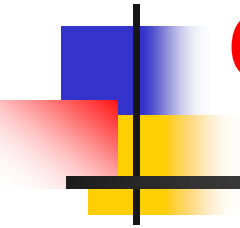
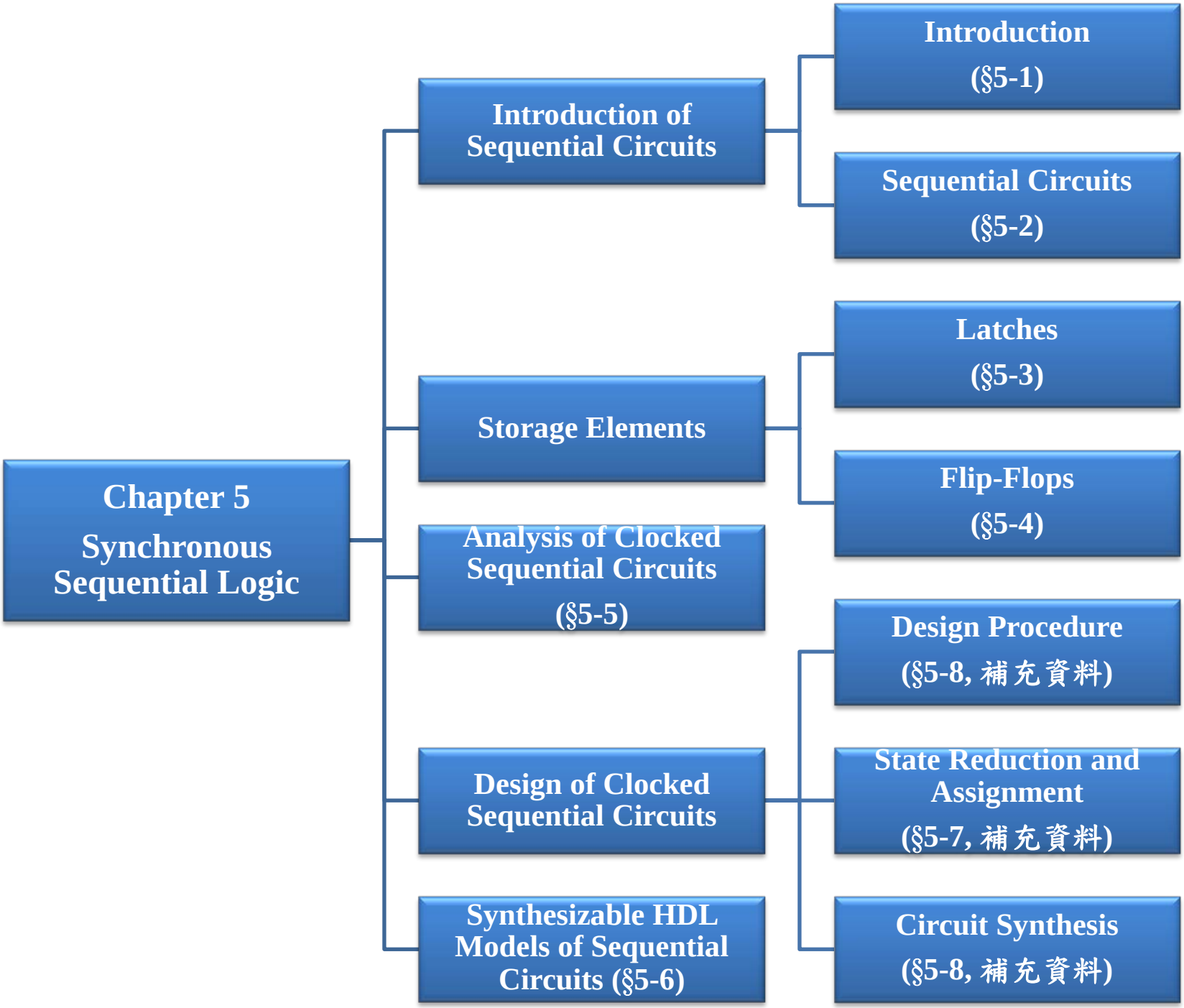


Chapter 5

Synchronous Sequential Logic

J.J. Shann

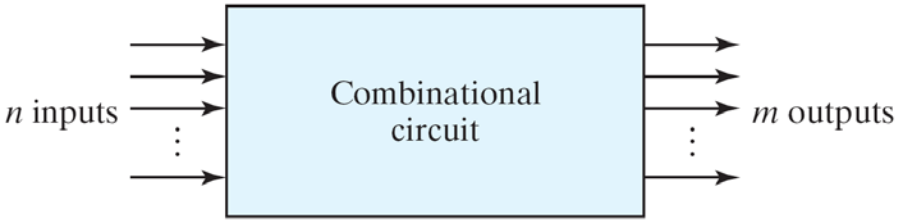
Contents


Chapter 5
Synchronous Sequential Logic
Introduction of Sequential Circuits
Introduction (§5-1)
Sequential Circuits (§5-2)
Storage Elements
Latches (§5-3)
Flip-Flops (§5-4)
Analysis of Clocked Sequential Circuits (§5-5)
Design of Clocked Sequential Circuits
Design Procedure (§5-8, 補充資料)
State Reduction and Assignment (§5-7, 補充資料)
Circuit Synthesis (§5-8, 補充資料)
Synthesizable HDL Models of Sequential Circuits (§5-6)

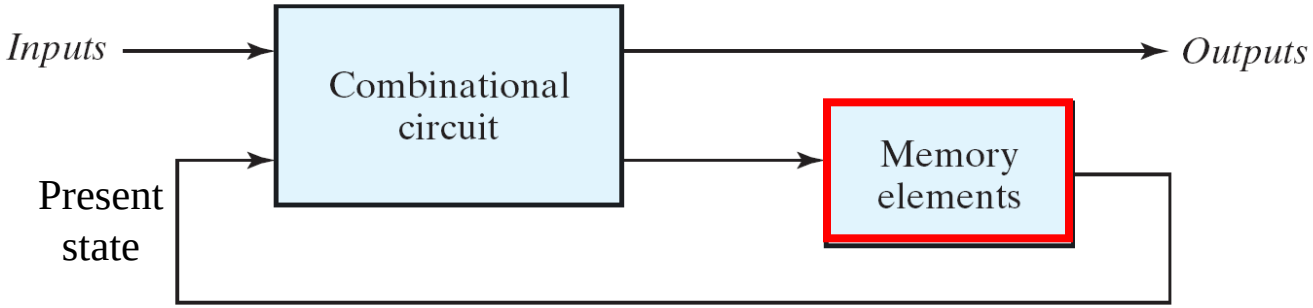
5-1 Introduction

- Combinational circuit: no memory elements
  - The outputs are entirely dependent on the current inputs.

    (inputs) ⇒ (outputs)

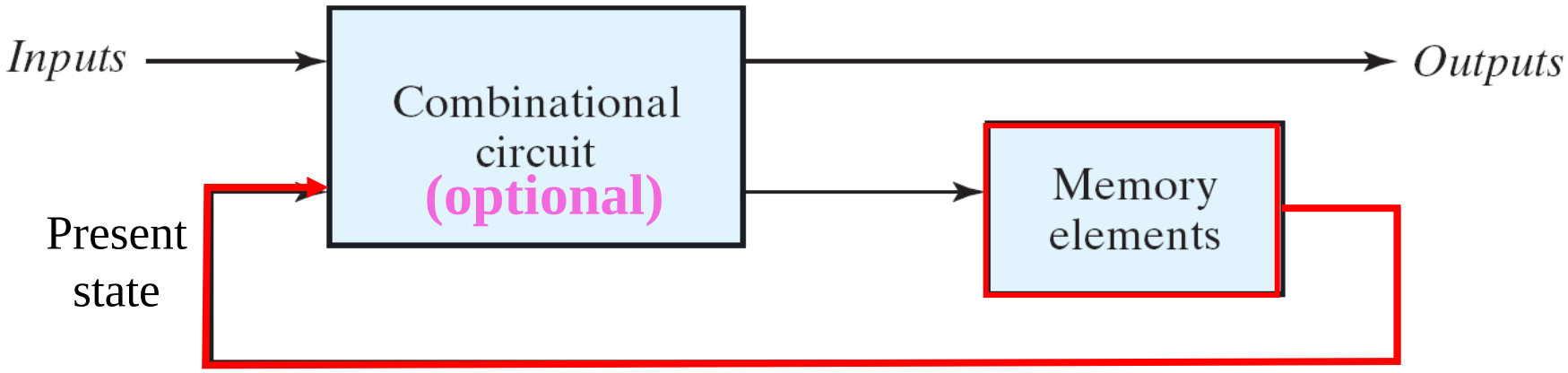


- Sequential circuit:
  - storage elements: devices capable of storing binary information (*state*)
  - (inputs, present state) ⇒ (outputs, next state)



5-2 Sequential Circuits

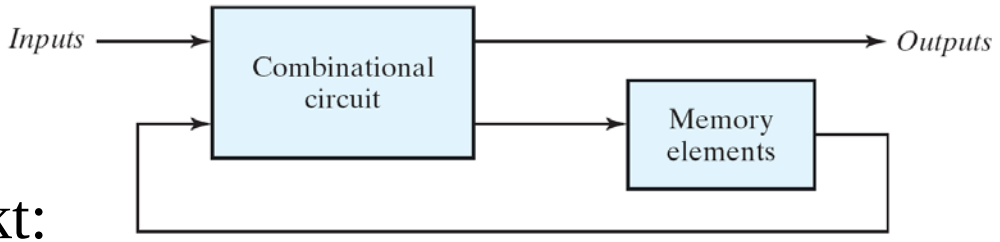
- Sequential circuit:
  - consists of a combinational circuit to which storage elements are connected to form a feedback path.



  - storage elements: devices capable of storing binary information (*state*)
  - (inputs, present state) ⇒ (outputs, next state)

Types of Sequential Circuits

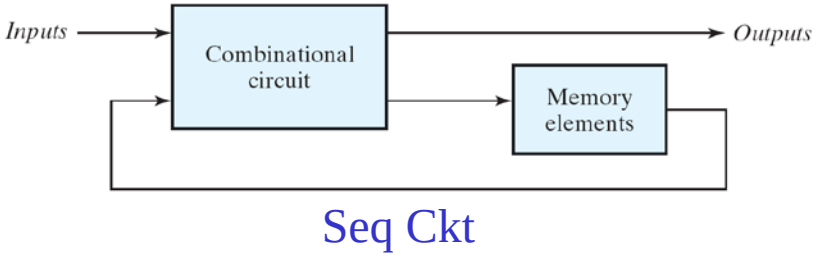
- Classification:
  - depends on the timing of their signals.



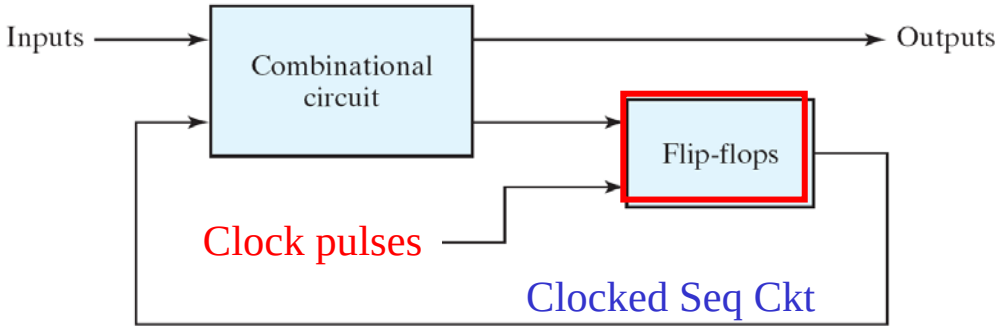
- Two main types:
  1. Synchronous seq ckt:
     - Its behavior can be defined from the knowledge of its signals at discrete instants of time.
     - Storage elements: e.g., **flip-flops** (§5-4)
  2. Asynchronous seq ckt: (Ch9, 4th ed.)
     - Its behavior depends upon the input signals at any instant of time and the order in which the inputs change.
     - Storage elements: e.g., **latches** (§5-3), feedback paths
     - Disadv.: may be unstable & difficult to design

Synchronous Sequential Circuits

- Clocked seq ckts: most commonly used sync seq ckts
  - is syn seq ckts that use clock pulses in the inputs of storage elements
  - has a master-clock generator to generate a periodic train of clock pulses
    - The clock pulses are distributed throughout the system.
    - Storage elements are affected only w/ the arrival of each pulse.
  - Adv.: seldom manifest instability problems
  - Storage elements: flip-flops
    - flip-flop: a binary cell capable of storing one bit of information
    - The state of a flip-flop can change only during a clock pulse transition.
      ⇒ The transition from one state to the next occurs only at predetermined time intervals dictated by the clock pulses.

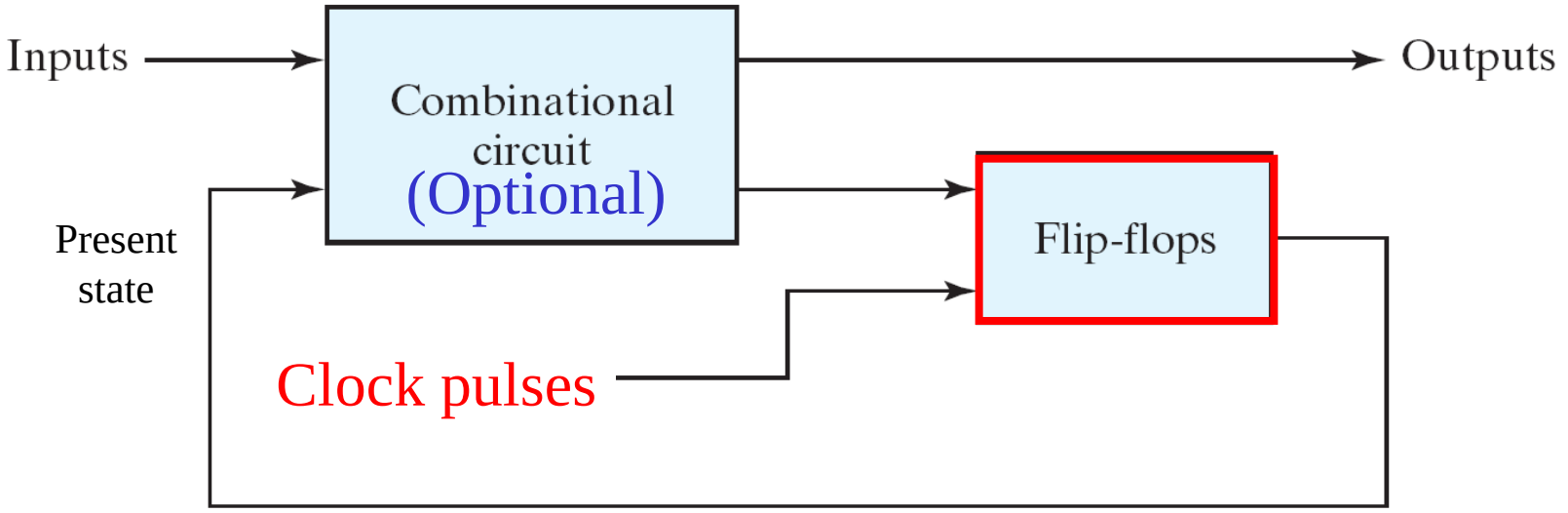


Seq Ckt

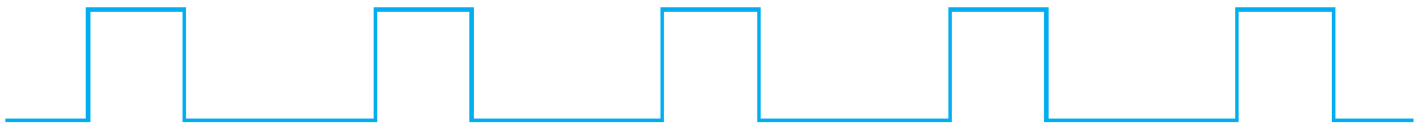


Clocked Seq Ckt

- Block diagram of a clocked sequential ckt:



(a) Block diagram



(b) Timing diagram of clock pulses

* Clock pulse transition: ⌜ or ⌞

5-3 Storage Elements: Latches

- Latches:
  - are asynchronous seq ckts
  - are the basic ckts from which all flip-flops are constructed

