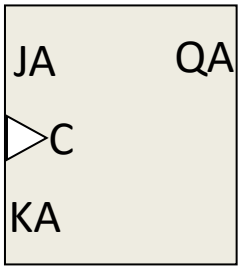
$K_C=1$

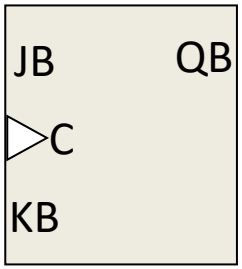
Design Example - 3

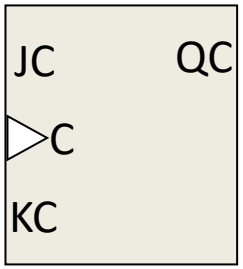
$J_A = BC$ $K_A = C$

$J_B = A'C$ $K_B = C$

$J_C = 1$ $K_C = 1$







We can also solve for outputs z1-z6 by looking at K-maps of A(t),B(t),C(t)

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1
0	0	1	0	1	0	0	1	1	1	0	0	0	1	1	1	1	1
0	1	0	0	1	1	0	0	1	1	1	0	0	0	0	0	1	1
0	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	0	0	1	1	1	0	0	0	0	0	1	1
1	0	1	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1
1	1	0															
1	1	1															

$J_A = BC$

$K_A = C$

$J_B = A'C$

$K_B = C$

$J_C = 1$

$K_C = 1$

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1
0	0	1	0	1	0	0	1	1	1	0	0	0	1	1	1	1	1
0	1	0	0	1	1	0	0	1	1	1	0	0	0	0	0	1	1
0	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	0	0	1	1	1	0	0	0	0	0	1	1
1	0	1	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1
1	1	0										0	0	0	0	1	1
1	1	1										1	1	0	1	1	1

$J_A = BC$

$K_A = C$

$J_B = A'C$

$K_B = C$

$J_C = 1$

$K_C = 1$

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1
0	0	1	0	1	0	0	1	1	1	0	0	0	1	1	1	1	1
0	1	0	0	1	1	0	0	1	1	1	0	0	0	0	0	1	1
0	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	0	0	1	1	1	0	0	0	0	0	1	1
1	0	1	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1
1	1	0	1	1	1							0	0	0	0	1	1
1	1	1	0	0	0							1	1	0	1	1	1

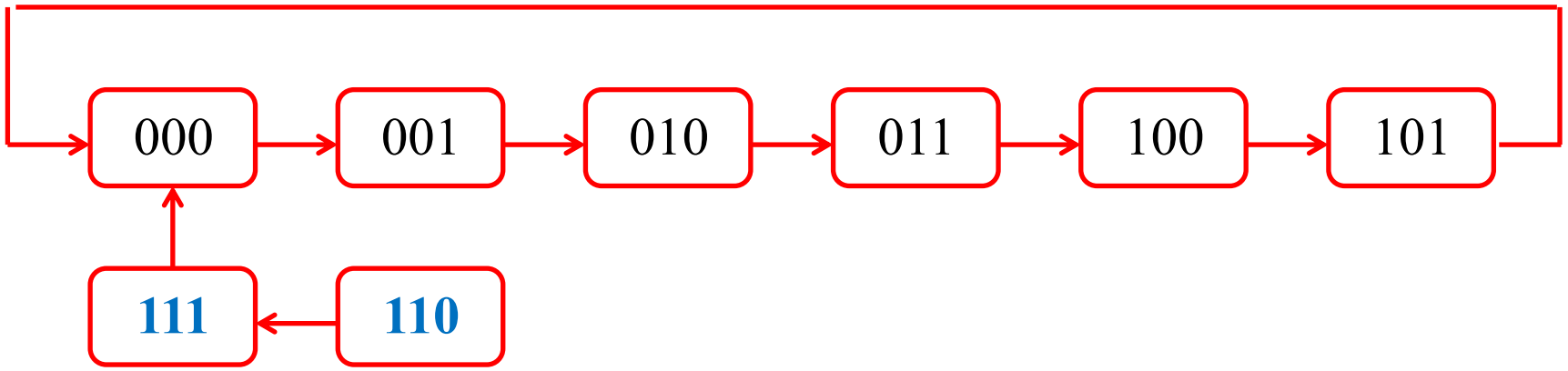
$J_A = BC$ $\quad A(t+1) = BCA' + C'A$ $\quad K_A = C$

$J_B = A'C$ $\quad B(t+1) = A'CB' + C'B$ $\quad K_B = C$

$J_C = 1$ $\quad C(t+1) = C'$ $\quad K_C = 1$

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1
0	0	1	0	1	0	0	1	1	1	0	0	0	1	1	1	1	1
0	1	0	0	1	1	0	0	1	1	1	0	0	0	0	0	1	1
0	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	0	0	1	1	1	0	0	0	0	0	1	1
1	0	1	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1
1	1	0	1	1	1							0	0	0	0	1	1
1	1	1	0	0	0							1	1	0	1	1	1



Design Example - 3

- now let’s redo the design with D FFs

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	D_A	D_B	D_C
0	0	0	0	0	1	1	1	1	0	0	0			
0	0	1	0	1	0	0	1	1	1	0	0			
0	1	0	0	1	1	0	0	1	1	1	0			
0	1	1	1	0	0	0	0	0	1	1	1			
1	0	0	1	0	1	0	0	1	1	1	0			
1	0	1	0	0	0	0	1	1	1	0	0			
1	1	0	1	1	1									
1	1	1	0	0	0									

$A(t+1)=BCA'+C'A$ =

$B(t+1)=A'CB'+C'B$ =

$C(t+1)=C'$ =

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	D_A	D_B	D_C
0	0	0	0	0	1	1	1	1	0	0	0			
0	0	1	0	1	0	0	1	1	1	0	0			
0	1	0	0	1	1	0	0	1	1	1	0			
0	1	1	1	0	0	0	0	0	1	1	1			
1	0	0	1	0	1	0	0	1	1	1	0			
1	0	1	0	0	0	0	1	1	1	0	0			
1	1	0	1	1	1									
1	1	1	0	0	0									

$$A(t+1)=BCA'+C'A = D_A$$

$$B(t+1)=A'CB'+C'B = D_B$$

$$C(t+1)=C' = D_C$$

Design Example - 3

A(t)	B(t)	C(t)	A (t+1)	B (t+1)	C (t+1)	z1	z2	z3	z4	z5	z6	D_A	D_B	D_C
0	0	0	0	0	1	1	1	1	0	0	0			1
0	0	1	0	1	0	0	1	1	1	0	0		1	
0	1	0	0	1	1	0	0	1	1	1	0		1	1
0	1	1	1	0	0	0	0	0	1	1	1	1		
1	0	0	1	0	1	0	0	1	1	1	0	1		1
1	0	1	0	0	0	0	1	1	1	0	0			
1	1	0	1	1	1							1	1	1
1	1	1	0	0	0									

$$A(t+1)=BCA'+C'A = D_A$$

$$B(t+1)=A'CB'+C'B = D_B$$

$$C(t+1)=C' = D_C$$

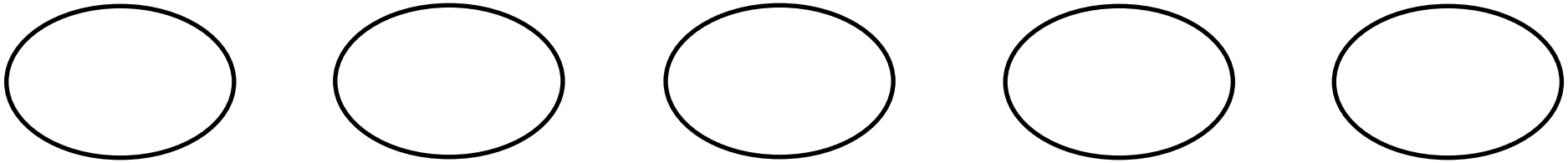
Design Example - 4

- Design a logic circuit with JK flip-flops that detects the sequence 1011 and outputs 1 in that case, 0 otherwise.

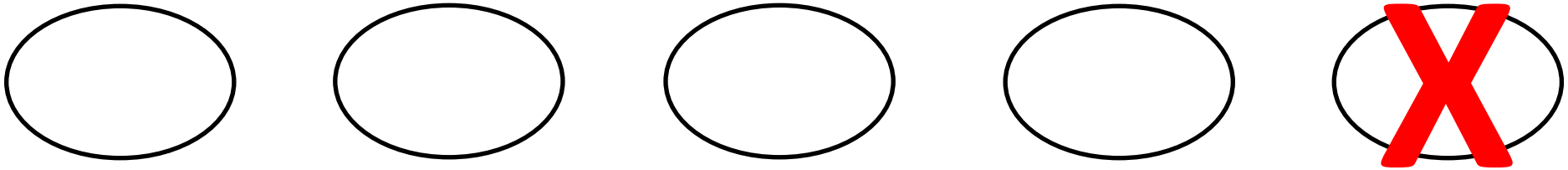
Design Example - 4

?