A. SR and $\overline{S}\overline{R}$ Latches

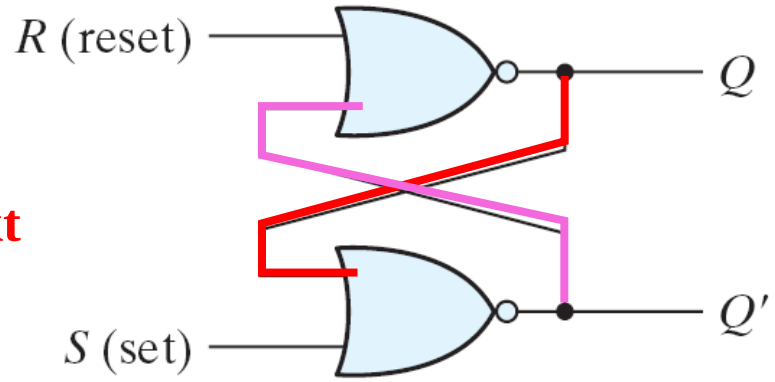
- SR latch
- $\overline{S}\overline{R}$ latch
- SR latch w/ control input

SR Latch

Q: the normal output
Q′ or $\overline{Q}$: the complement output

- SR latch: w/ NOR gates

Async Seq Ckt



(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	(forbidden)

(b) Function table

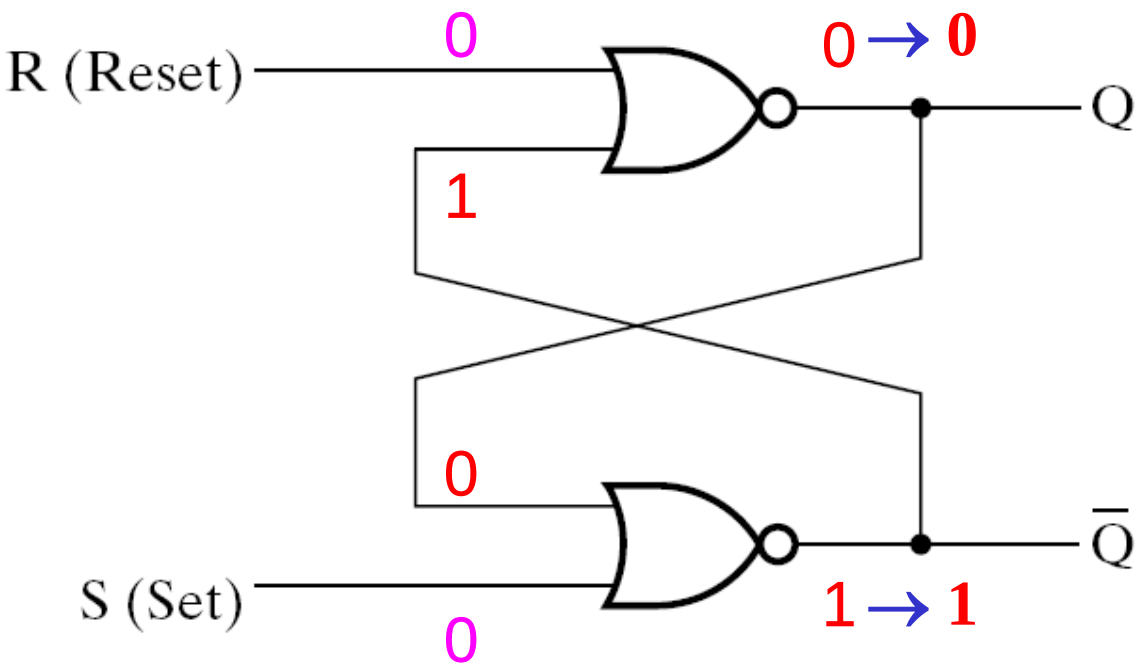
- Useful states:
  - Set state: Q = 1, $\overline{Q}$ = 0
  - Reset state: Q = 0, $\overline{Q}$ = 1
- Undefined states: Q = $\overline{Q}$

S	R	Q^+
0	0	No change ($Q^+ = Q$)
0	1	Reset ($Q^+ = 0$)
1	0	Set ($Q^+ = 1$)
1	1	Indeterminate

Q^+: next state of Q

Characteristic table

(Case 1)

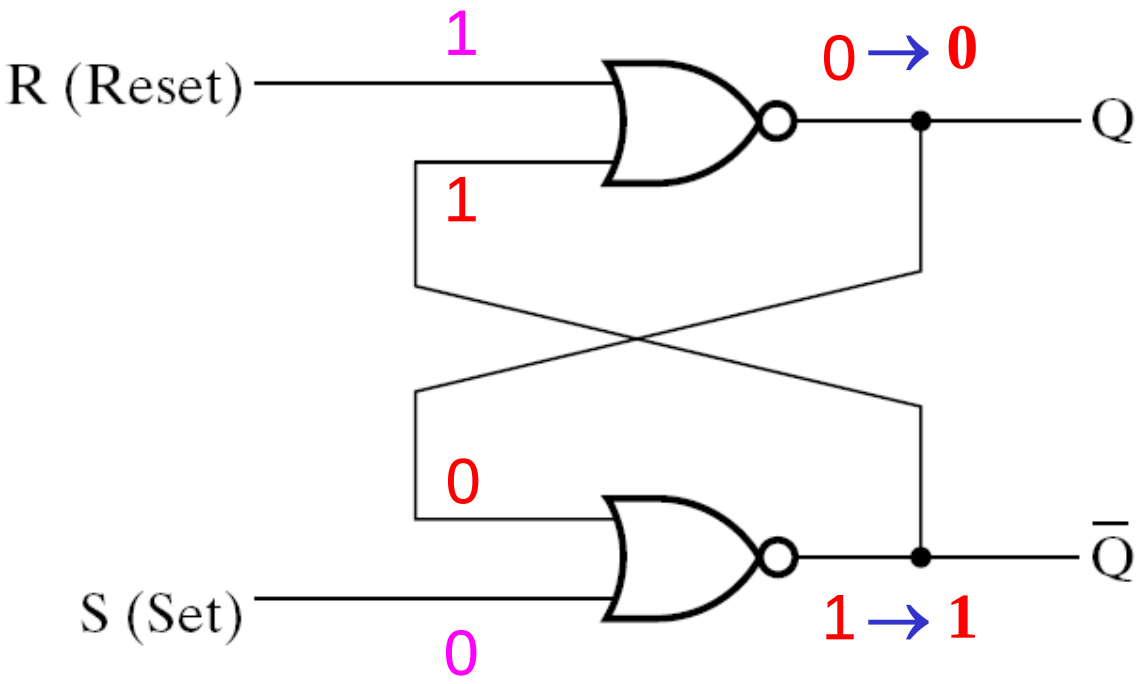


S R	Q $\overline{Q}$	Q^+ $\overline{Q}^+$
0 0	0 1	✓
	1 0	✓
0 1	0 1	
	1 0	
1 0	0 1	
	1 0	
1 1	0 1	
	1 0	

S	R	Q^+
0	0	
0	1	
1	0	
1	1	

Q^+: next state of Q

(Case 2)

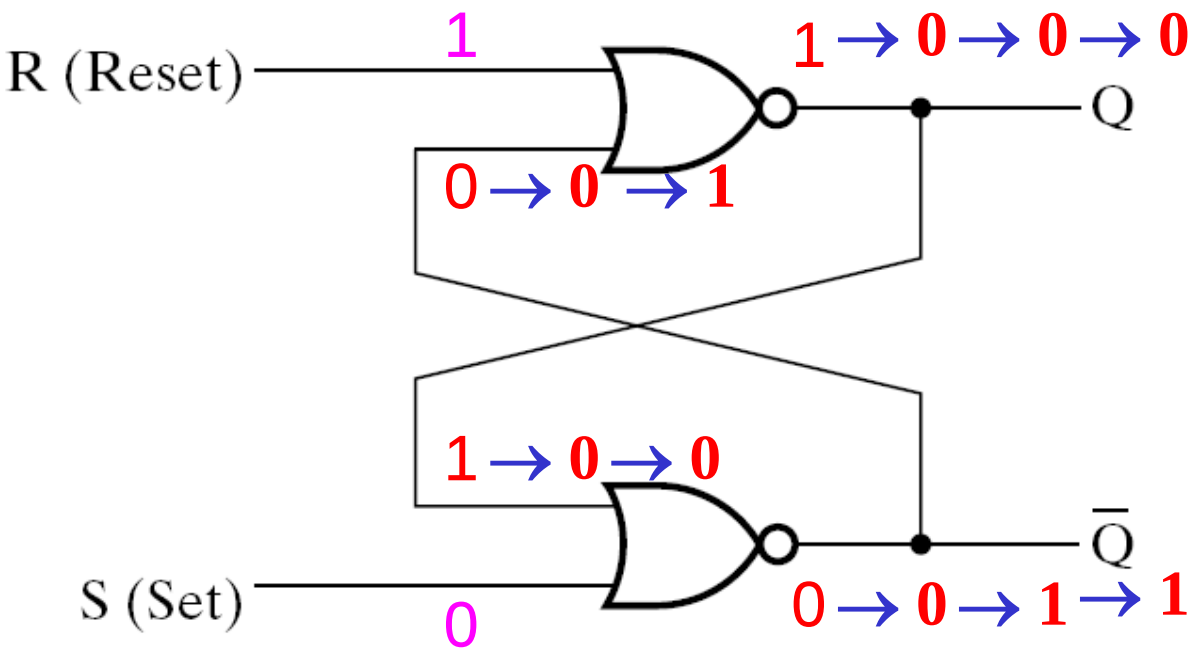


S R	Q $\overline{Q}$	Q^+ $\overline{Q}^+$
0 0	0 1 1 0	0 1 1 0
0 1	**0 1** 1 0	✓
1 0	0 1 1 0	✓
1 1	0 1 1 0	

S	R	Q^+
0	**0**	**No change ($Q^+ = Q$)**
0	**1**	
1	**0**	
1	**1**	

Q^+: next state of Q

(Case 3)

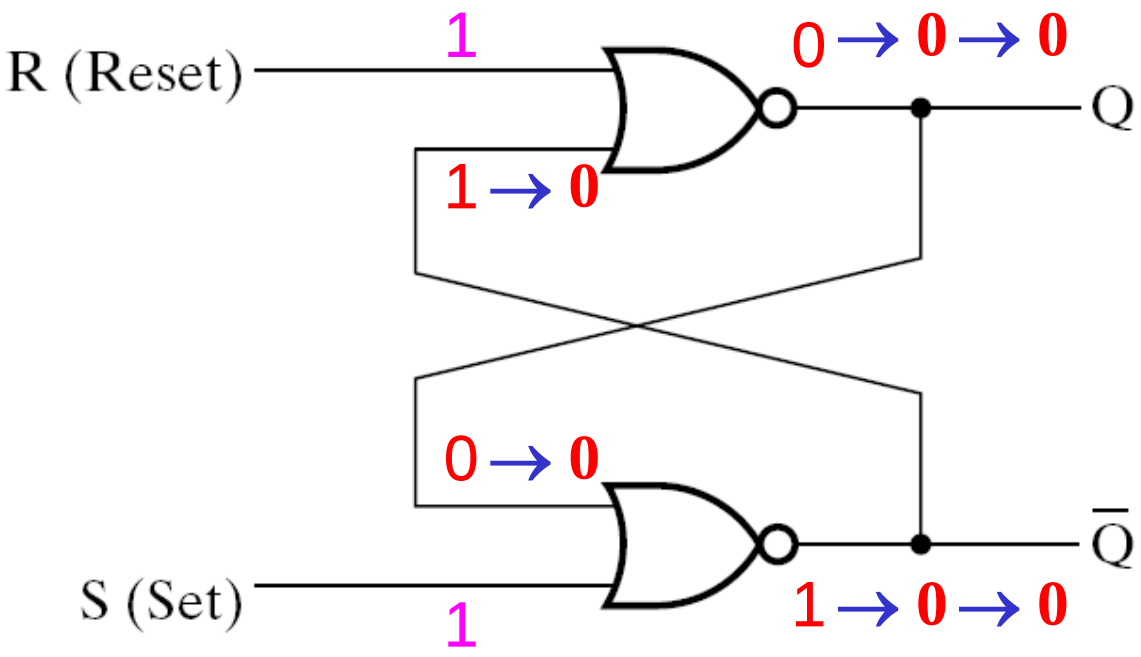


S R	Q $\overline{Q}$	Q^+ $\overline{Q}^+$
0 0	0 1 1 0	0 1 1 0
0 1	0 1 **1 0**	0 1 ✓
1 0	0 1 1 0	✓ 1 0
1 1	0 1 1 0	

S	R	Q^+
0	**0**	**No change ($Q^+ = Q$)**
0	**1**	
1	**0**	
1	**1**	

Q^+: next state of Q

(Case 4)

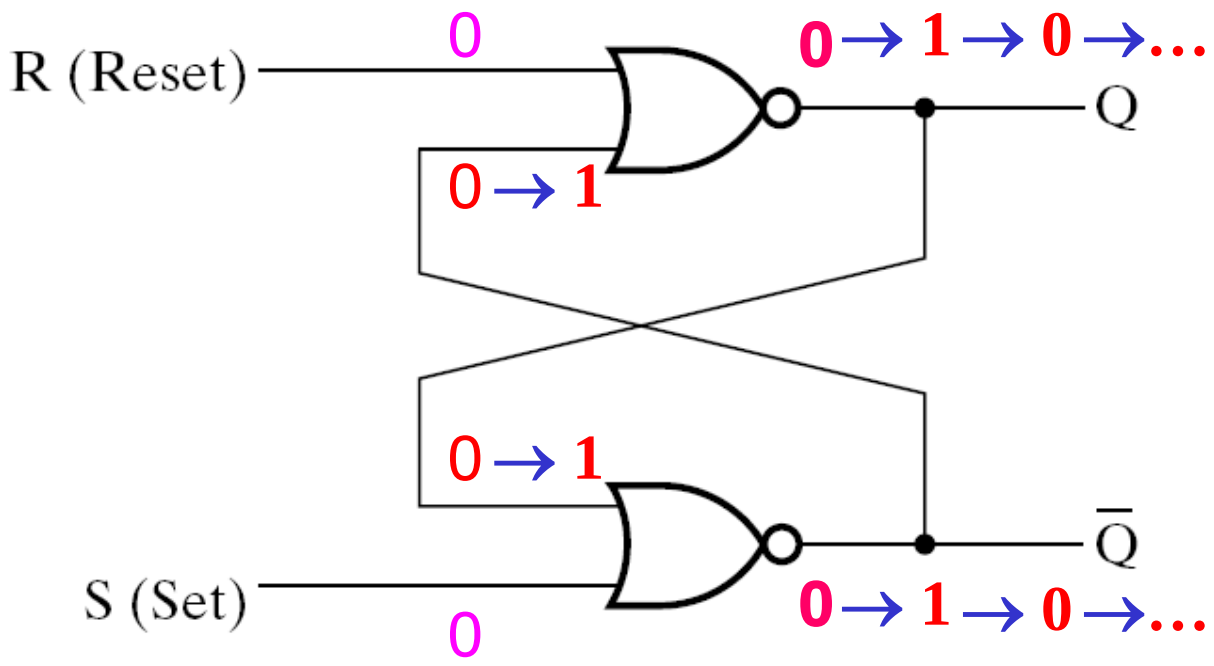


S R	Q $\overline{Q}$	Q^+ $\overline{Q}^+$
0 0	0 1 1 0	0 1 1 0
0 1	0 1 1 0	0 1 0 1
1 0	0 1 1 0	1 0 1 0
1 1	**0 1** 1 0	✓ ✓

S	R	Q^+
0	**0**	**No change ($Q^+ = Q$)**
0	**1**	**Reset ($Q^+ = 0$)**
1	**0**	**Set ($Q^+ = 1$)**
1	**1**	

Q^+: next state of Q

(Case 4+)

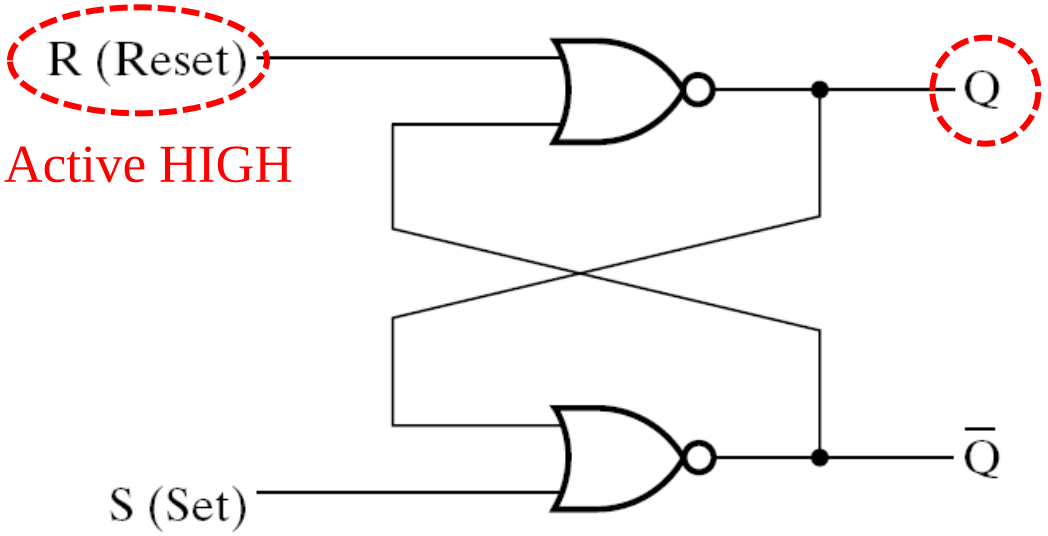


S R	Q $\overline{Q}$	Q^+ $\overline{Q}^+$
0 0	0 1 1 0	0 1 1 0
0 1	0 1 1 0	0 1 0 1
1 0	0 1 1 0	1 0 1 0
1 1	0 1 1 0	0 0 0 0

S	R	Q^+
0	**0**	**No change ($Q^+ = Q$)**
0	**1**	**Reset ($Q^+ = 0$)**
1	**0**	**Set ($Q^+ = 1$)**
1	**1**	

Q^+: next state of Q

- Summary:



S R	Q $\overline{Q}$	Q^+ $\overline{Q}^+$
0 0	0 1 1 0	0 1 1 0
0 1	0 1 1 0	0 1 0 1
1 0	0 1 1 0	1 0 1 0
1 1	0 1 1 0	0 0 0 0

Characteristic table

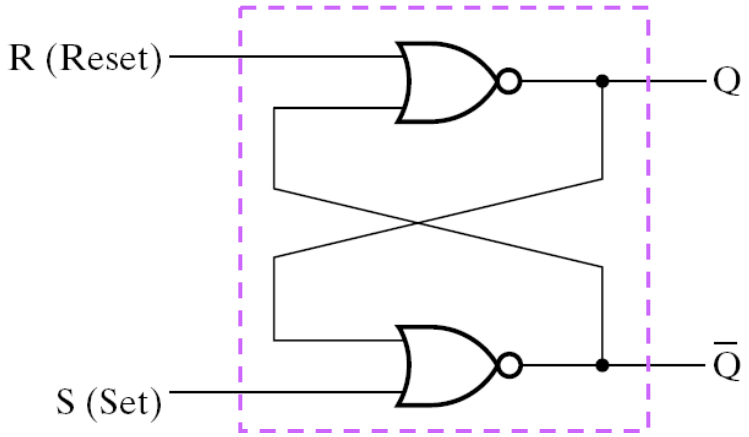
S	R	Q^+
0	0	No change ($Q^+ = Q$)
0	1	Reset ($Q^+ = 0$)
1	0	Set ($Q^+ = 1$)
1	1	Indeterminate

Q^+: next state of Q

Characteristic equation:

$$Q^+ = S'R'Q + SR'$$
$$= SR' + R'Q$$
$$Q^+ = S + R'Q \quad \text{(if } SR = 0\text{)}$$

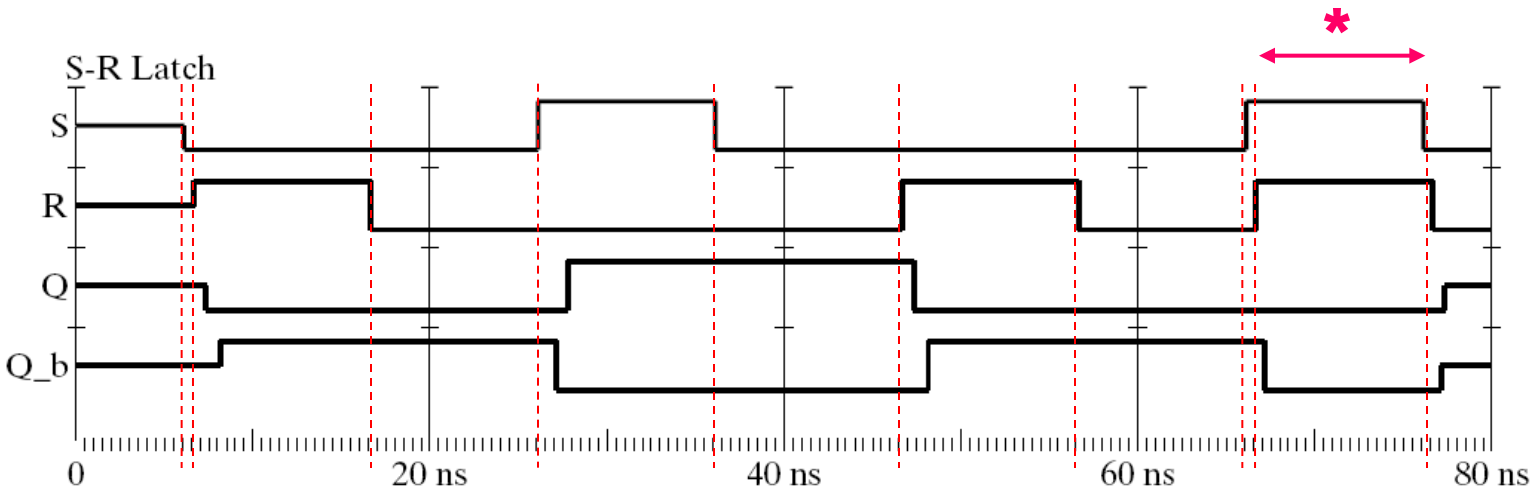
Logic simulation of SR latch behavior



Characteristic table

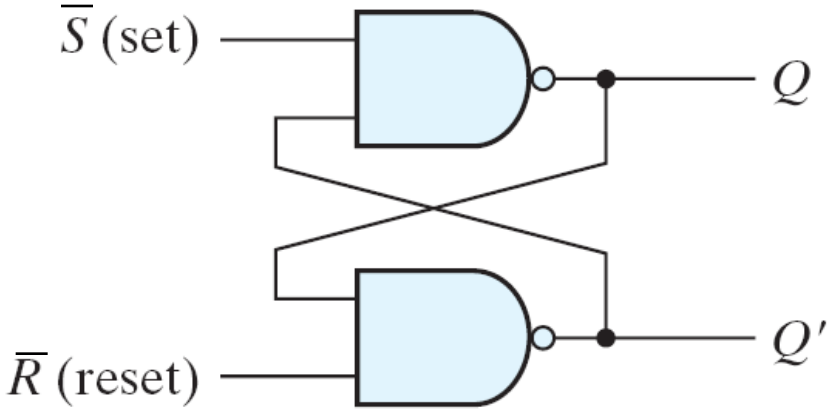
S	R	Q^+
0	0	No change ($Q^+ = Q$)
0	1	Reset ($Q^+ = 0$)
1	0	Set ($Q^+ = 1$)
1	1	Indeterminate

Q^+: next state of Q



$\overline{S}$ $\overline{R}$ Latch

- $\overline{S}$ $\overline{R}$ latch: w/ NAND gates



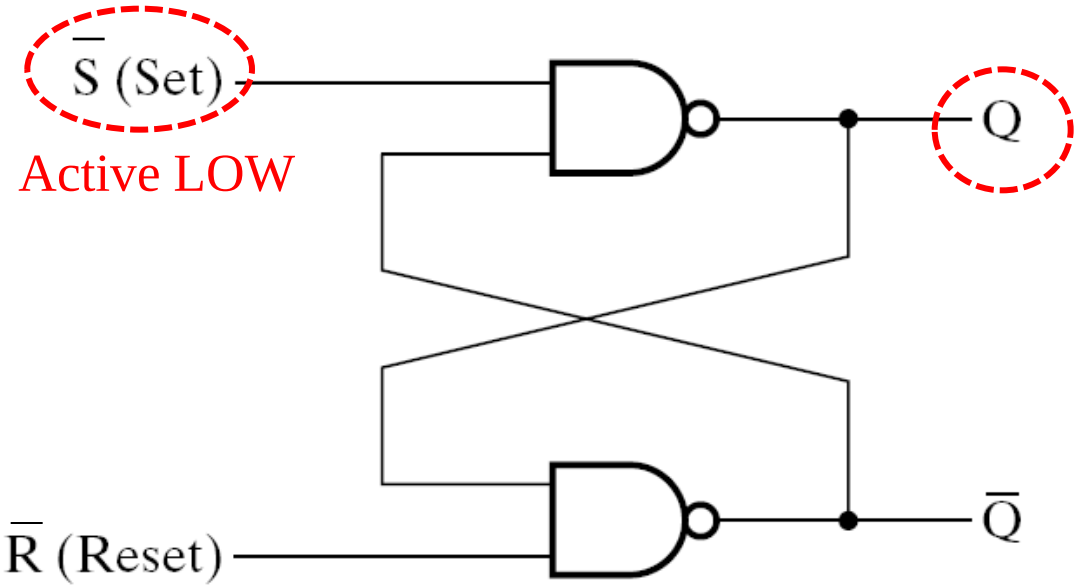
(a) Logic diagram

$\overline{S}$	$\overline{R}$	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)

(b) Function table

Characteristic Table

$\overline{S}$	$\overline{R}$	Q^+
1	1	No change ($Q^+ = Q$)
1	0	Reset ($Q^+ = 0$)
0	1	Set ($Q^+ = 1$)
0	0	Indeterminate



$\overline{S}$ $\overline{R}$	Q $\overline{Q}$	Q^+ $\overline{Q}^+$
1 1	0 1 1 0	
1 0	0 1 1 0	
0 1	0 1 1 0	
0 0	0 1 1 0	

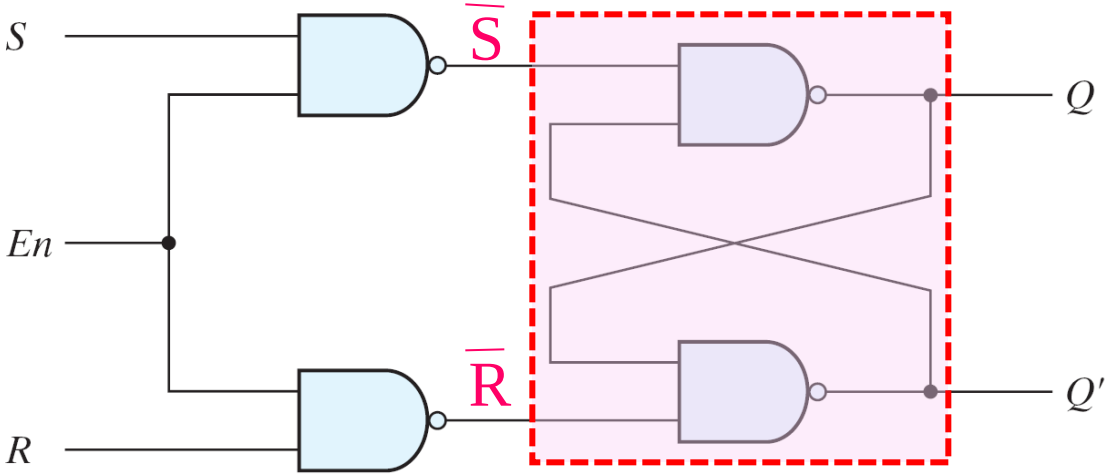
Characteristic Table

$\overline{S}$	$\overline{R}$	Q^+
1	1	No change ($Q^+ = Q$)
1	0	Reset ($Q^+ = 0$)
0	1	Set ($Q^+ = 1$)
0	0	Indeterminate

SR Latch w/ Control Input

$\overline{S}$	$\overline{R}$	Q^+
1	1	No change ($Q^+ = Q$)
1	0	Reset ($Q^+ = 0$)
0	1	Set ($Q^+ = 1$)
0	0	Indeterminate

- SR latch w/ control input:



(a) Logic diagram

enable signal $\overline{S} = \overline{R} = 1$

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

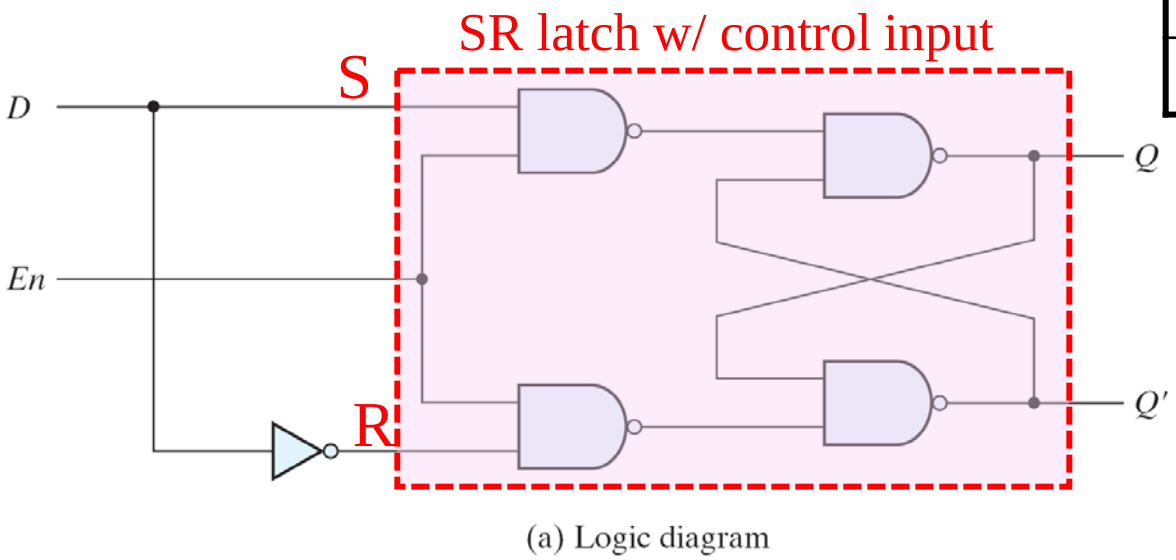
(b) Function table

- The indeterminate condition makes this ckt difficult to manage & it is seldom used in practice.
- It is an important ckt (other latches & flip-flops are constructed from it)

B. D Latch (Transparent Latch)

$Q^+ = D$

- D latch (w/ control input):



(a) Logic diagram

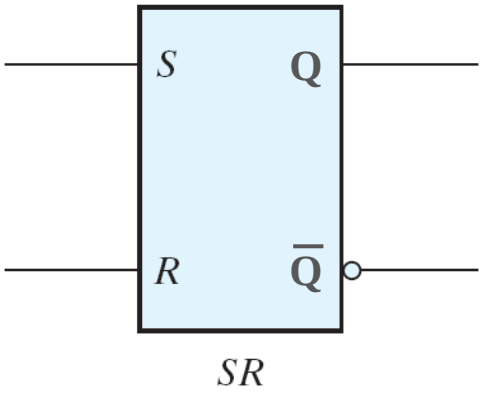
En D	*En* S R	Q+
0 ×	0 × ×	Q (No change)
1 0	1 0 1	0 (Reset)
1 1	1 1 0	1 (Set)

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

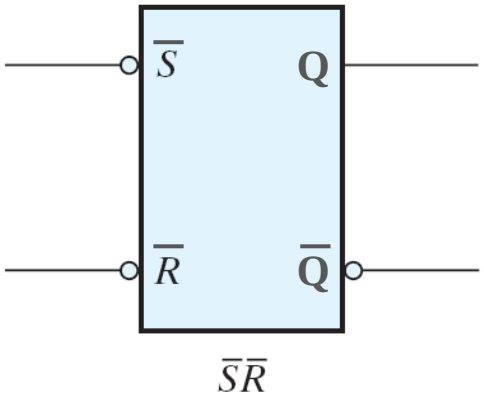
(b) Function table

- Disadv. of a latch w/ control input:
  - The state of latch may keep changing for as long as the control input (*En*) stays in the active level.