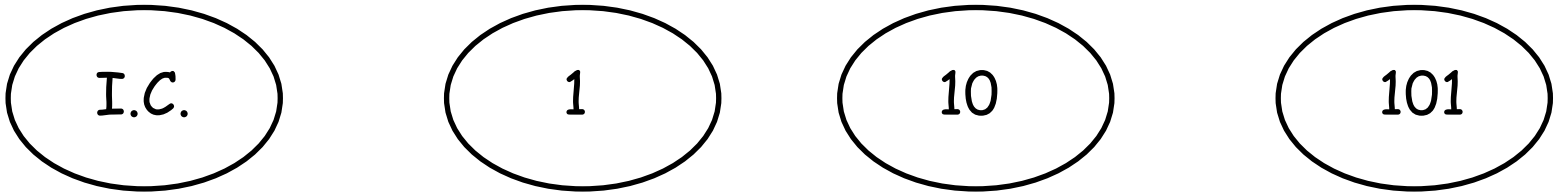
Design Example - 4



Design Example - 4



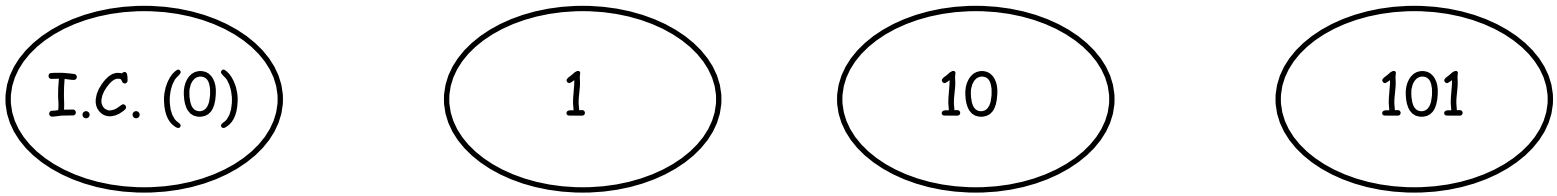
Design Example - 4



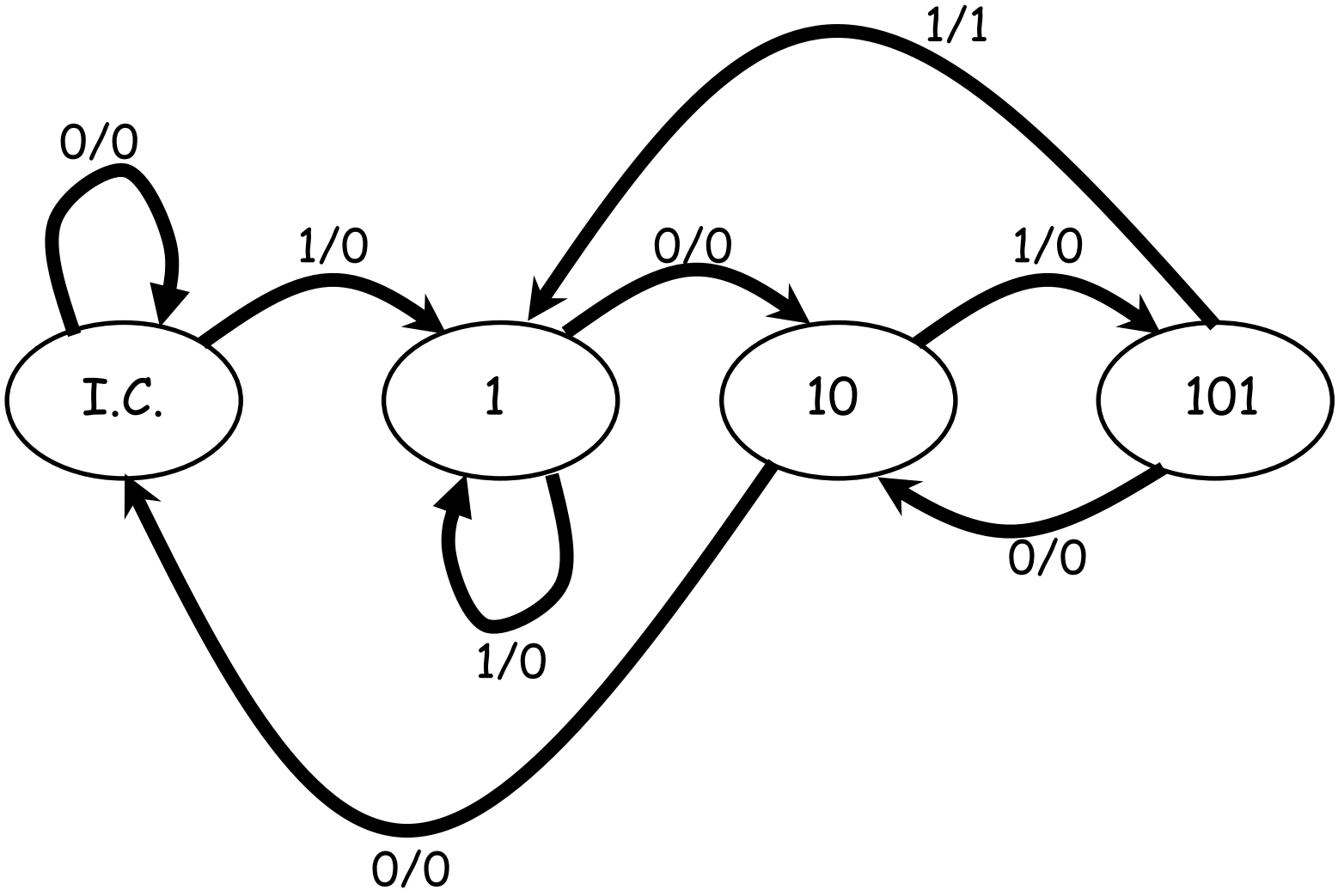
Design Example - 4

consider the sequence:
0010010101011011001

consider the sequence:
0010010101011011001

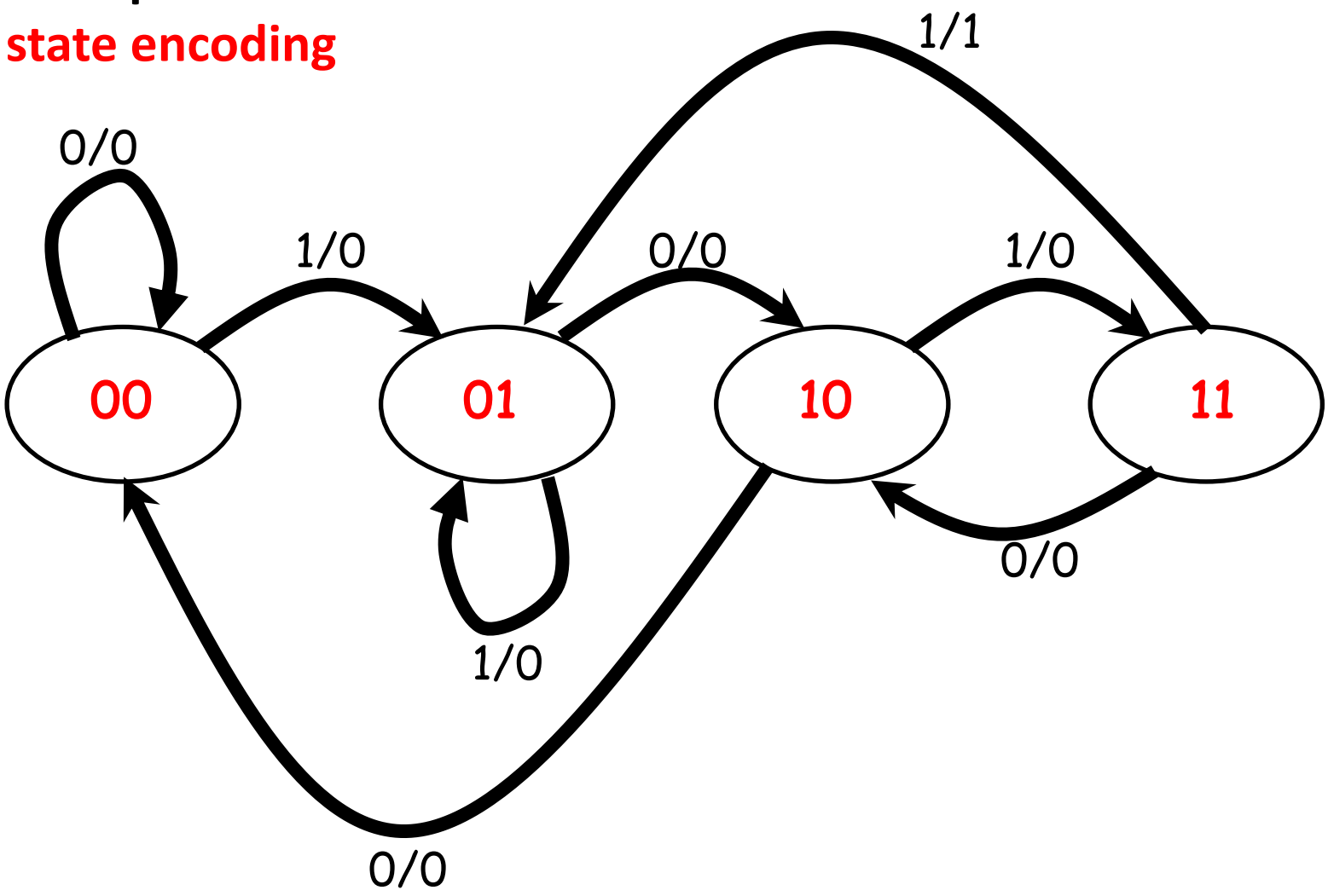


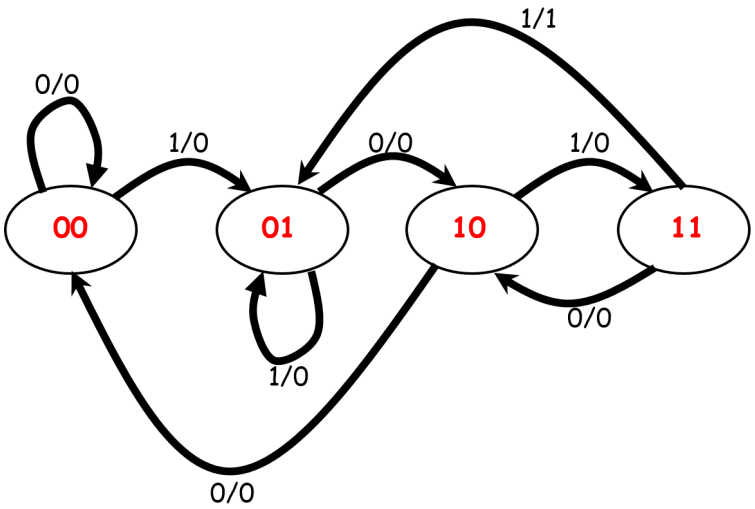
Design Example - 4



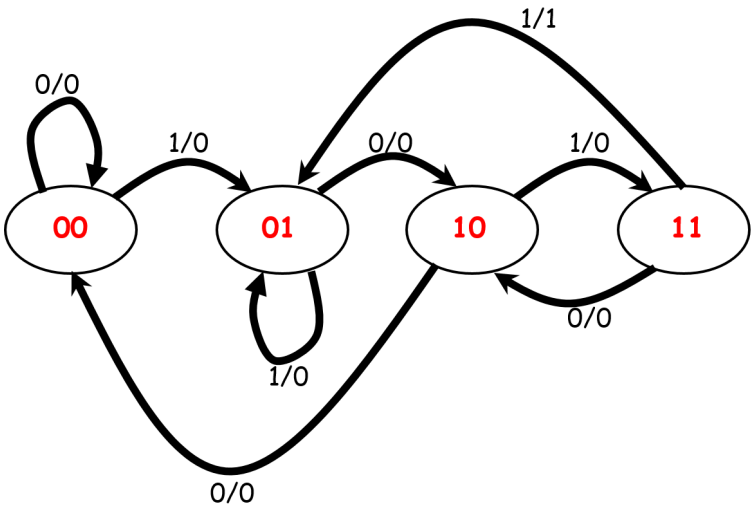
Design Example - 4

now we represent
with state encoding



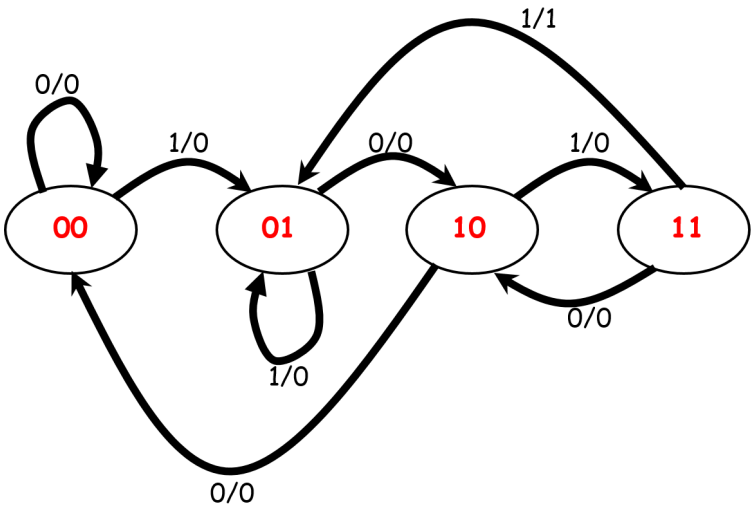


x(t)	A(t)	B(t)	A (t+1)	B (t+1)	Z	J_A	K_A	J_B	K_B
0	0	0							
0	0	1							
0	1	0							
0	1	1							
1	0	0							
1	0	1							
1	1	0							
1	1	1							