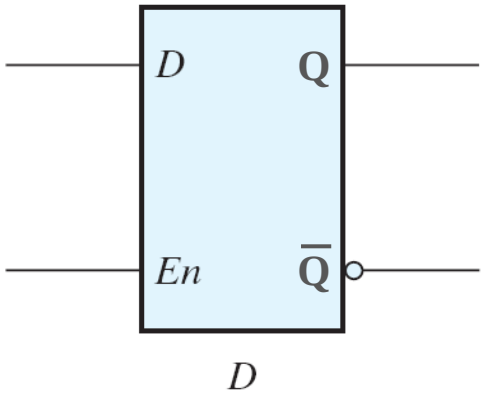
C. Graphic Symbols for Latches



SR Latch



$\overline{S}\overline{R}$ Latch



D Latch
with enable input

5-4 Storage Elements: Flip-Flops

- Review: Latch w/ control input
  - can change state in response to the inputs anytime the control input is in the active level.

    $\Rightarrow$ The latch is controlled by the "level" of its control input.
- Flip-flop: memory element for ***sync*** seq ckt
  - responses only to a "transition" of a triggering input called the "clock."
  - Positive-edge trigger: $0 \rightarrow 1$
  - Negative-edge trigger: $1 \rightarrow 0$

- Clock response in latch & flip-flop:

(a) Response to positive level

(b) Positive-edge response

(c) Negative-edge response

- Flip-flop: ← Latches

1. Master-slave flip-flop: pulse-triggered f-f

   Employ two latches in a special configuration that isolates the output of the flip-flop from being affected while its input is changing.

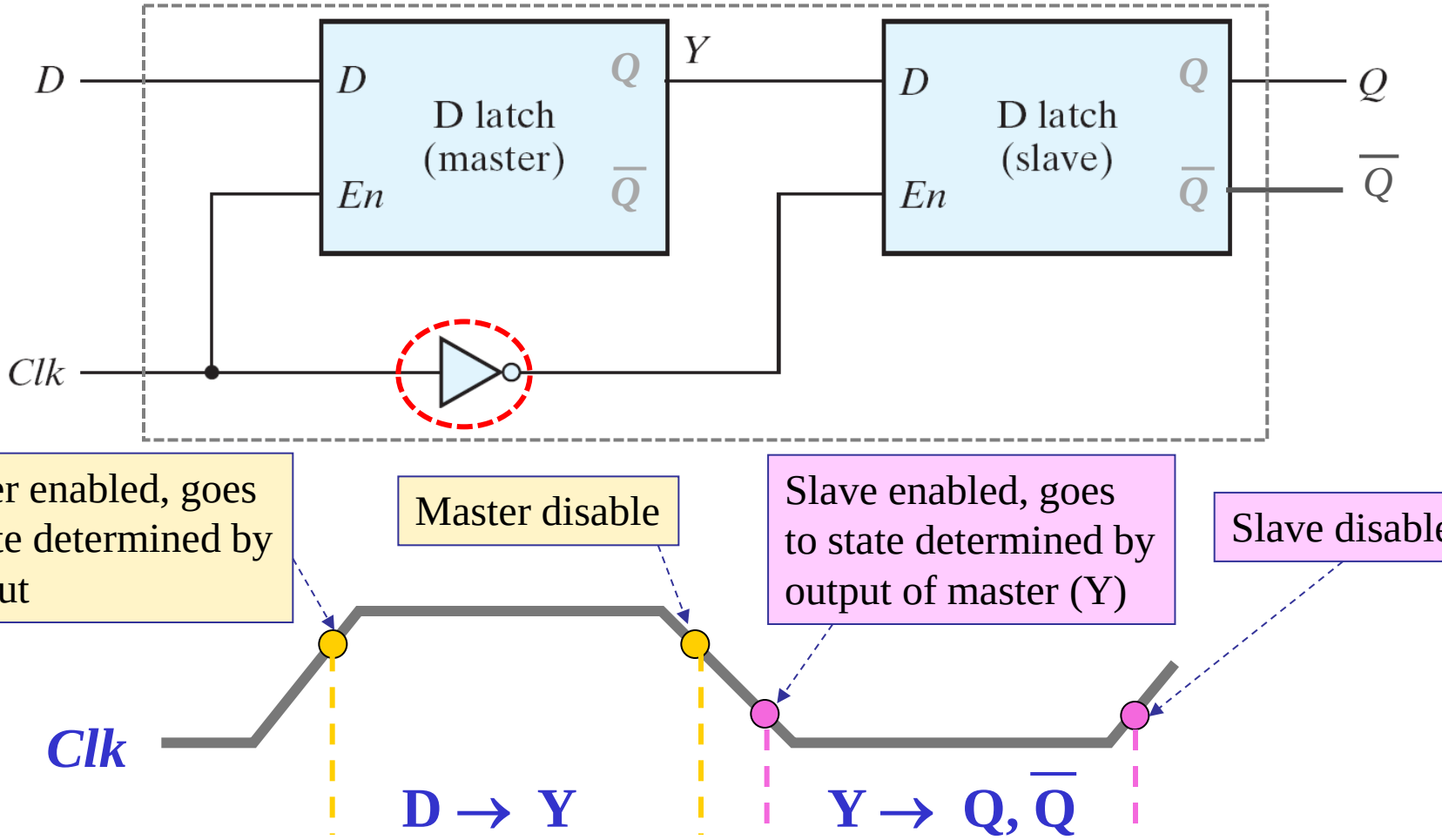
2. Edge-triggered flip-flop:

   Produce a flip-flop that triggers only during a signal transition, and is disabled during the rest of the clock pulse duration.

A. Master-Slave Flip-Flops

*** Q may change only during the *negative* edge of the clock.**

- Master-slave D flip-flop:

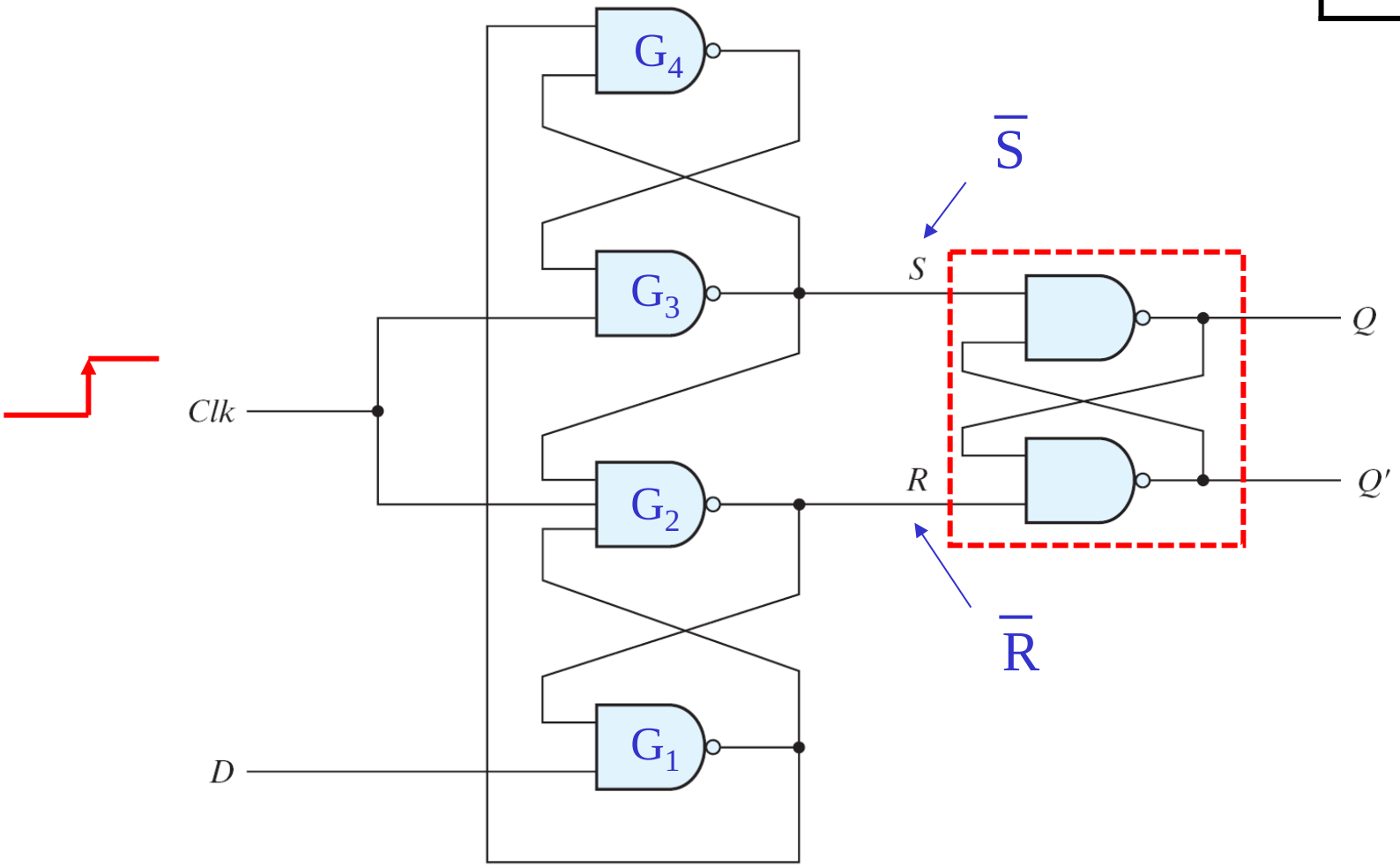


B. Edge-Triggered D Flip-Flop

$Q^+ = D$

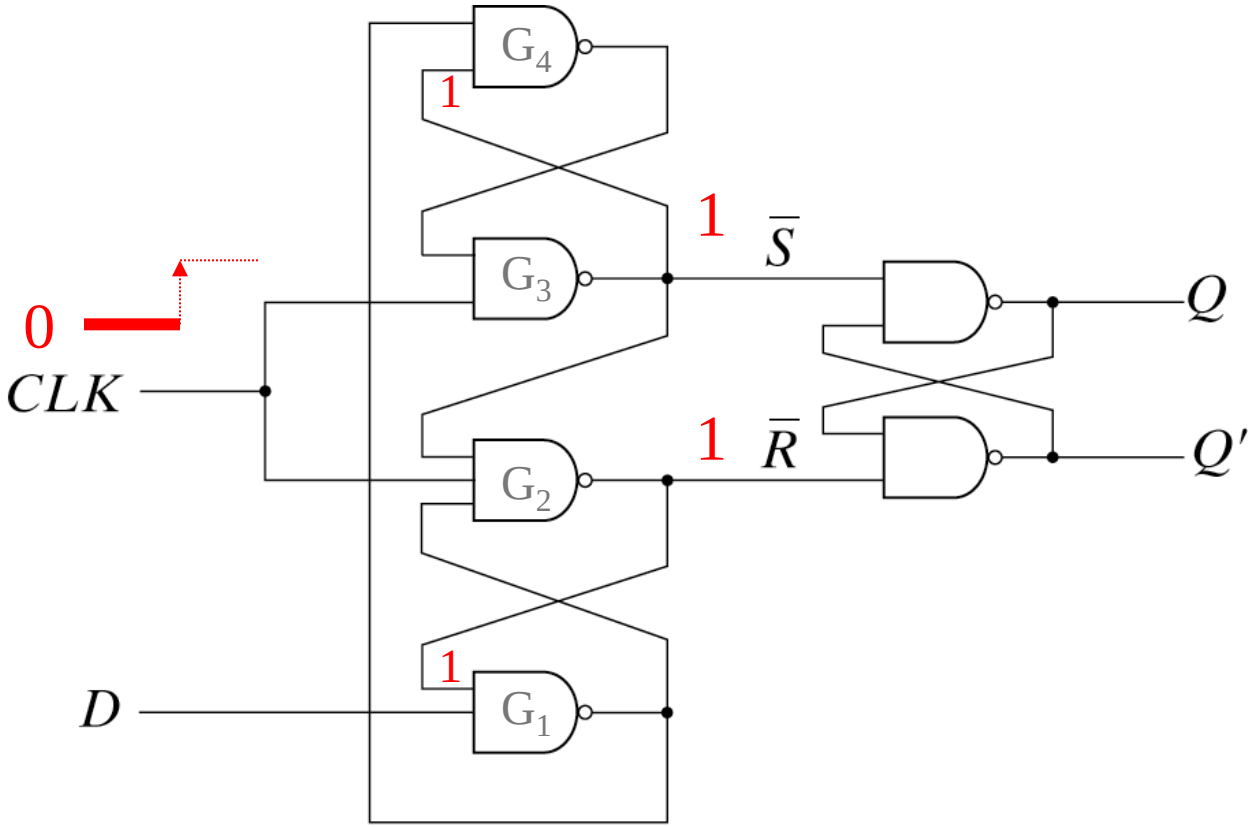
D	Q+
0	0
1	1

- D-Type Positive-Edge-Triggered Flip-Flop



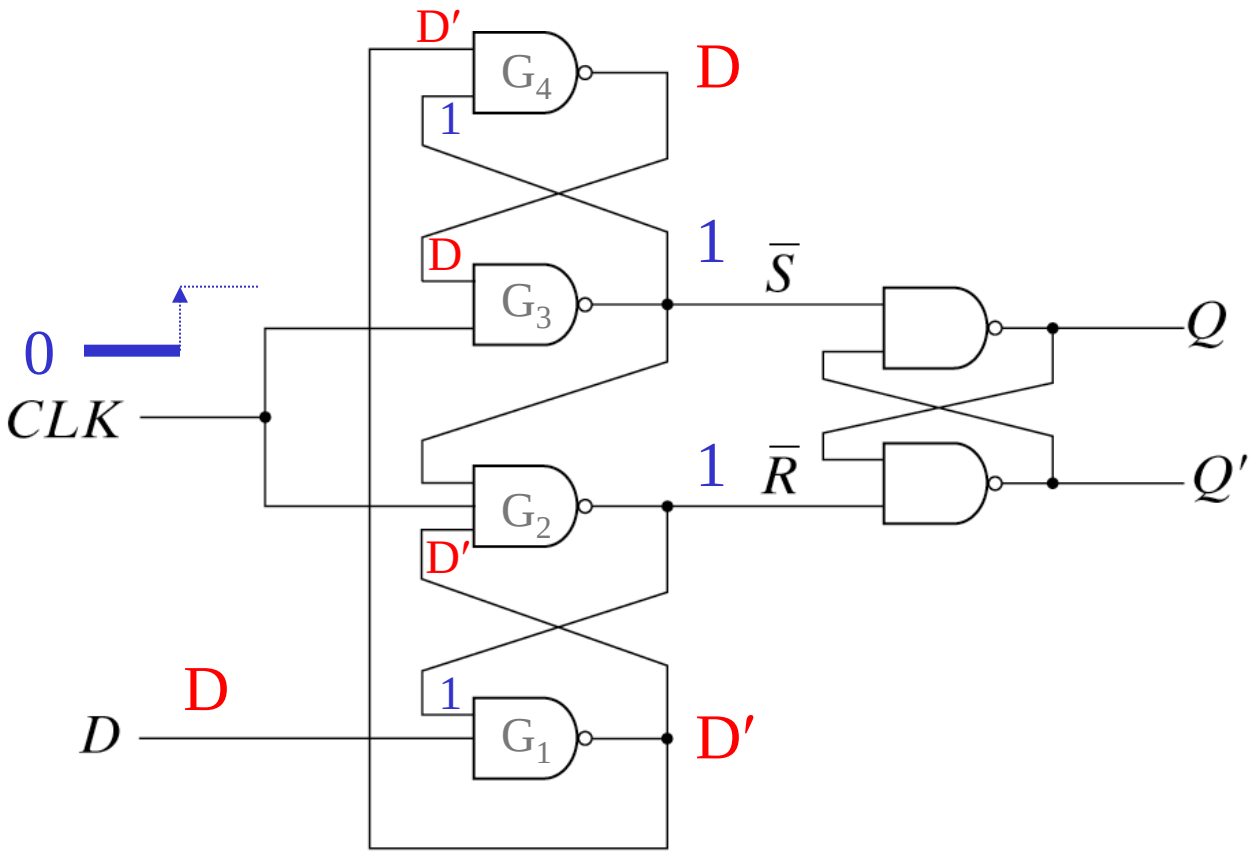
$\overline{S}$	$\overline{R}$	Q^+
1	1	No change ($Q^+ = Q$)
1	0	Reset ($Q^+ = 0$)
0	1	Set ($Q^+ = 1$)
0	0	Indeterminate