Q(t)Q(t+1)	00	01	11	10
J,K	0,X	1,X	X,0	X,1

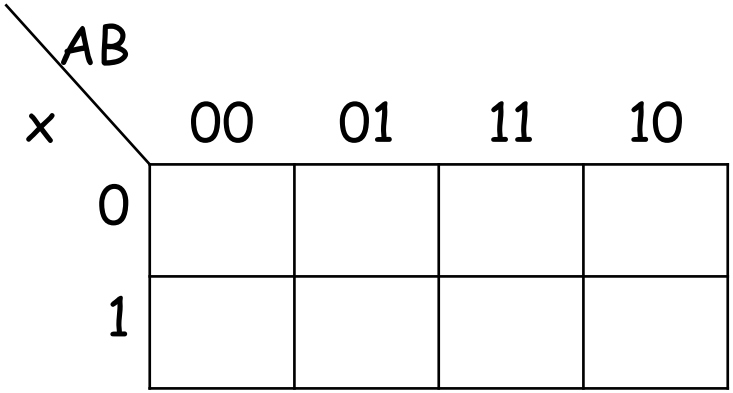
x(t)	A(t)	B(t)	A (t+1)	B (t+1)	Z	J_A	K_A	J_B	K_B
0	0	0	0	0	0				
0	0	1	1	0	0				
0	1	0	0	0	0				
0	1	1	1	0	0				
1	0	0	0	1	0				
1	0	1	0	1	0				
1	1	0	1	1	0				
1	1	1	0	1	1				



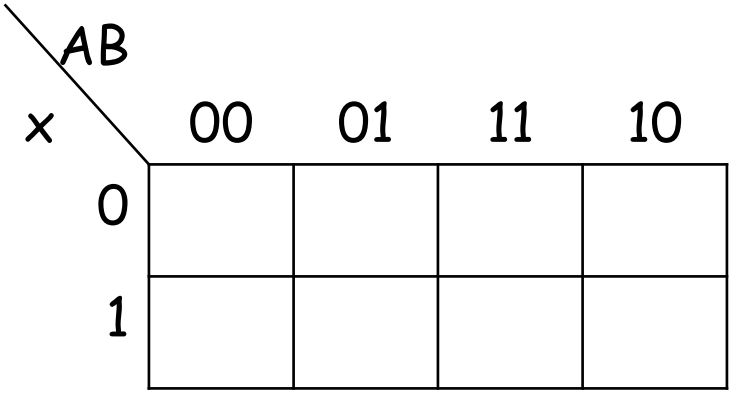
Q(t)Q(t+1)	00	01	11	10
J,K	0,X	1,X	X,0	X,1

x(t)	A(t)	B(t)	A (t+1)	B (t+1)	Z	J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	X	0	X
0	0	1	1	0	0	1	X	X	1
0	1	0	0	0	0	X	1	0	X
0	1	1	1	0	0	X	0	X	1
1	0	0	0	1	0	0	X	1	X
1	0	1	0	1	0	0	X	X	0
1	1	0	1	1	0	X	0	1	X
1	1	1	0	1	1	X	1	X	0

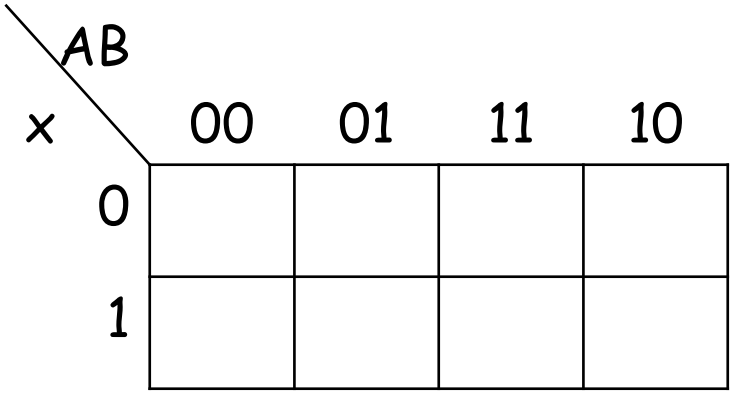
Design Example - 4



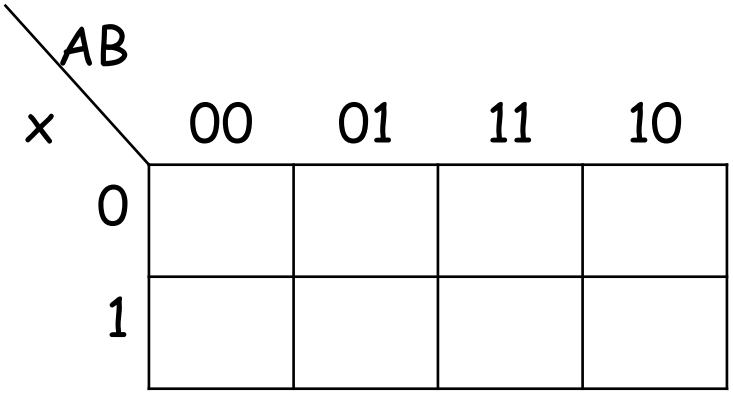
$J_A =$



$K_A =$



$J_B =$



$K_B =$

Design Example - 4

x \ AB	00	01	11	10
0	0	1	X	X
1	0	0	X	X

$J_A = x'B$

x \ AB	00	01	11	10
0	X	X	0	1
1	X	X	1	0

$K_A = xB + x'B'$

x \ AB	00	01	11	10
0	0	X	X	0
1	1	X	X	1

$J_B = x$

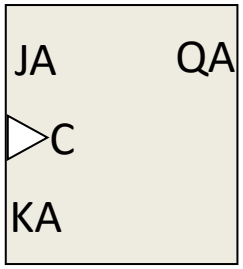
x \ AB	00	01	11	10
0	X	1	1	X
1	X	0	0	X

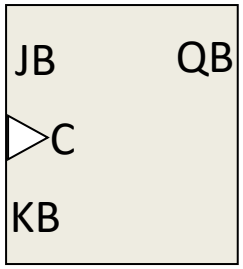
$K_B = x'$

Design Example - 4

$J_A = x'B$ $K_A = xB + x'B'$

$J_B = x$ $K_B = x'$



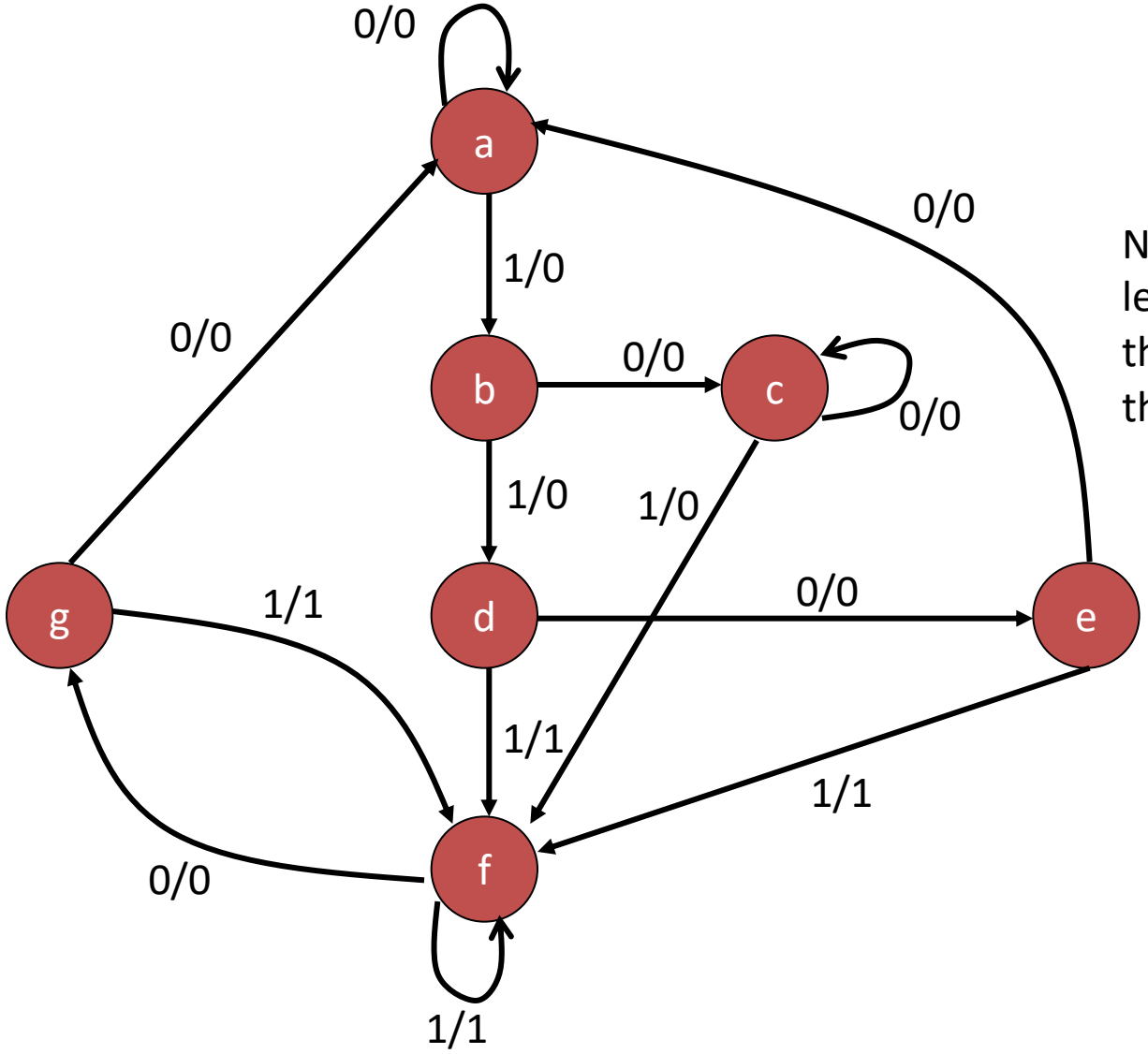


Design Example - 4

State Reduction and Assignment

- In the design process of sequential circuits certain techniques are useful in reducing the circuit complexity
  - state reduction
  - state assignment
- State reduction
  - Fewer states ➔ fewer number of flip-flops
  - m flip-flops ➔ 2^m states
  - Example: m = 5 ➔ $2^m = 32$
    - If we reduce the number of states to 21 do we reduce the number of flip-flops?

Example: State Reduction



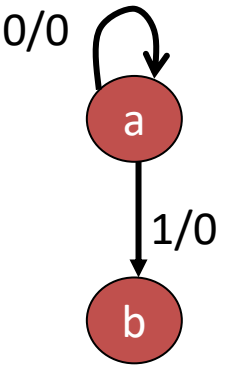
Note that we use letters to designate the states for the time being

Example: State Reduction

state	a	a	b	c	f	g	f	f	g	a	a		
input	0	1	0	1	0	1	1	0	0	0	0		
output	0	0	0	0	0	1	1	0	0	0			

- What is important?
  - not the states
  - but the output values the circuit generates
- Therefore, the problem is to find a circuit
  - **with fewer number of states,**
  - **but that produces the same output pattern for any given input pattern, starting with the same initial state**

State Reduction Technique 1/7



- Step 1: get a state table

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

State Reduction Technique 2/7

- Step 2: Inspect the state table for equivalent states
  - Equivalent states: states
    1. that produce exactly the same output
    2. whose next states are identical
  - for each input combination

State Reduction Technique 3/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

State Reduction Technique 3/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

- States “e” and “g” are equivalent
- One of them can be removed

State Reduction Technique 3/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

- States “e” and “g” are equivalent
- One of them can be removed

State Reduction Technique 4/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

- We keep looking for equivalent states

State Reduction Technique 4/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	f	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

- We keep looking for equivalent states
- >> d & f are now equivalent

State Reduction Technique 5/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	d	0	0
d	e	d	0	1
e	a	d	0	1

- We keep looking for equivalent states

State Reduction Technique 5/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	c	d	0	0
d	e	d	0	1
e	a	d	0	1

- We keep looking for equivalent states
- >> b & c are now equivalent

State Reduction Technique 6/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	b	d	0	0
d	e	d	0	1
e	a	d	0	1

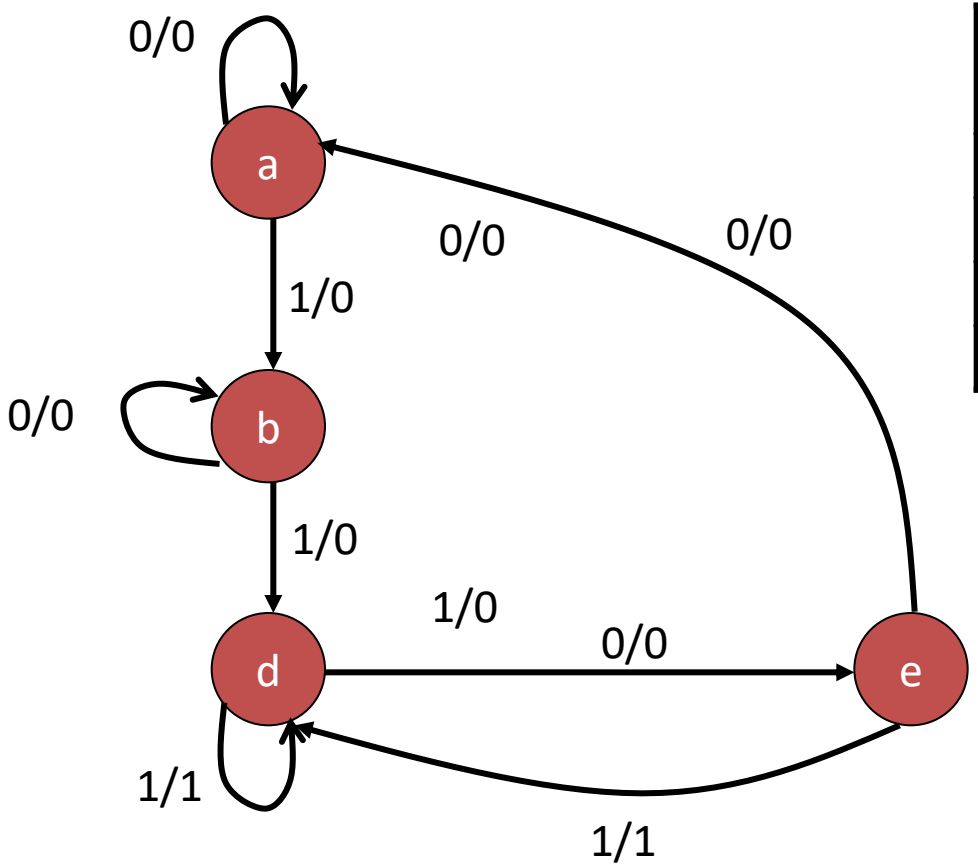
- Any more?

State Reduction Technique 6/7

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	b	d	0	0
d	e	d	0	1
e	a	d	0	1

- Any more?
- NO! We stop when there are no remaining equivalent states

State Reduction Technique 7/7

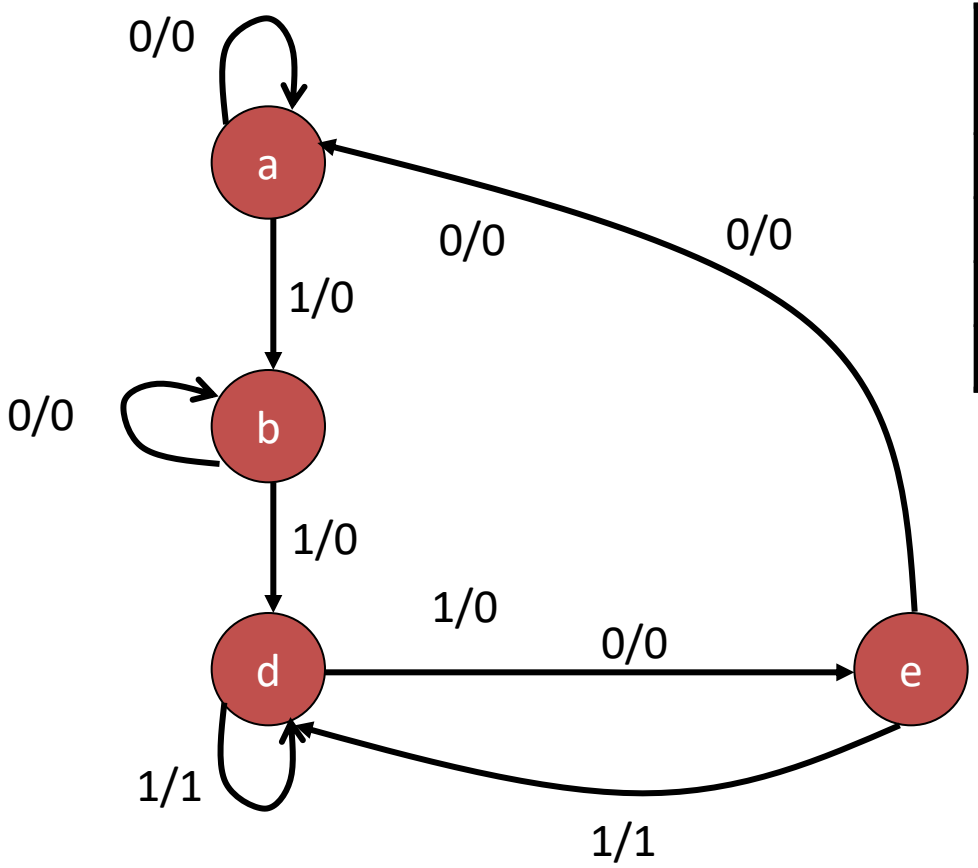


present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	b	d	0	0
d	e	d	0	1
e	a	d	0	1

How many flip flops do we need?

state	a	a	b	b	d	e	d	d	e	a	a		
input	0	1	0	1	0	1	1	0	0	0	0		
output	0	0	0	0	0	1	1	0	0	0			

State Reduction Technique 7/7



present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	b	d	0	0
d	e	d	0	1
e	a	d	0	1

Now we only need 2 flip-flops

state	a	a	b	b	d	e	d	d	e	a	a		
input	0	1	0	1	0	1	1	0	0	0	0		
output	0	0	0	0	0	1	1	0	0	0			

State Assignments 1/4

- We have to assign binary values to each state
- If we have m states, then we need a code with minimum n bits, where $n = \lceil \log_2 m \rceil$
- There are different ways of encoding
- Example: Eight states: S_0, S_1, S_2, S_3, S_4, S_5 , S_6 , S_7

State	Binary	Gray	One-hot
S_0	000	000	000001
S_1	001	001	000010
S_2	010	011	000100
S_3	011	010	001000
S_4	100	110	010000
S_5	101	111	100000
S_6	111	101	100000
S_7	111	100	100000

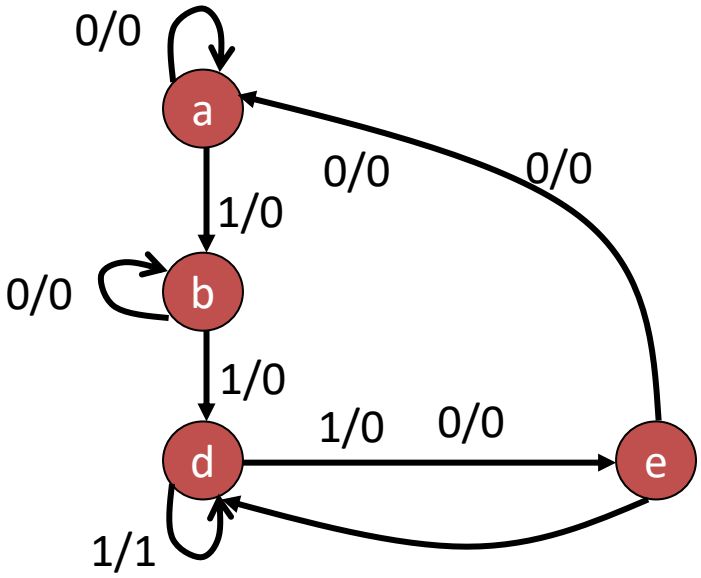
State Assignments 2/4

- The circuit complexity depends on the state encoding (assignment) scheme
- Previous example: **Binary state encoding**

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
(a) 00	00	01	0	0
(b) 01	01	10	0	0
(d) 10	11	10	0	1
(e) 11	00	10	0	1

State Assignments 3/4

- **Gray encoding**



present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
(a) 00	00	01	0	0
(b) 01	01	11	0	0
(d) 11	10	11	0	1
(e) 10	00	11	0	1

State Assignments 4/4

- **One-hot encoding**

present state	next state		Output	
	x = 0	x = 1	x = 0	x = 1
(a) 0001	0001	0010	0	0
(b) 0010	0010	0100	0	0
(d) 0100	1000	0100	0	1
(e) 1000	0001	0100	0	1

- **Binary Encoding**: It is almost used everywhere, in all state machines, by default. Less FFs as compared to One-Hot.