- When clock is **LOW**:



$\overline{S}$	$\overline{R}$	Q^+
1	1	No change ($Q^+ = Q$)
1	0	Reset ($Q^+ = 0$)
0	1	Set ($Q^+ = 1$)
0	0	Indeterminate

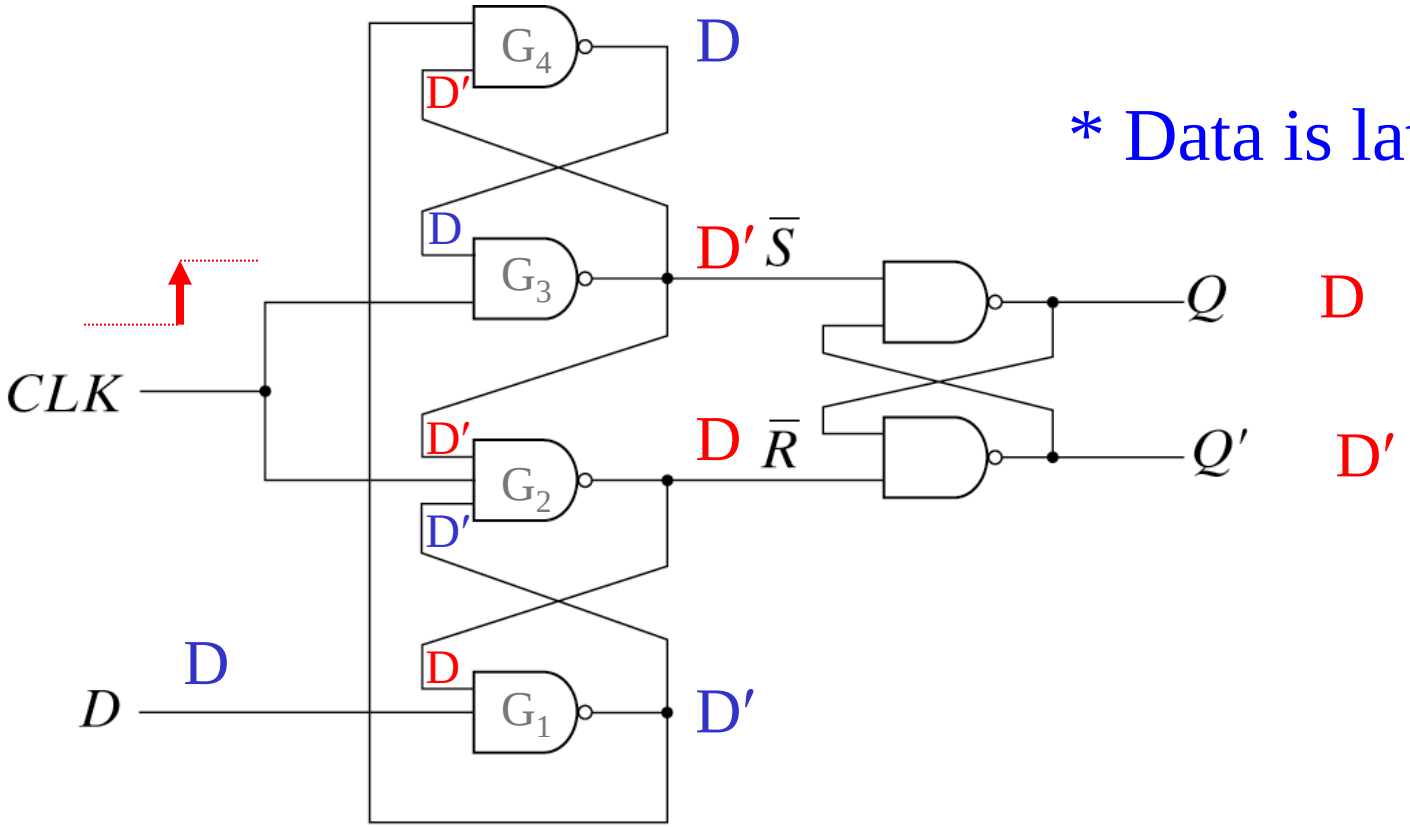
- When clock is **LOW**:



No change!

$\overline{S}$	$\overline{R}$	Q^+
1	1	No change ($Q^+ = Q$)
1	0	Reset ($Q^+ = 0$)
0	1	Set ($Q^+ = 1$)
0	0	Indeterminate

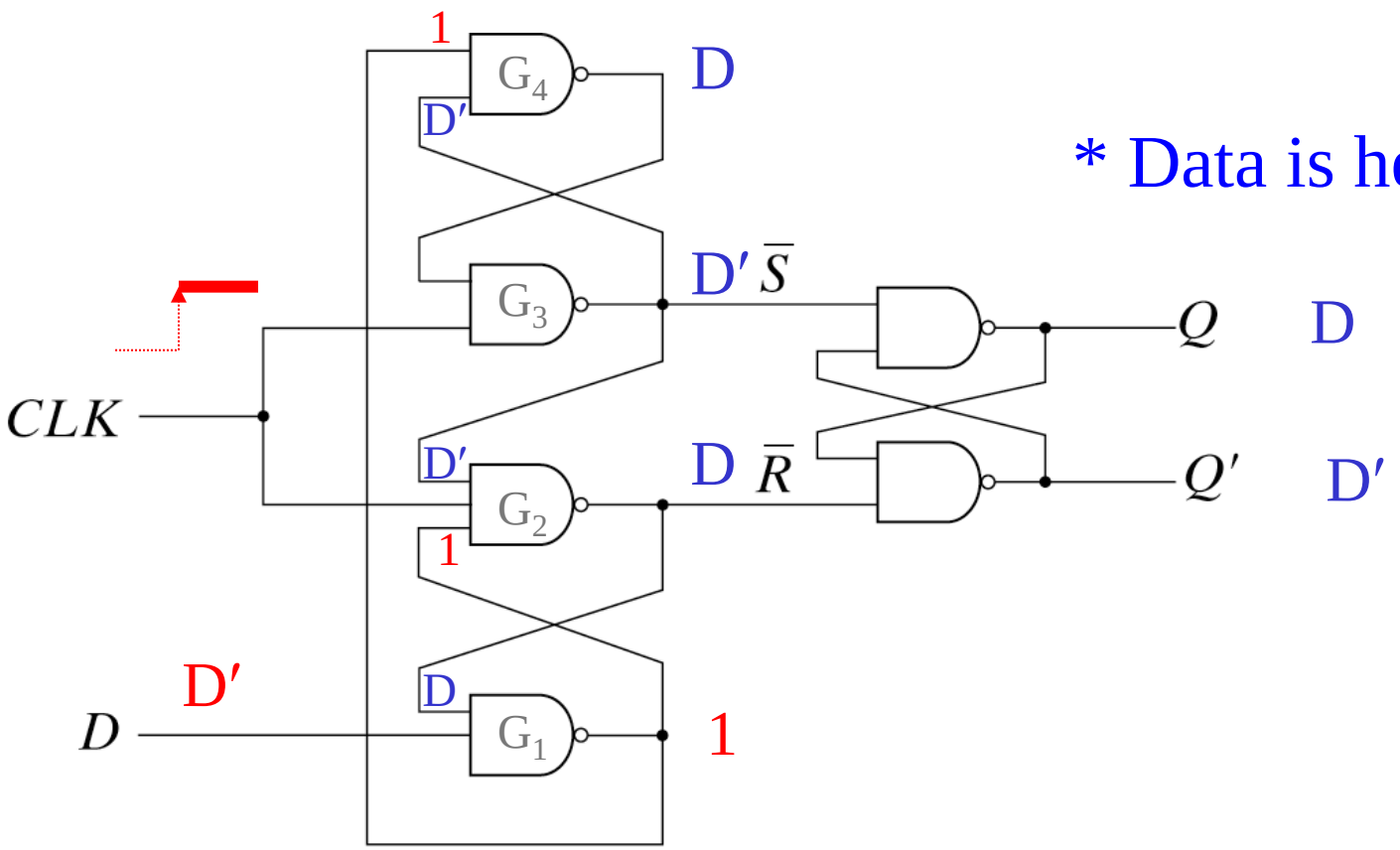
- When clock goes **LOW-to-HIGH**:



* Data is latched!

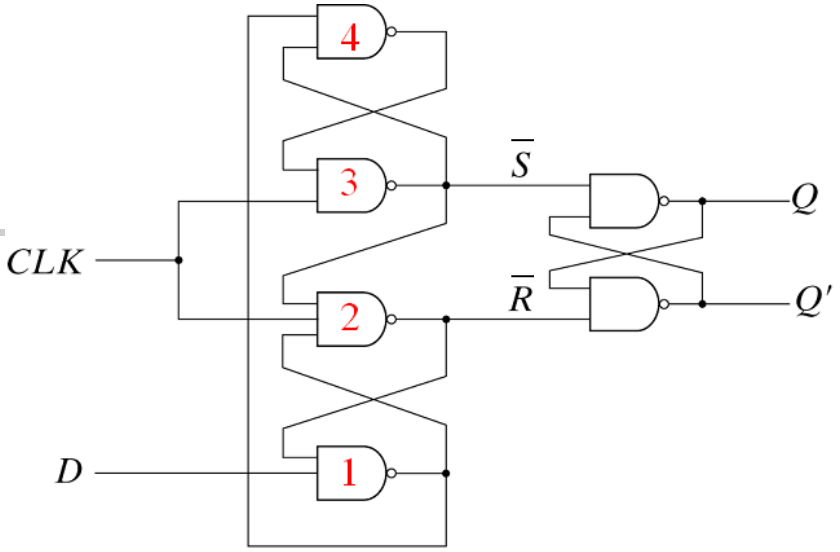
$\overline{S}$	$\overline{R}$	Q^+
1	1	No change ($Q^+ = Q$)
1	0	Reset ($Q^+ = 0$)
0	1	Set ($Q^+ = 1$)
0	0	Indeterminate

- When clock is **HIGH**:



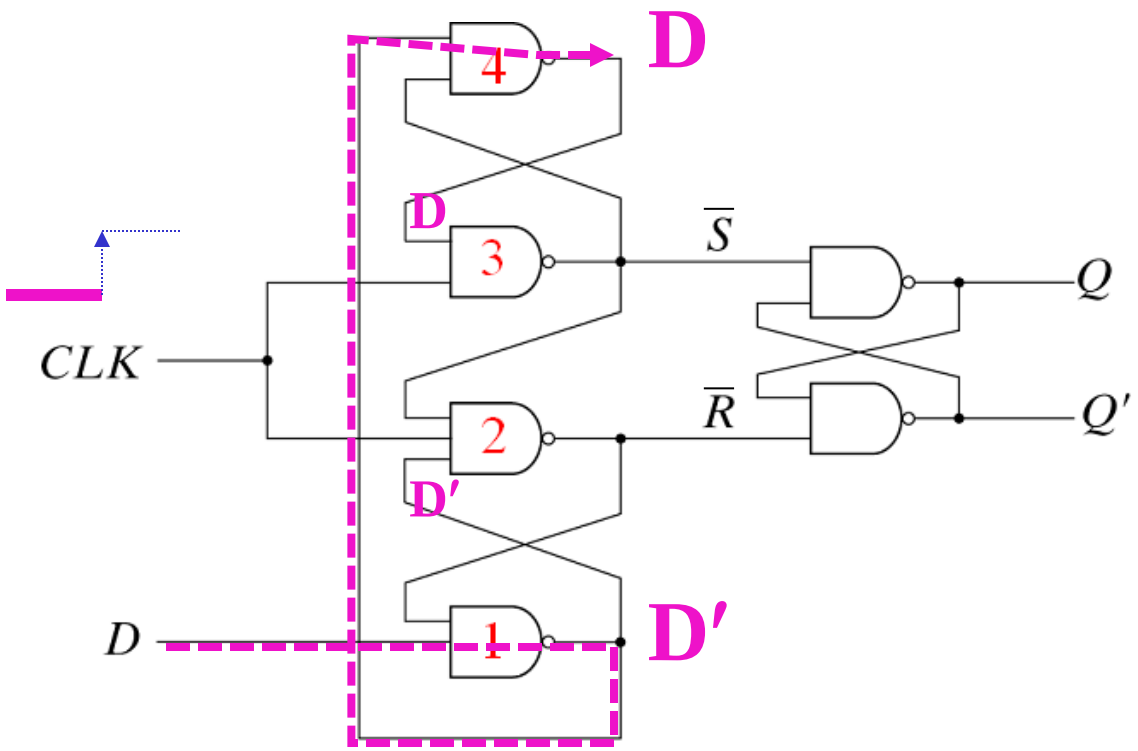
* Data is held!

Timing

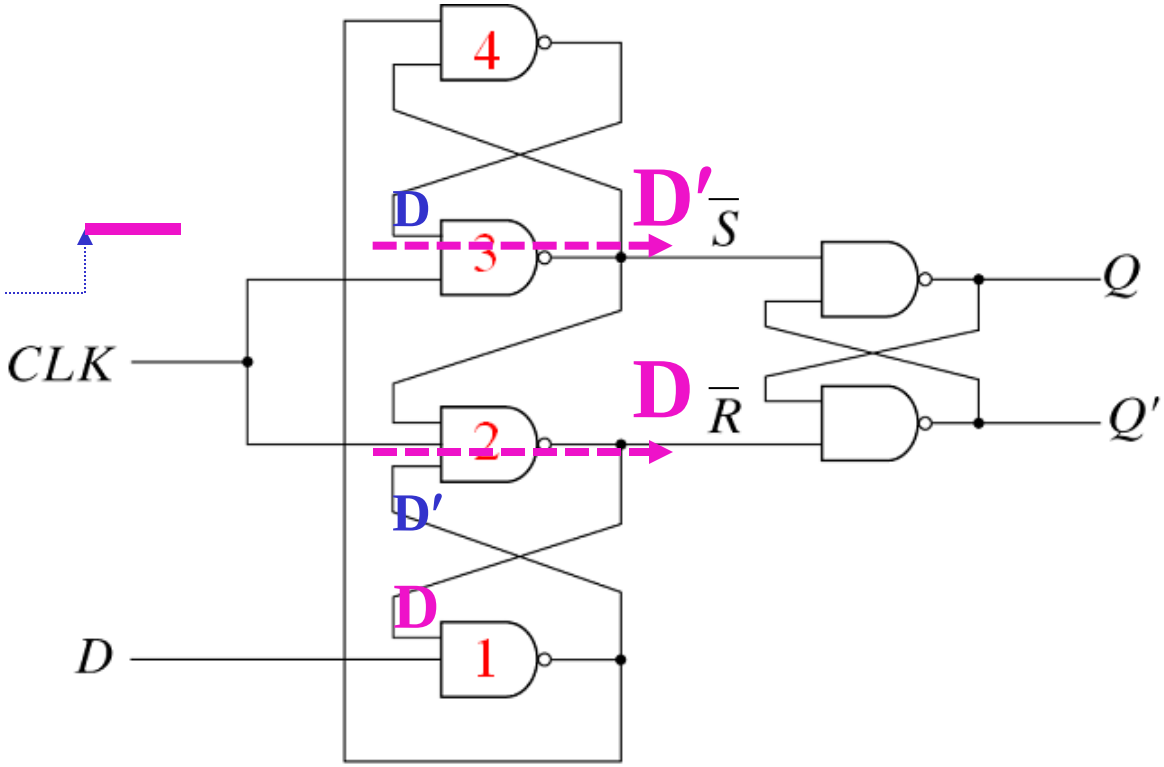


- Setup time: G1 + G4
  - the minimum time for which the D input must be maintained at a constant value prior to the occurrence of the clock transition
- Hold time: max(G2, G3)
  - the minimum time for which the D input must not change after the application of the positive transition of the clock
- Propagation delay time:
  - the time interval b/t the trigger edge and the stabilization of the output to a new state