- **One-Hot Encoding**: If you need to design a faster state machine, you would benefit by one-hot-encoding, because you won't have to decode the state.

- **Gray Encoding**: Uses the same number of FFs as Binary Encoding, but it has a great advantage over Binary En. in certain cases. Because it has a hamming distance of 1 between two codes, it is a very reliable count. i.e only one bit changes when the count advances.

Designing Sequential Circuits

- Combinational circuits
  - can be designed given a truth table
- Sequential circuits
  - We need,
    - state diagram

    or
    - state table
  - Two parts
    - flip-flops: number of flip-flops is determined by the number of states
    - combinational part:
      - output equations
      - flip-flop input equations

Design Process

- Once we know the types and number of flip-flops, design process is reduced to design process of combinational circuits
- Therefore, we can apply the techniques of combinational circuit design

Design Steps (cont.)

- The design steps

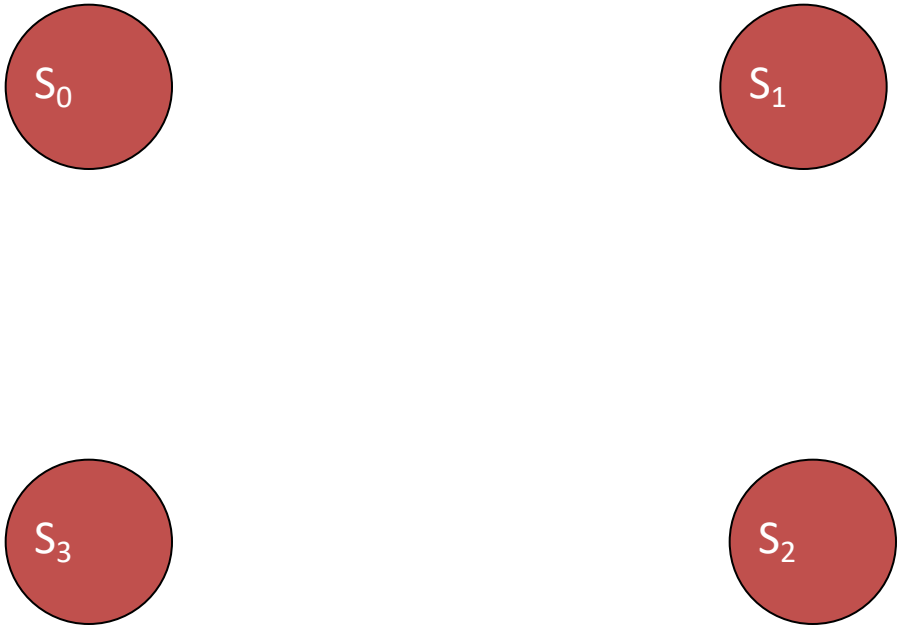
1. Given a verbal description of desired operation, derive the state diagram from that.
2. Reduce the number of states if necessary and possible
3. Do state assignment
4. Obtain the encoded state table
5. Derive the simplified flip-flop input equations
6. Derive the simplified output equations
7. Draw the logic diagram

Example

- Verbal description:
  - "we want a Moore-type sequential circuit that detects three or more consecutive 1's in a string of bits"
  - Input: string of bits of any length
  - Output:
    - "1" if the circuit detects the pattern in the string
    - "0" otherwise

Example: State Diagram

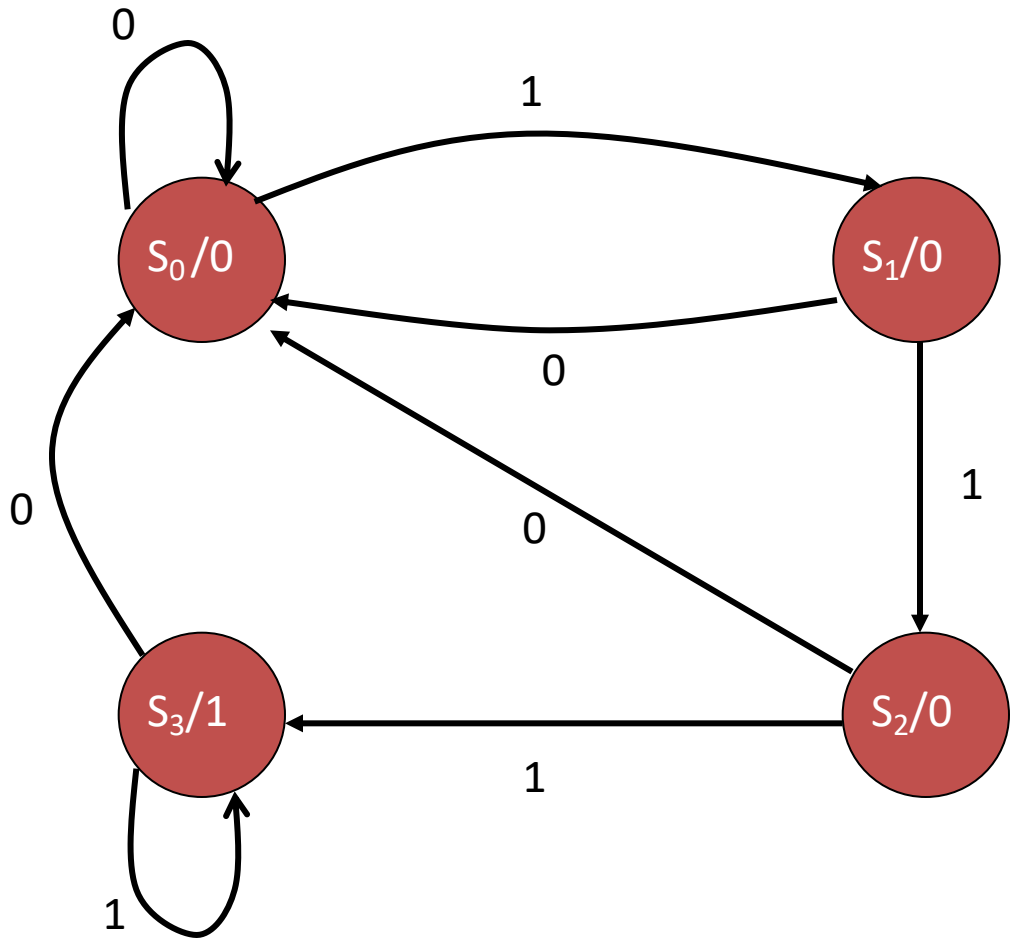
- Step 1: Derive the state diagram



What kind is this?
Mealy or Moore?

Example: State Diagram

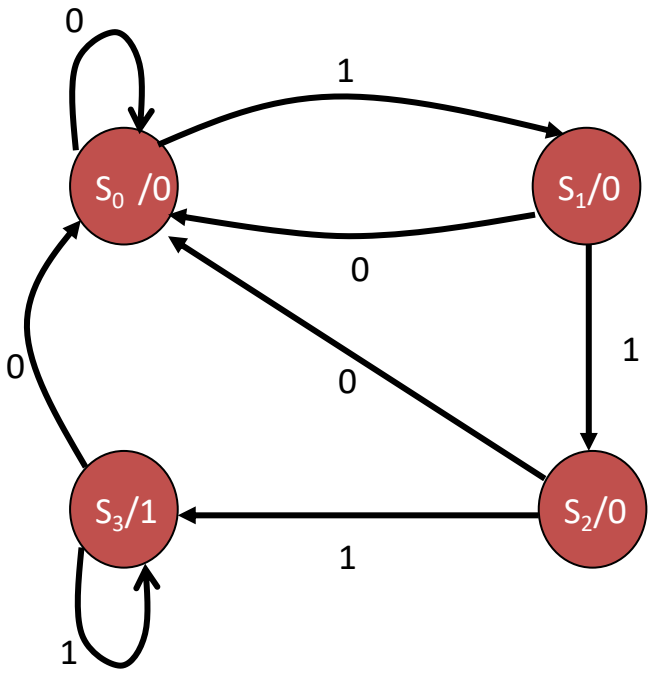
- Step 1: Derive the state diagram



Moore Machine

Synthesis with D Flip-Flops 1/5

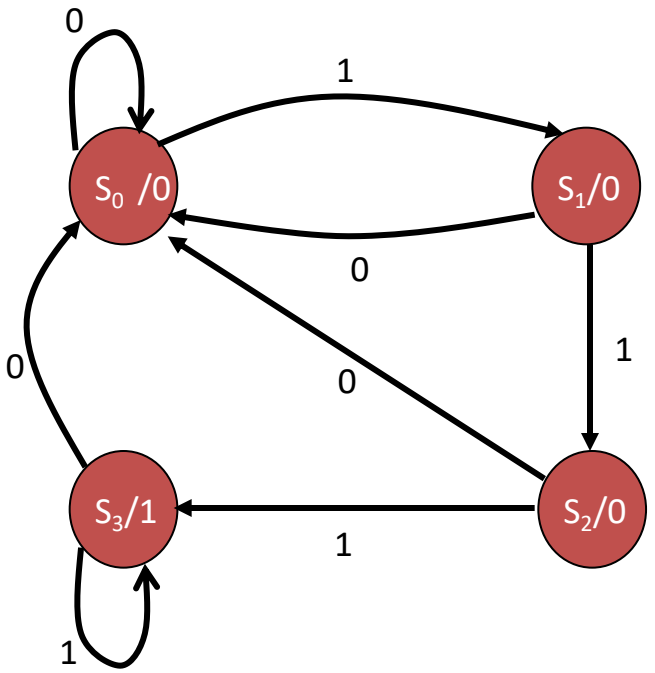
- The number of flip-flops
  - Four states
  - ? flip-flops
- State reduction
  - not possible in this case
- State Assignment
  - Use binary encoding
    - s_0 → 00
    - s_1 → 01
    - s_2 → 10
    - s_3 → 11