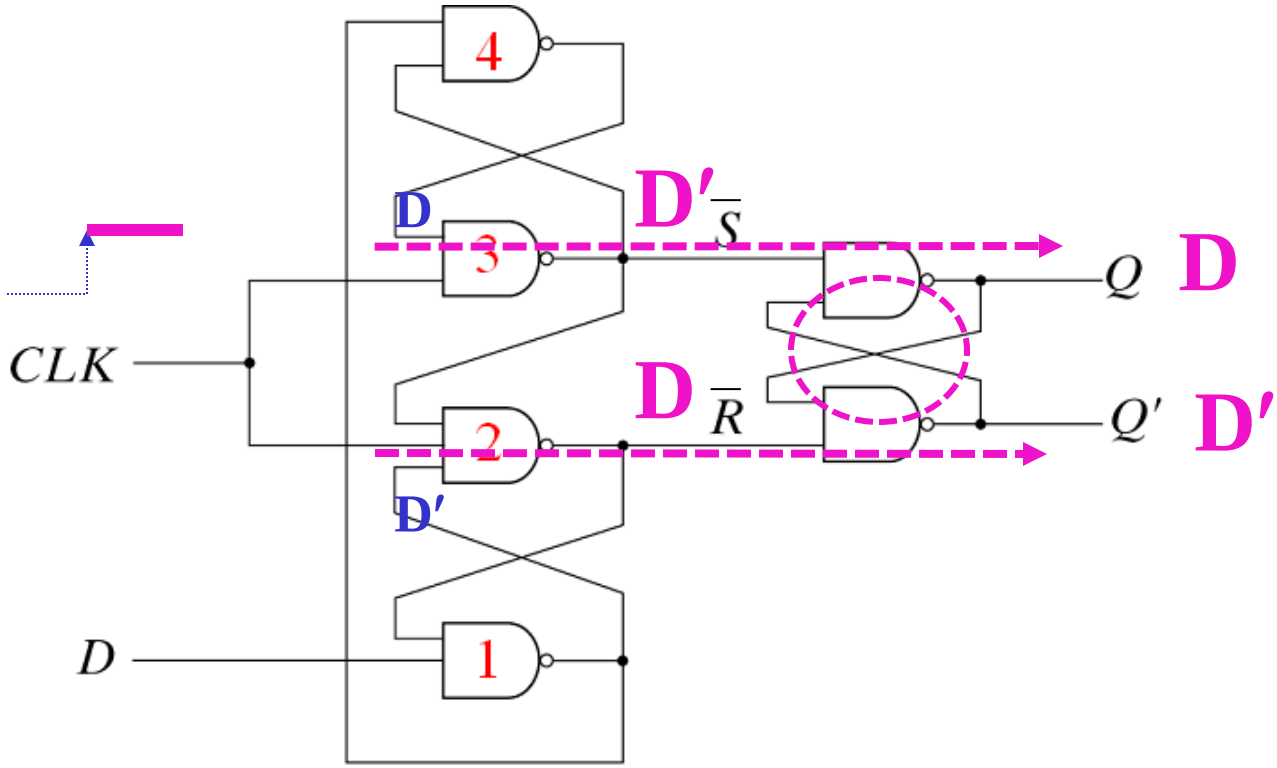
- Setup time: G1 + G4
  - the minimum time for which the D input must be maintained at a constant value prior to the occurrence of the clock transition



- Hold time: max(G2, G3)
  - the minimum time for which the D input must not change after the application of the positive transition of the clock

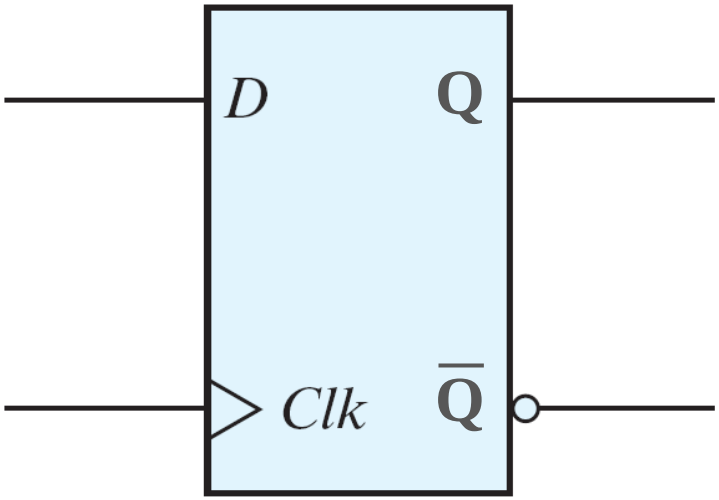


- Propagation delay time:
  - the time interval b/t the trigger edge and the stabilization of the output to a new state

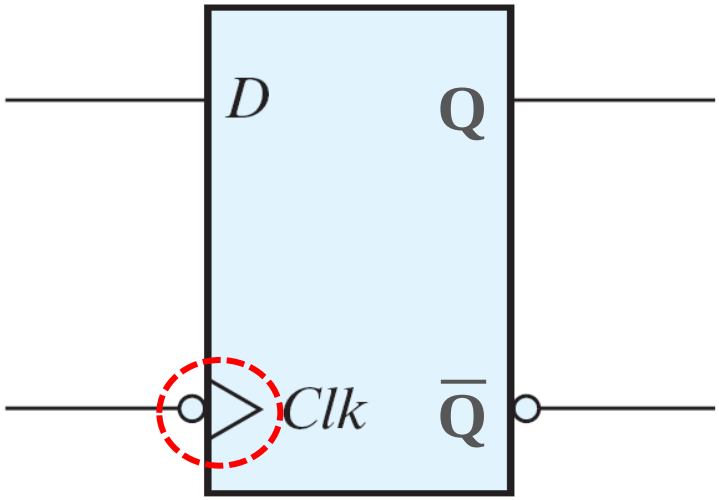


Graphic symbol

- Graphic symbol for the edge-triggered D flip-flop:



(a) Positive-edge



(a) Negative-edge

C. Direct Inputs

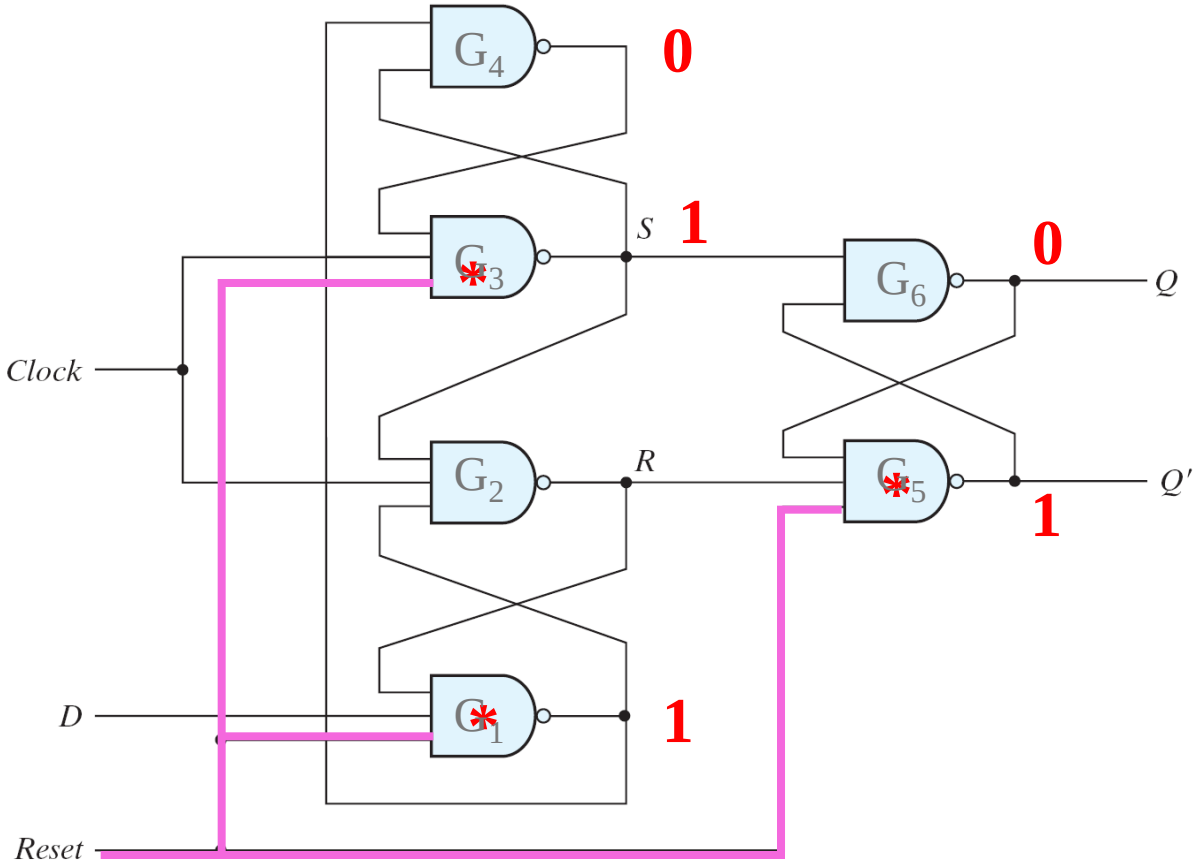
- ***Direct*** inputs: ***asynchronous*** inputs
  - are used to force the flip-flop to a particular state independent of the clock.
    - When power is turned on in a digital system, the state of the flip-flops is unknown.
    - The direct inputs are useful for bringing all flip-flops in the system to a known starting state prior to the clocked operation.
  - Types of direct inputs:
    - ***Preset***: direct set, sets the f-f to 1 (Q = 1, $\overline{Q}$ = 0)
    - ***Clear***: direct reset, sets the f-f to 0 (Q = 0, $\overline{Q}$ = 1)

Example

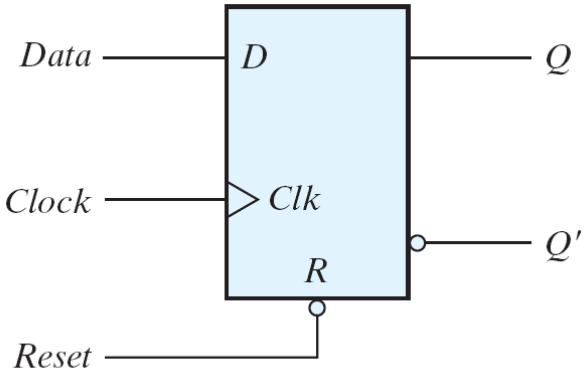
- E.g.:

  D flip-flop w/ *asynchronous reset*

  * Active LOW



(a) Circuit diagram



(b) Graphic symbol

R	Clk	D	Q	Q′
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

(b) Function table

D. Other Flip-Flops

- JK flip-flop
- T flip-flop

J-K Flip-Flop

- SR flip-flop

S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Indeterminate

- JK flip-flop

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Q′

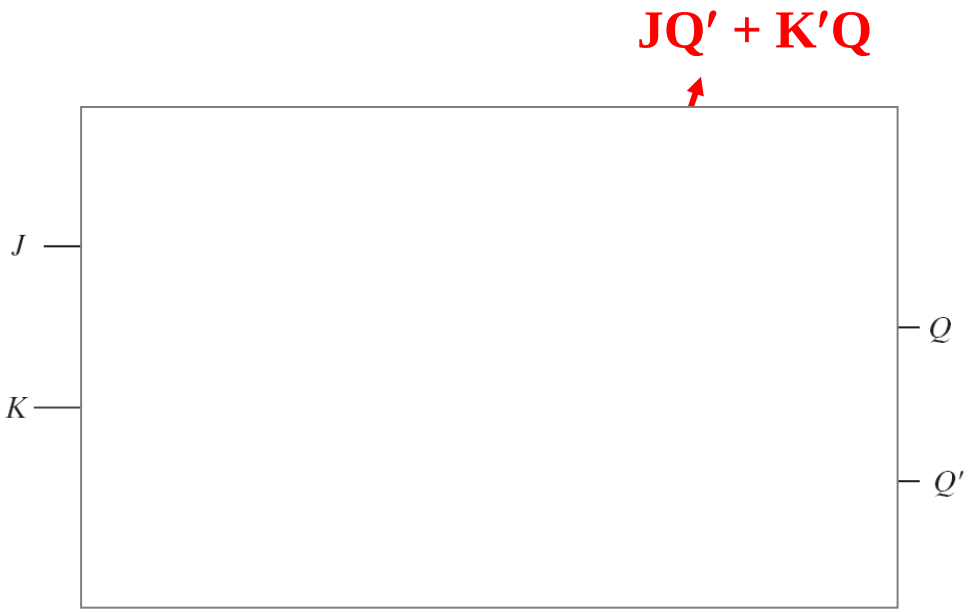
$$Q^+ = J'K'Q + JK' + JKQ'$$
$$= JQ' + K'Q$$

- JK flip-flop

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Q′

$Q^+ = JQ' + K'Q$

Circuit diagram

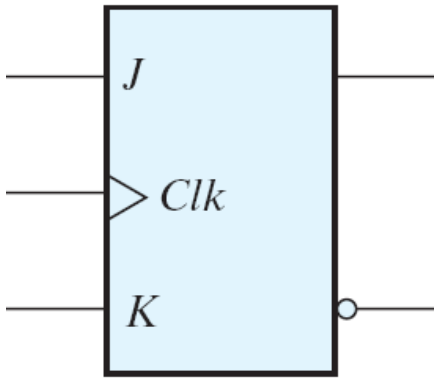


$Q^+ = D = JQ' + K'Q$

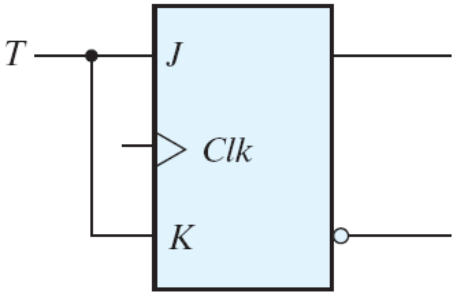
Verification

J	K	D	Q^+
0	0	Q	Q
0	1	0	0
1	0	1	1
1	1	Q′	Q′

Graphic symbol



T Flip-Flop



(a) From *JK* flip-flop

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Q′

$Q^+ = JQ' + K'Q$

J = K

$\Rightarrow$

T	Q^+
0	Q
1	Q′

$Q^+ = T'Q + TQ'$

From D flip-flop:

$Q^+ = D = TQ' + T'Q$

$= T \oplus Q$

E. Characteristic Tables & Equations

JK Flip-Flop

J	K	Q(t + 1)	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

$Q^+ = JQ' + K'Q$

D Flip-Flop

D	Q(t + 1)	
0	0	Reset
1	1	Set

$Q^+ = D$

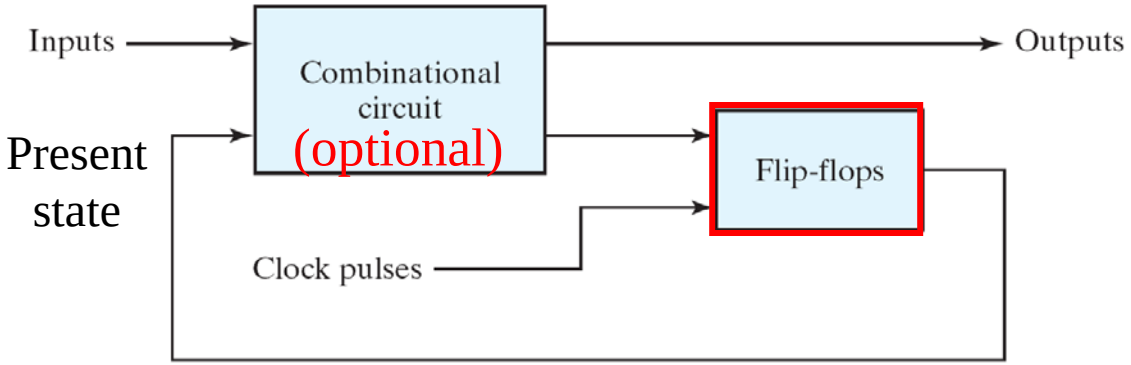
T Flip-Flop

T	Q(t + 1)	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

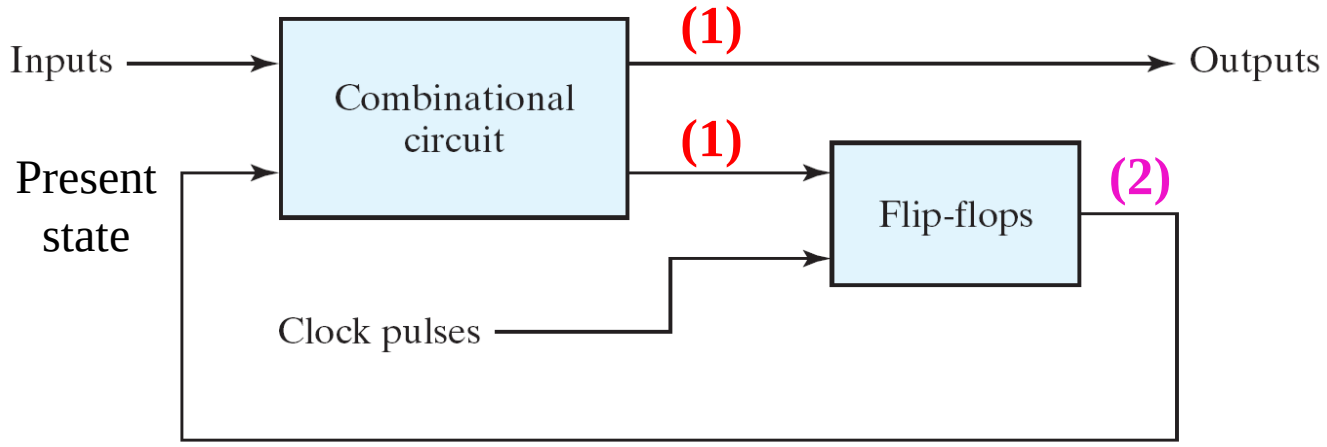
$Q^+ = TQ' + T'Q$

5-5 Analysis of Clocked Sequential Ckts

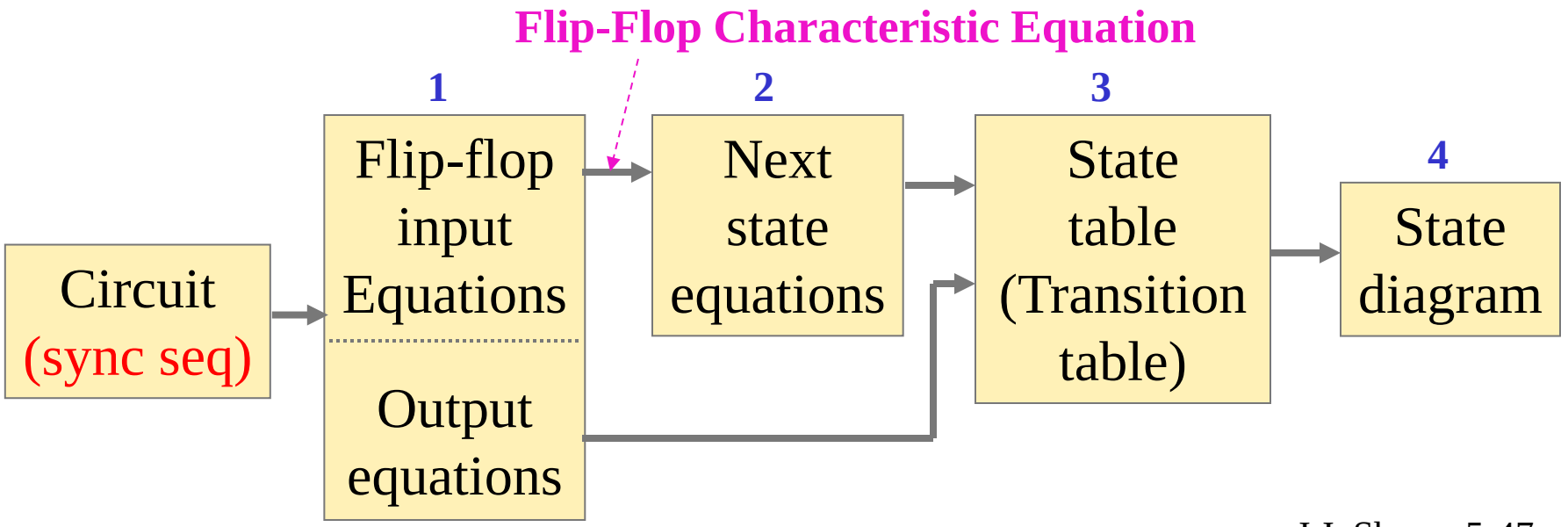
- Synchronous seq ckt:



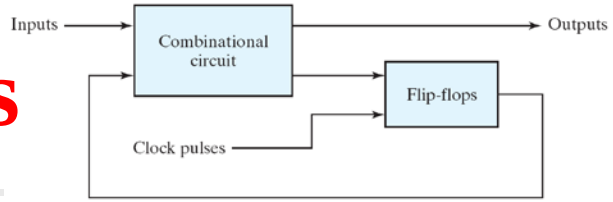
  - includes f-fs w/ the clock inputs driven directly or indirectly by a clock signal and the direct sets and resets are unused during the normal functioning of the ckt
  - The behavior of a seq ckt is determined from the inputs, outputs, and present state of the ckt.
- Analysis of a seq ckt:
  - obtain a suitable description that demonstrates the time sequence of inputs, outputs, and states



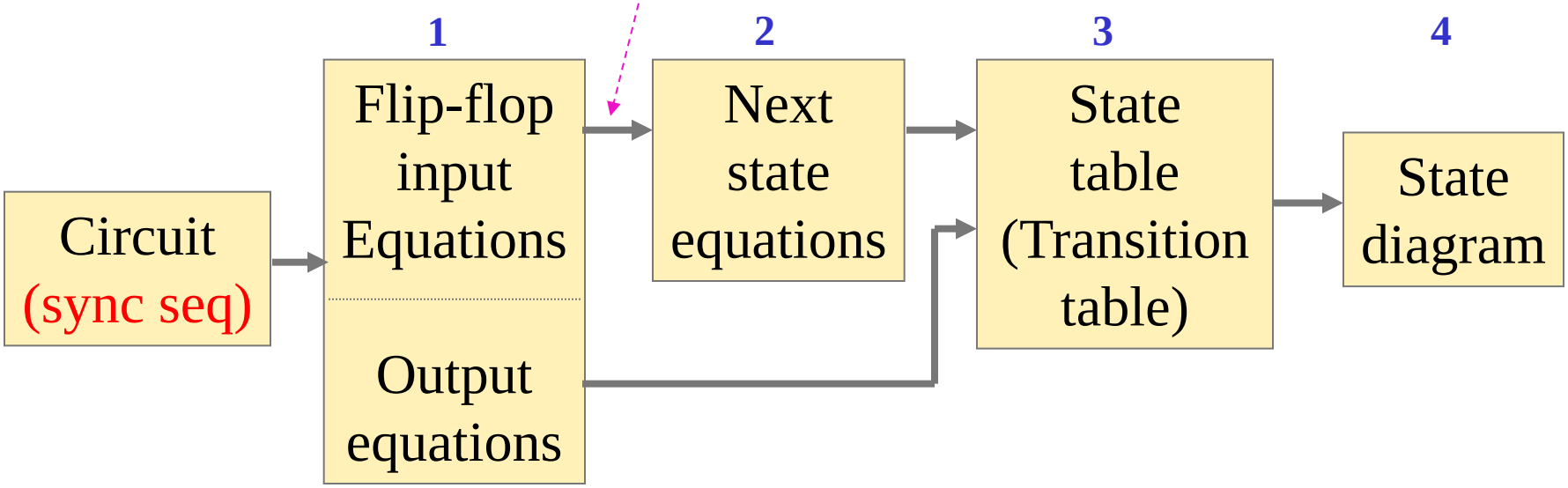
- Analysis procedure:



A. Analysis w/ *D* Flip-Flops



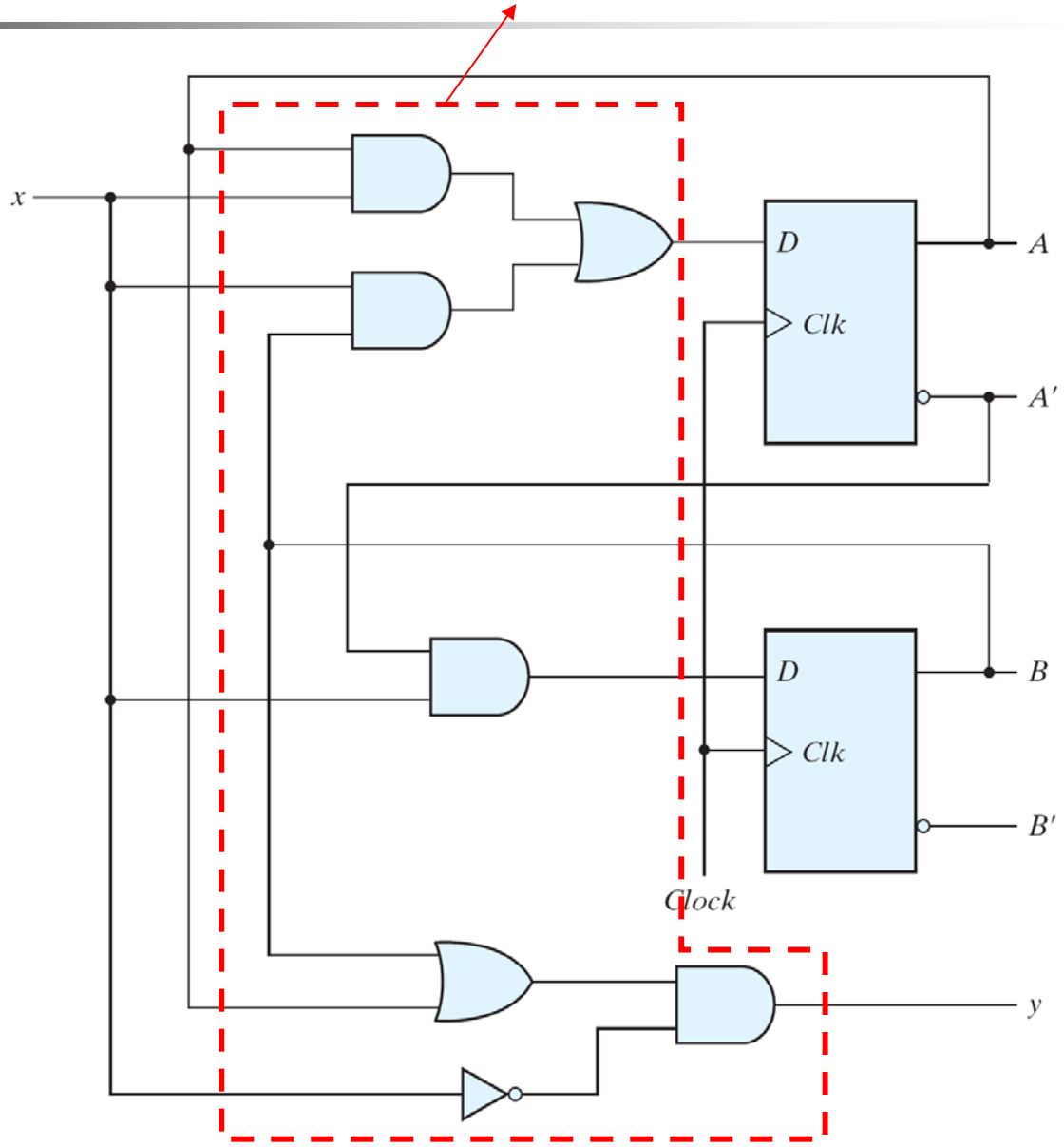
Flip-Flop Characteristic Equation

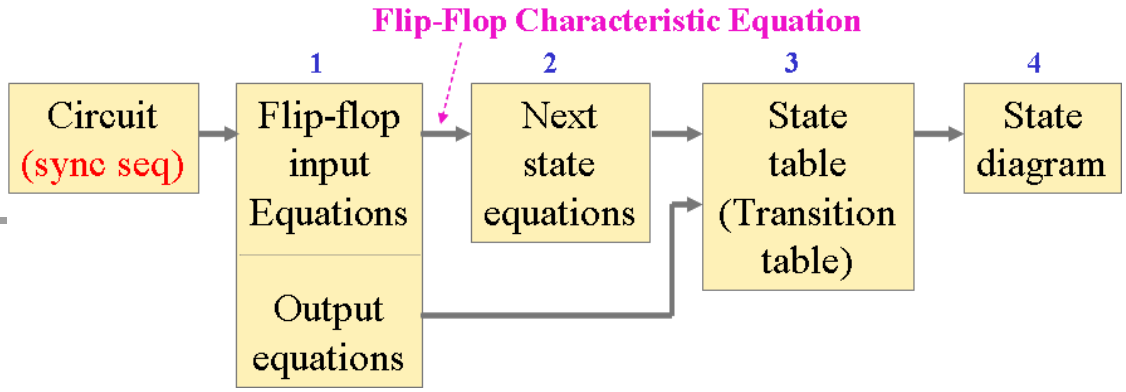


For D flip-flops: $Q^+ = D$

Example 1

- E.g. 1:





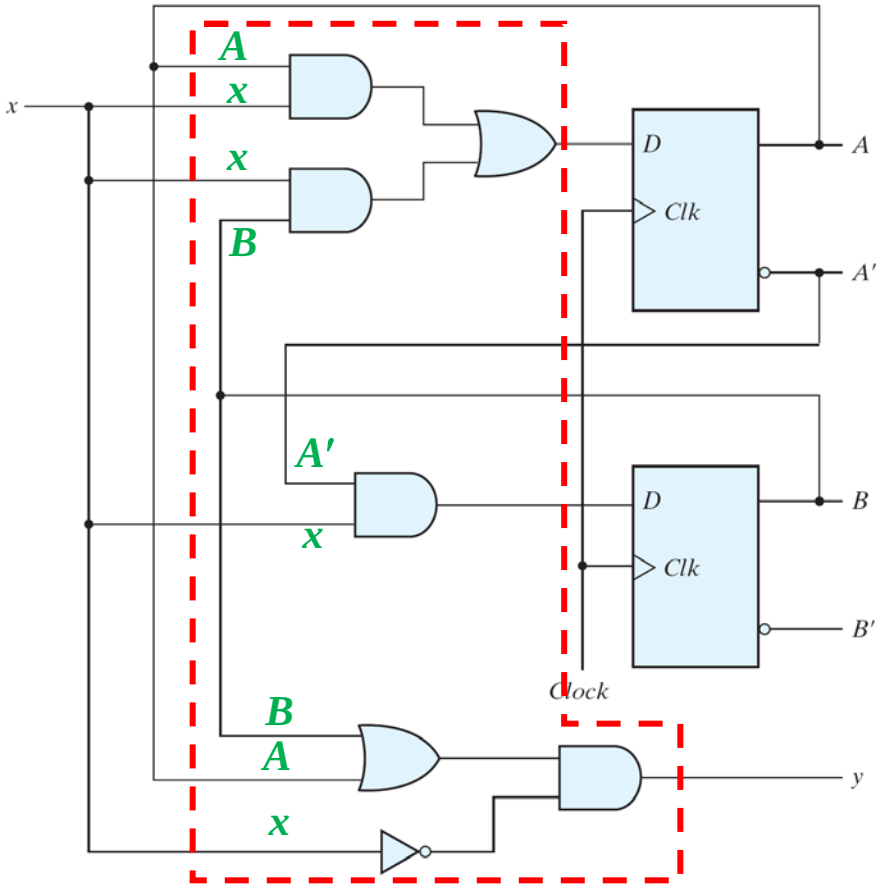
<Analysis>

1. Flip-flop input equations & Output equations:

$$D_A = AX + BX$$

$$D_B = \overline{A}X$$

$$Y = (A + B)\overline{X}$$

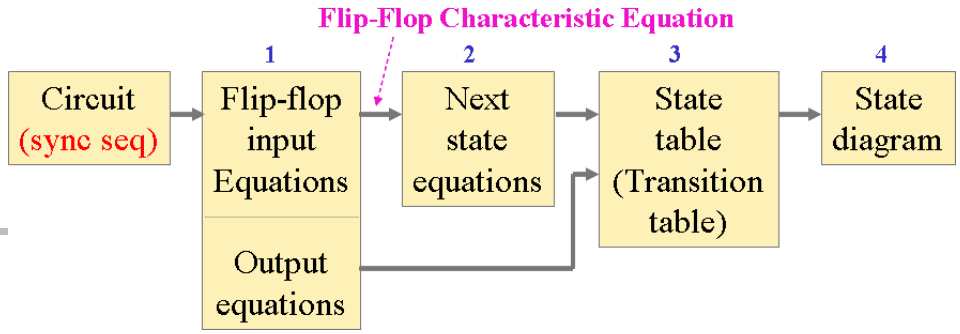


2. Next state equation:

($Q^+ = D$)

$$A(t+1) = D_A = AX + BX$$

$$B(t+1) = D_B = \overline{A}X$$



3. State table:

Next state equations:

$$A(t+1) = AX + BX$$

$$B(t+1) = \overline{A}X$$

Output equations:

$$Y = A\overline{X} + B\overline{X}$$

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			

(One-dimensional)
State table

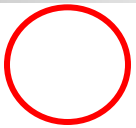
Present State		Input	Next State		Output
A	B	x	A^+	B^+	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