Synthesis with D Flip-Flops 2/5

- Step 4: Obtain the state table

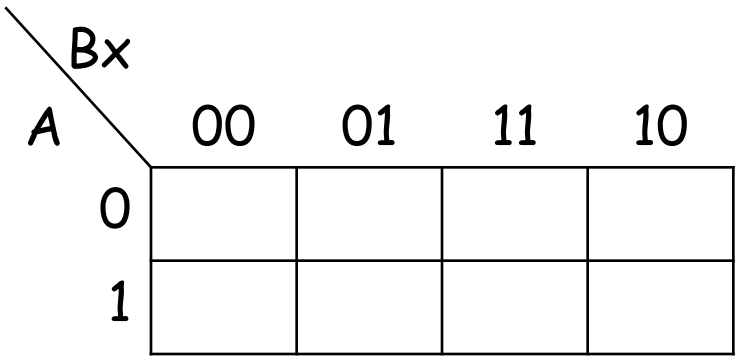
Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			



Synthesis with D Flip-Flops 3/5

- Step 5: Choose the flip-flops
  - D flip-flops
- Step 6: Derive the simplified flip-flop input equations
  - Boolean expressions for D_A and D_B

Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1



D_A =

Synthesis with D Flip-Flops 3/5

- Step 5: Choose the flip-flops
  - D flip-flops
- Step 6: Derive the simplified flip-flop input equations
  - Boolean expressions for D_A and D_B

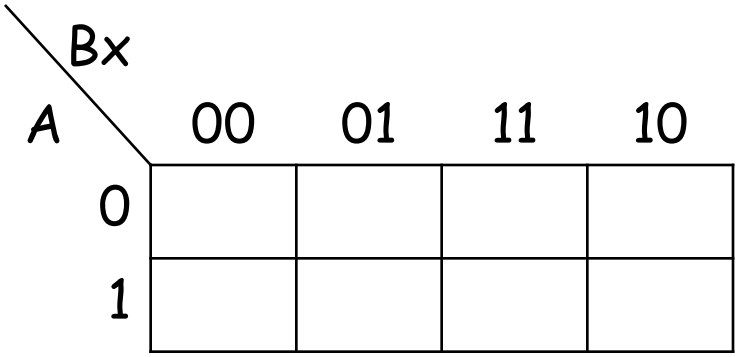
Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

A \ Bx	00	01	11	10
0	0	0	1	0
1	0	1	1	0

$D_A = Ax + Bx$

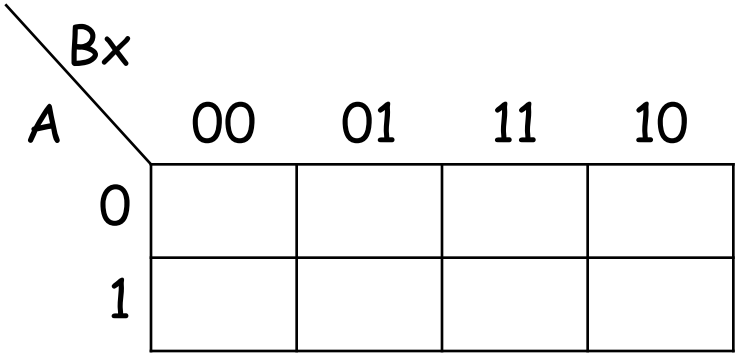
Synthesis with D Flip-Flops 3/5

Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1



D_B =

- Step 7: Derive the simplified output equations
  - Boolean expressions for y.



y =

Synthesis with D Flip-Flops 3/5

Present state		Input	Next state		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

A \ Bx	00	01	11	10
0	0	1	0	0
1	0	1	1	0

$D_B = Ax + B'x$

- Step 7: Derive the simplified output equations
  - Boolean expressions for y.

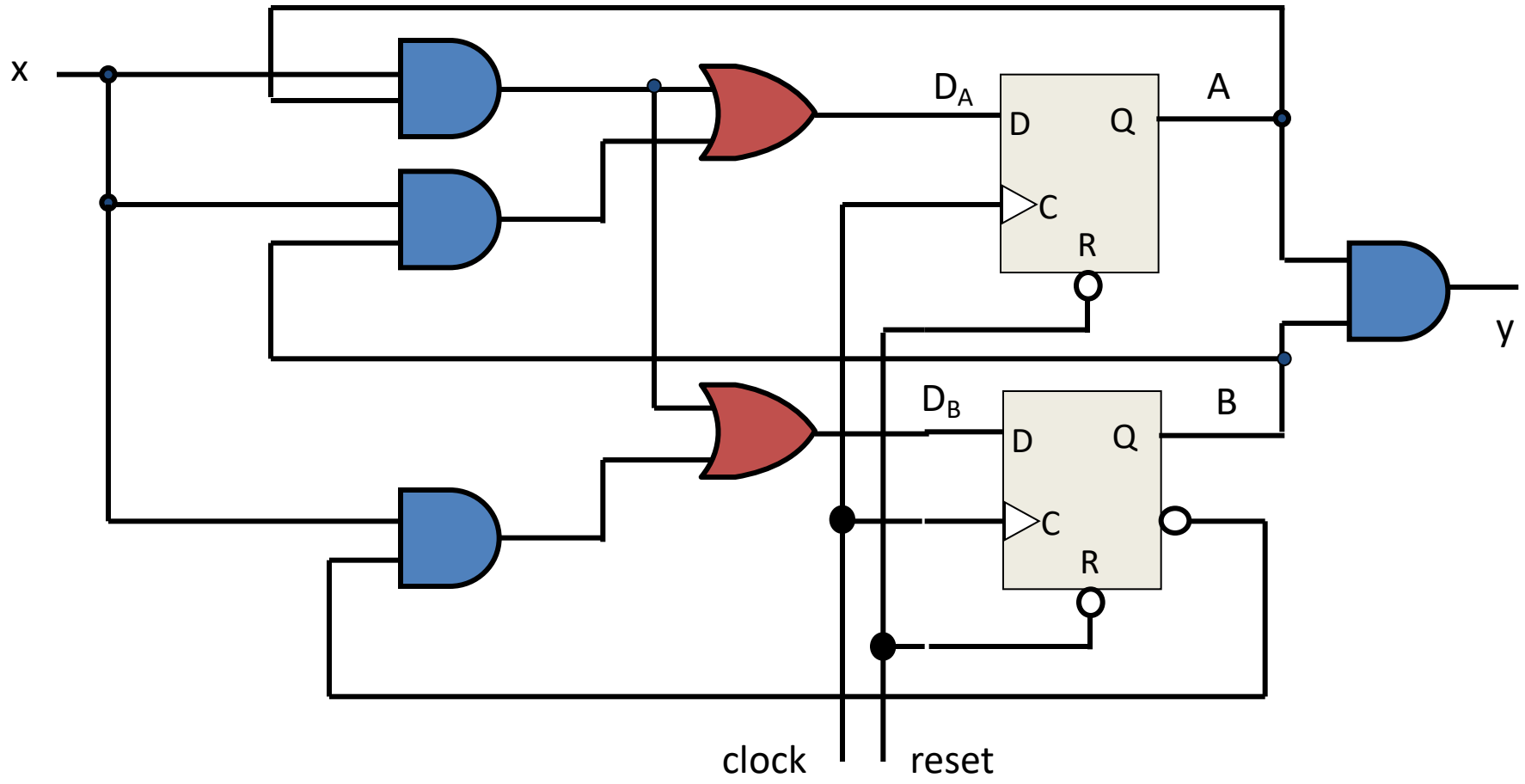
A \ Bx	00	01	11	10
0	0	0	0	0
1	0	0	1	1

$y = AB$

Synthesis with D Flip-Flops 5/5

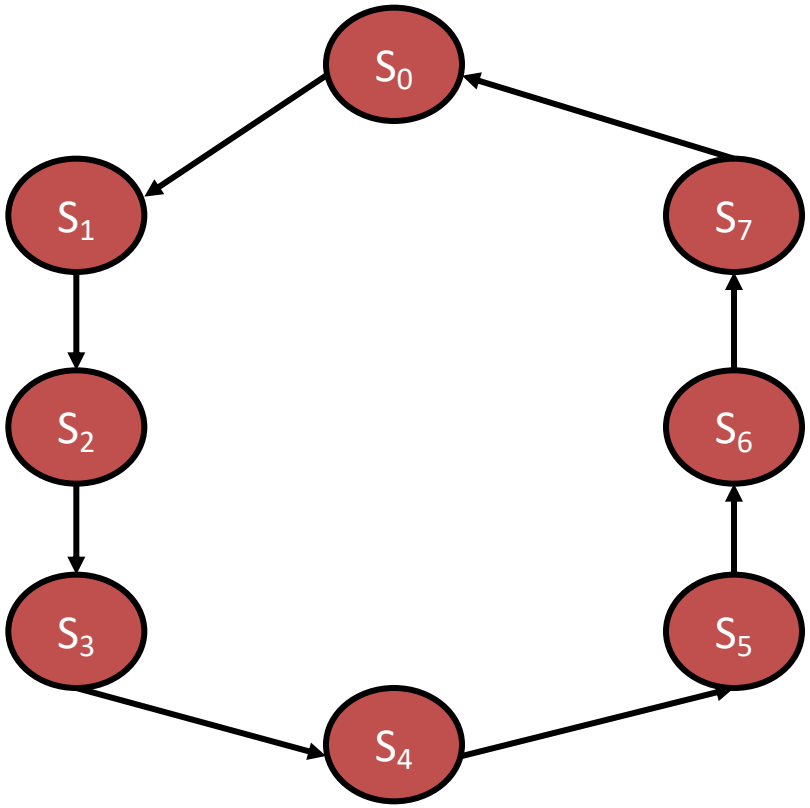
- Step 8: Draw the logic diagram

$D_A = Ax + Bx$ $D_B = Ax + B'x$ $y = AB$



Synthesis with T Flip-Flops 1/4

- Example: 3-bit binary counter with T flip-flops



State Diagram

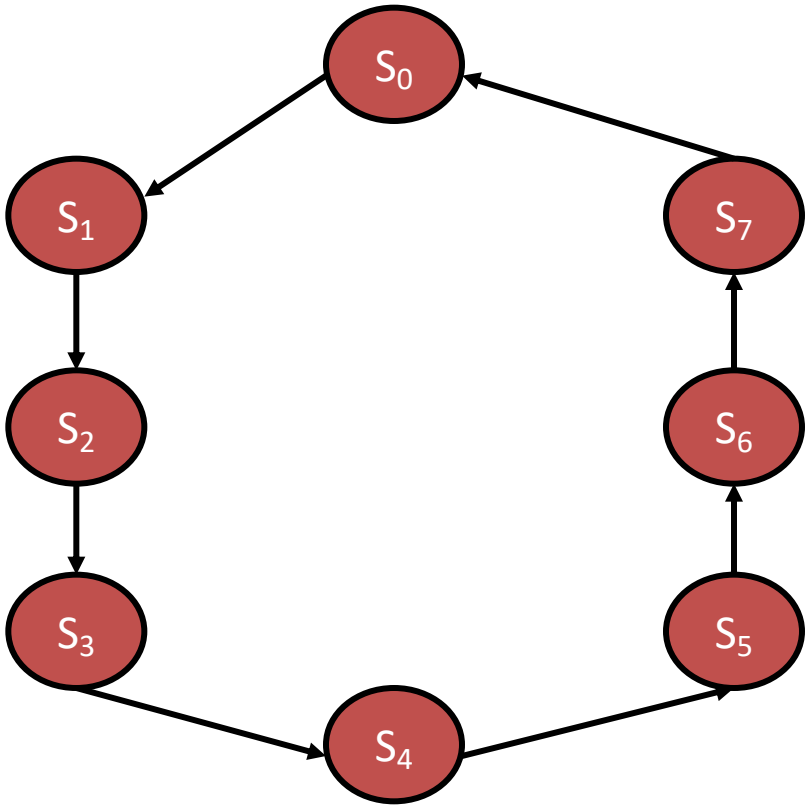
How many flip-flops?

State assignments:

- S_0 → 000
- S_1 → 001
- S_2 → 010
- ...
- S_7 → 111

Synthesis with T Flip-Flops 1/4

- Example: 3-bit binary counter with T flip-flops
  0→1→2→ … → 7 → 0 → 1 → 2



State Diagram

How many flip-flops?

State assignments:

- S_0 → 000
- S_1 → 001
- S_2 → 010
- . . .
- S_7 → 111

Synthesis with T Flip-Flops 2/4

- **State Table**

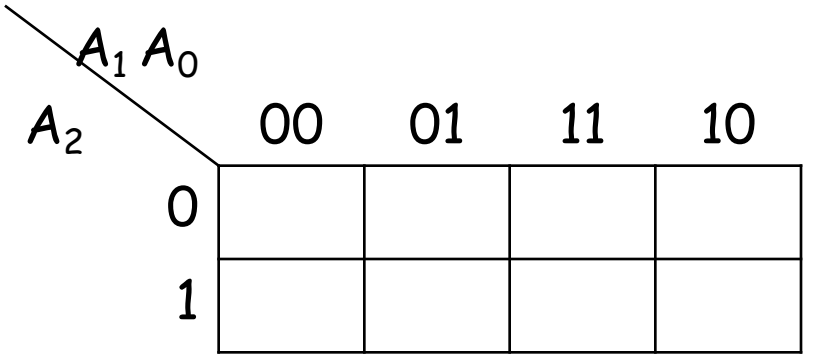
Q(t)Q(t+1)	00	01	11	10
T	0	1	0	1

present state			next state			FF inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_2	T_1	T_0
0	0	0	0	0	1			
0	0	1	0	1	0			
0	1	0	0	1	1			
0	1	1	1	0	0			
1	0	0	1	0	1			
1	0	1	1	1	0			
1	1	0	1	1	1			
1	1	1	0	0	0			

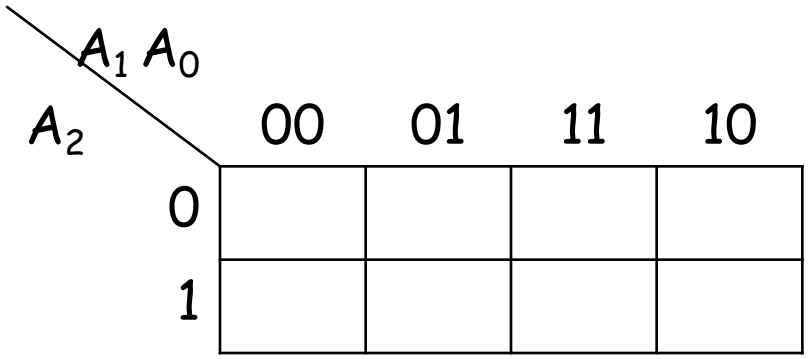
Synthesis with T Flip-Flops 3/4

Present state			FF inputs		
A_2	A_1	A_0	T_2	T_1	T_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	0	0	1
1	1	1	1	1	1

- Flip-Flop input equations



T_2 =



T_1 =

T_0 = ?

Synthesis with T Flip-Flops 3/4

Present state			FF inputs		
A_2	A_1	A_0	T_2	T_1	T_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	0	0	1
1	1	1	1	1	1

- Flip-Flop input equations

A_2 \ A_1A_0	00	01	11	10
0	0	0	1	0
1	0	0	1	0

$T_2 = A_1A_0$

A_2 \ A_1A_0	00	01	11	10
0	0	1	1	0
1	0	1	1	0

$T_1 = A_0$

$T_0 = 1$

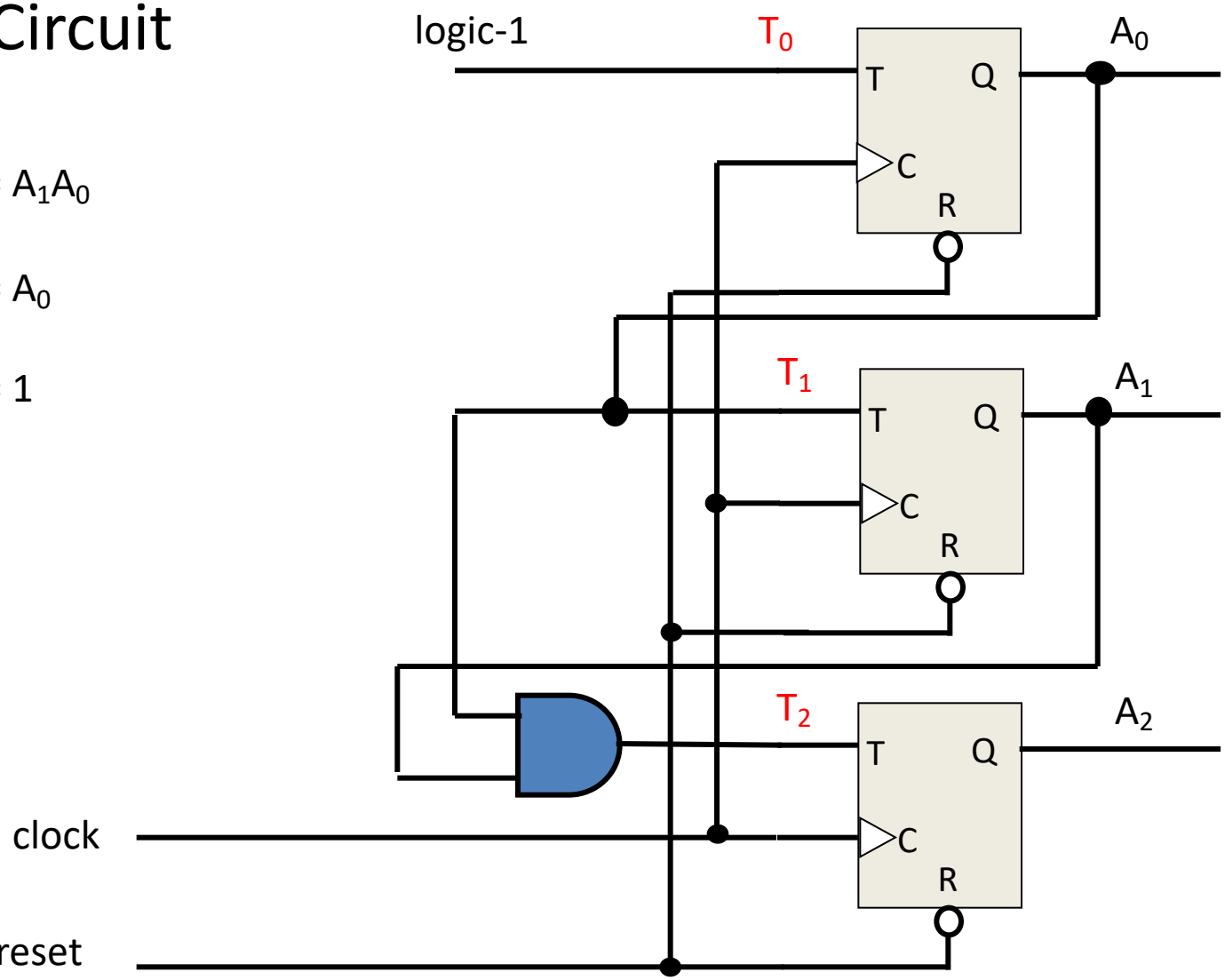
Synthesis with T Flip-Flops 4/4

- Circuit

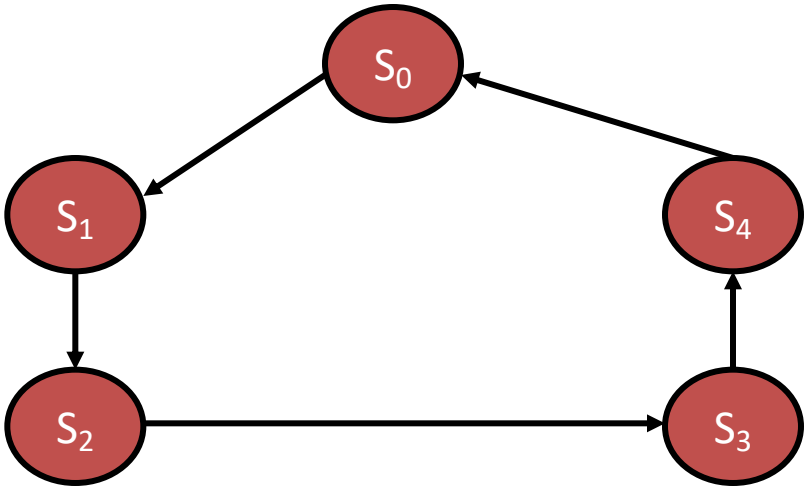
$T_2 = A_1A_0$

$T_1 = A_0$

$T_0 = 1$



Unused States

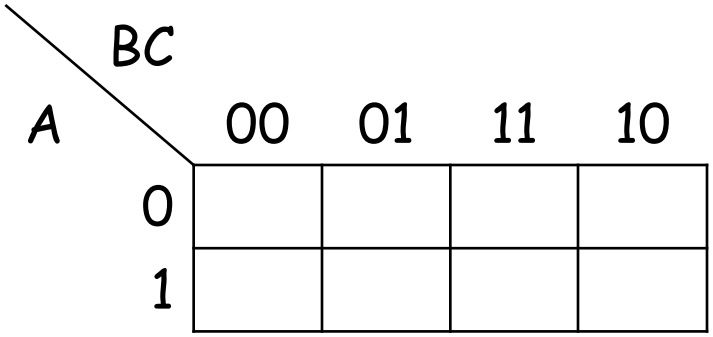


Modulo-5 counter

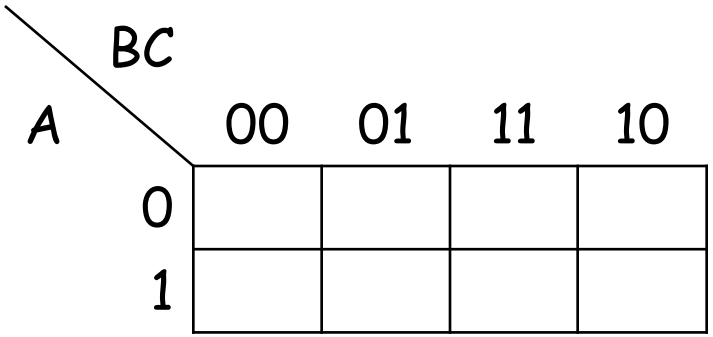
Present State			Next State		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0

Example: Unused States 1/4

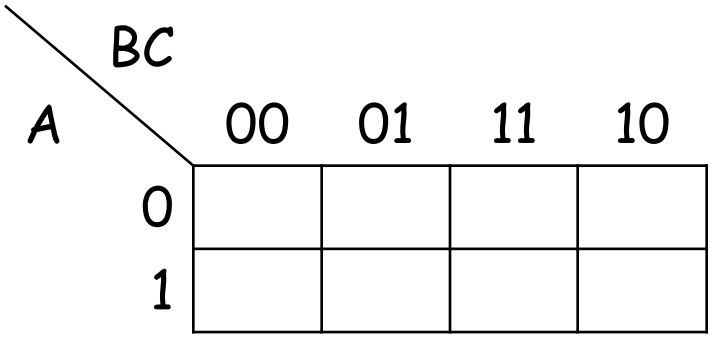
Present State			Next State		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0



A(t+1) =



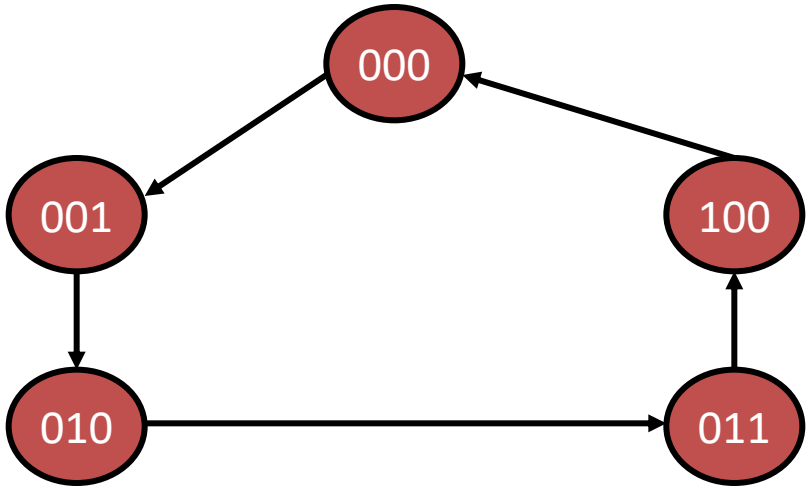
B(t+1) =



C(t+1) =

Example: Unused States 2/4

Present State			Next State		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1			
1	1	0			
1	1	1			



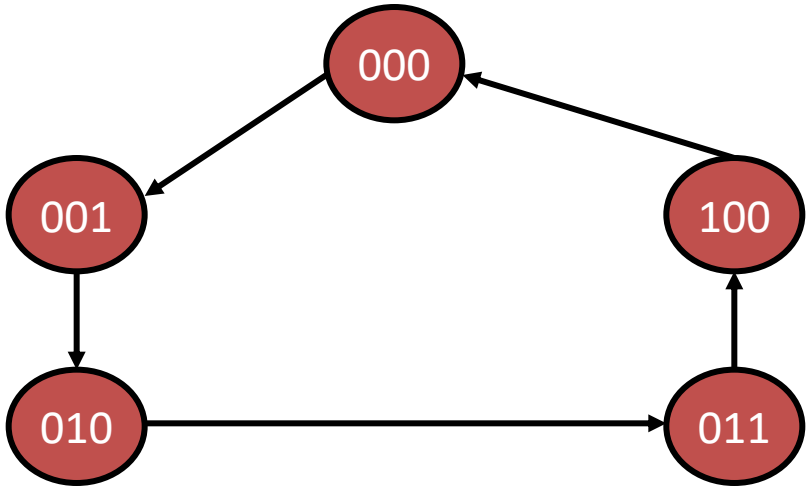
$A(t+1) = BC$

$B(t+1) = B \oplus C$

$C(t+1) = A'C'$

Example: Unused States 2/4

Present State			Next State		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	0	0



$A(t+1) = BC$

$B(t+1) = B \oplus C$

$C(t+1) = A'C'$

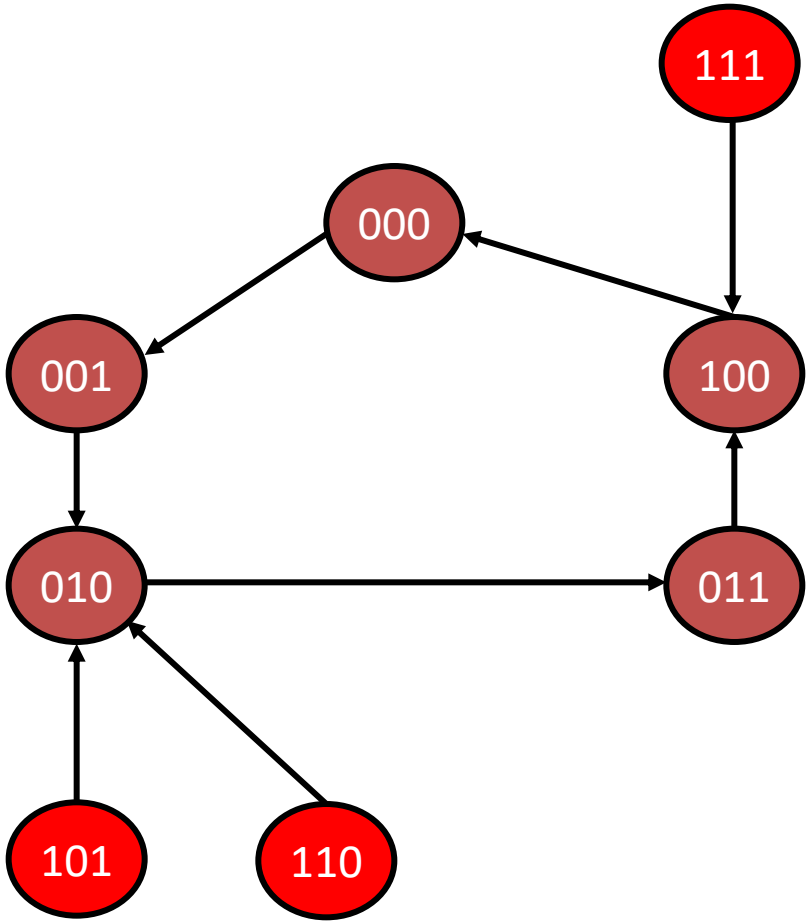
Example: Unused States 2/4

Present State			Next State		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	0	0

$A(t+1) = BC$

$B(t+1) = B \oplus C$

$C(t+1) = A'C'$



Example: Unused States 3/4

- This time <u>not</u> using don't care conditions, instead assigning them all 0s

Present State			Next State		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0

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

A(t+1) = A'BC

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

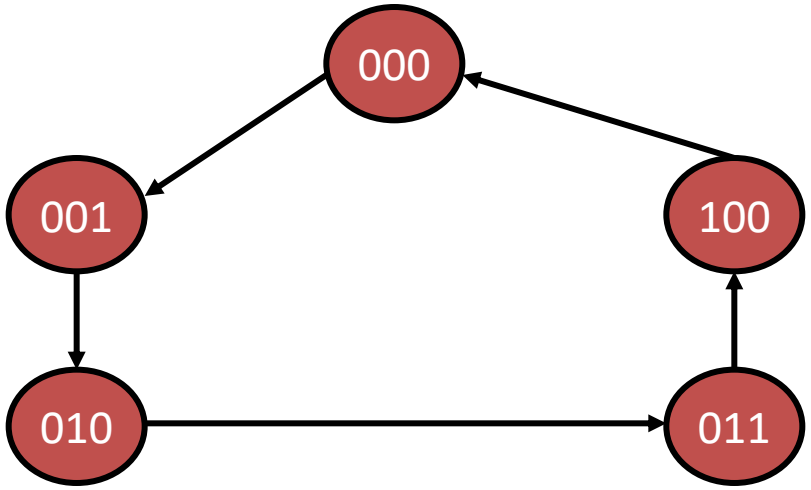
B(t+1) = A'B'C + A'BC'
= A'(B ⊕ C)

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

C(t+1) = A'C'

Example: Unused States 4/4

Present State			Next State		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	**0**	**1**			
1	**1**	**0**			
1	**1**	**1**			



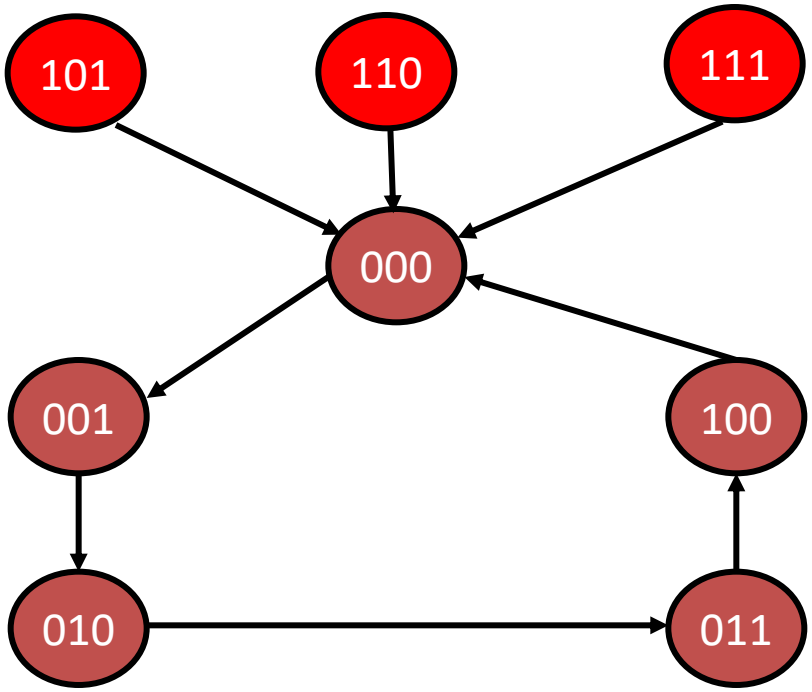
$A(t+1) = A'BC$

$B(t+1) = A'(B \oplus C)$

$C(t+1) = A'C'$

Example: Unused States 4/4

Present State			Next State		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	**0**	**1**			
1	**1**	**0**			
1	**1**	**1**			



$A(t+1) = A'BC$

$B(t+1) = A'(B \oplus C)$

$C(t+1) = A'C'$