⇓

Two-dimensional
state table

Present State		Next State				Output	
		$x = 0$		$x = 1$		$x = 0$	$x = 1$
A	B	A^+	B^+	A^+	B^+	y	y
0	0	0	0	0	1	0	0
0	1						
1	0						
1	1						

4. State diagram:

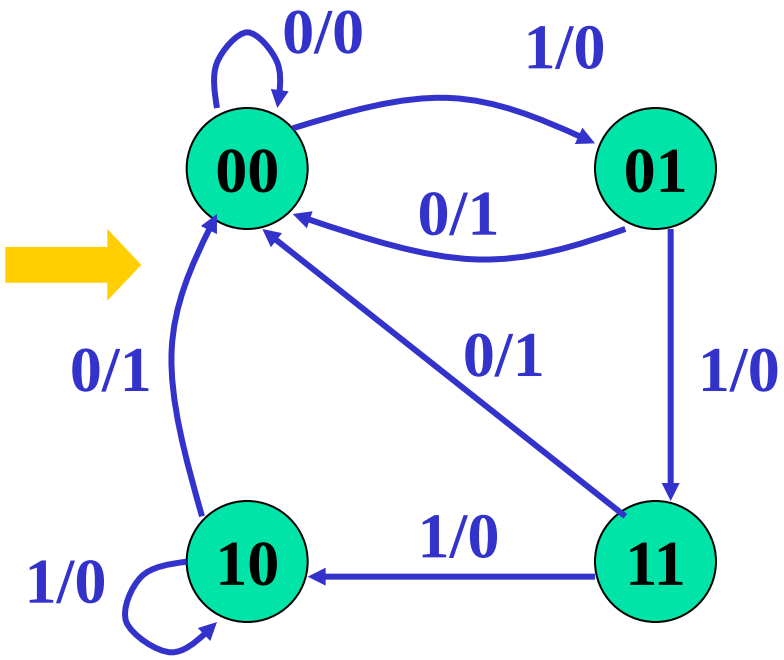


input/output

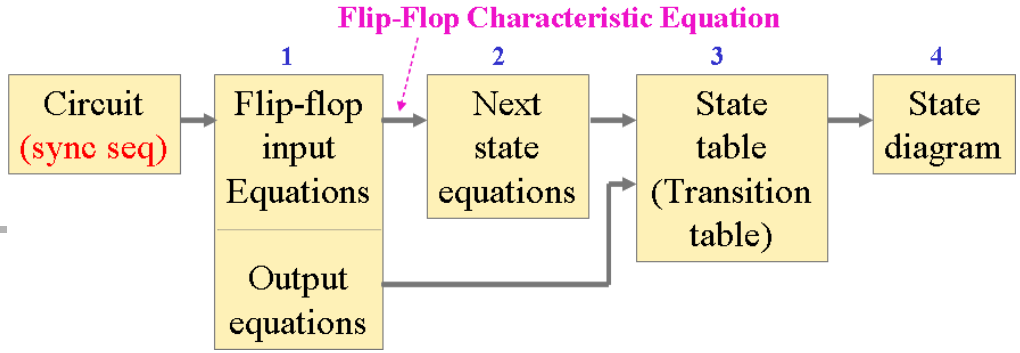
X/Y

Present State		Next State				Output	
		$x = 0$		$x = 1$		$x = 0$	$x = 1$
A	B	A^+	B^+	A^+	B^+	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

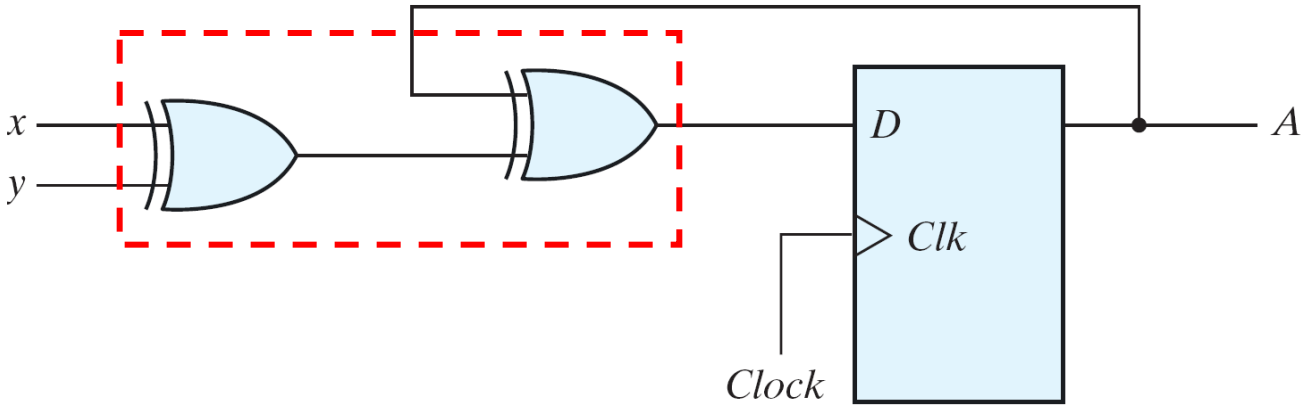
State table



Example 2



- E.g. 2:



1. Flip-Flop input equations & Output equations:

$D_A = A \oplus X \oplus Y$

$Z = A$

2. Next-state equation: **$Q^+ = D$**

$A^+ = D_A = A \oplus X \oplus Y$

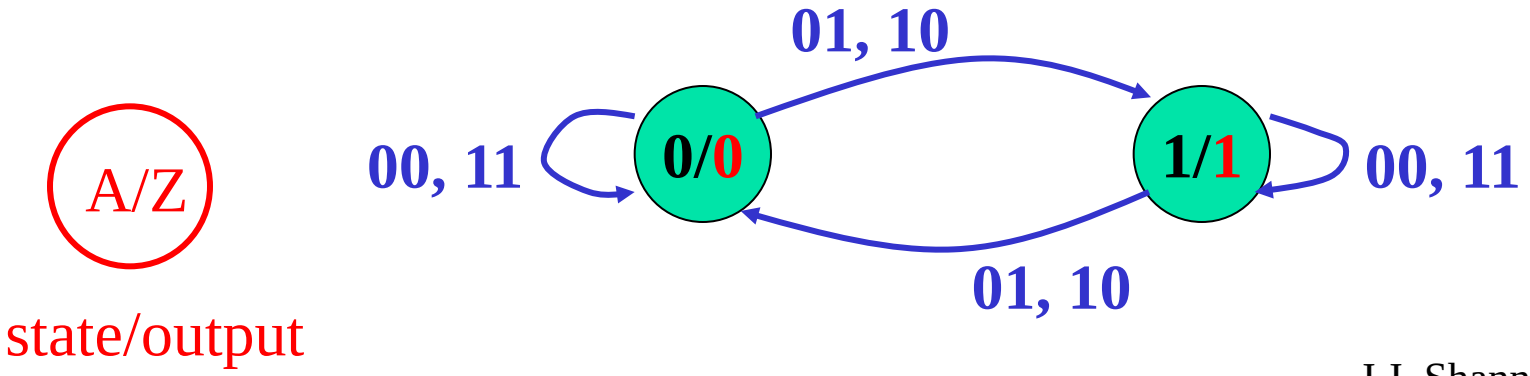
3. State table:

$$A^{+} = A \oplus X \oplus Y$$

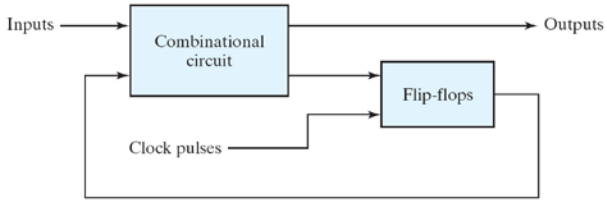
$$Z = A$$

Present state	Inputs		Next state	Output
A	X	Y	A^{+}	Z
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

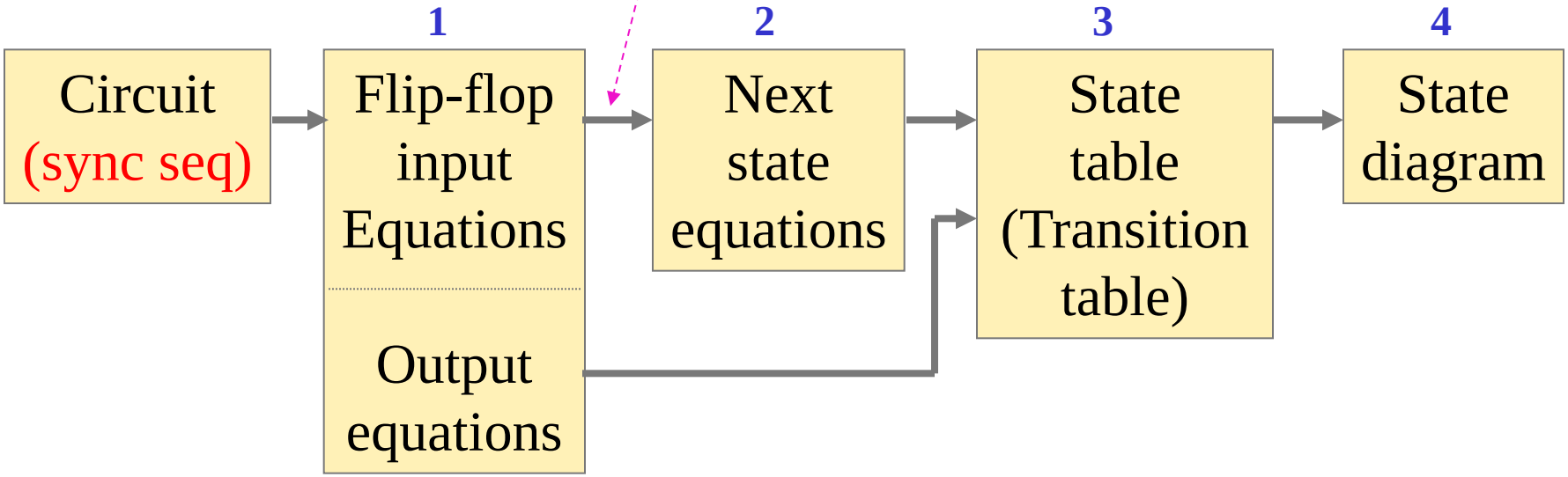
4. State diagram:



B. Analysis w/ *JK* Flip-Flop



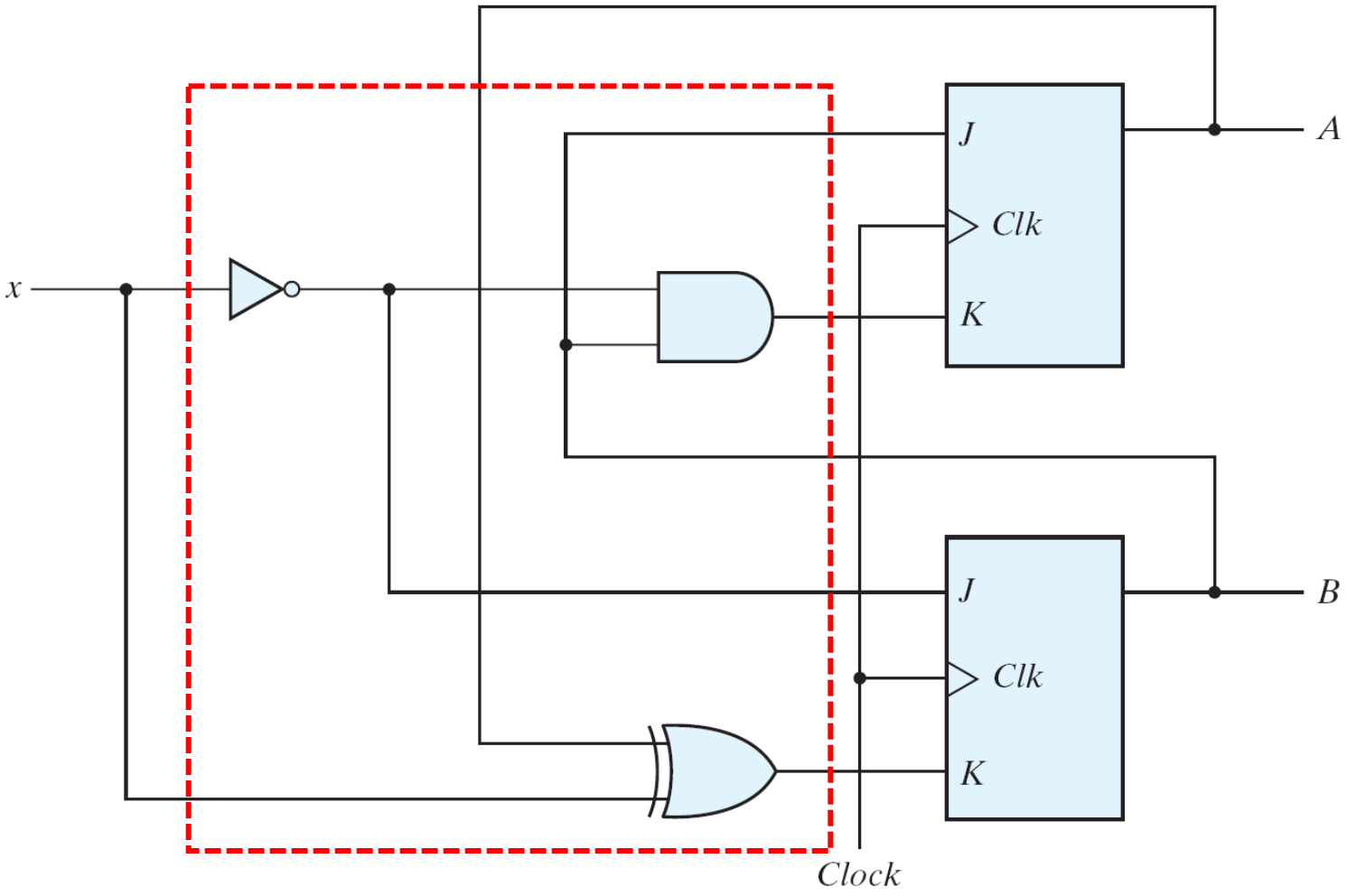
Flip-Flop Characteristic Equation

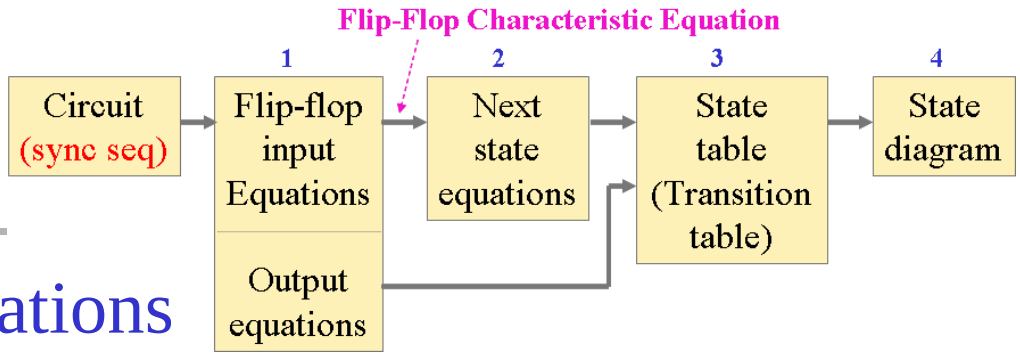


For JK flip-flops: $Q^+ = JQ' + K'Q$

Example

- Example:





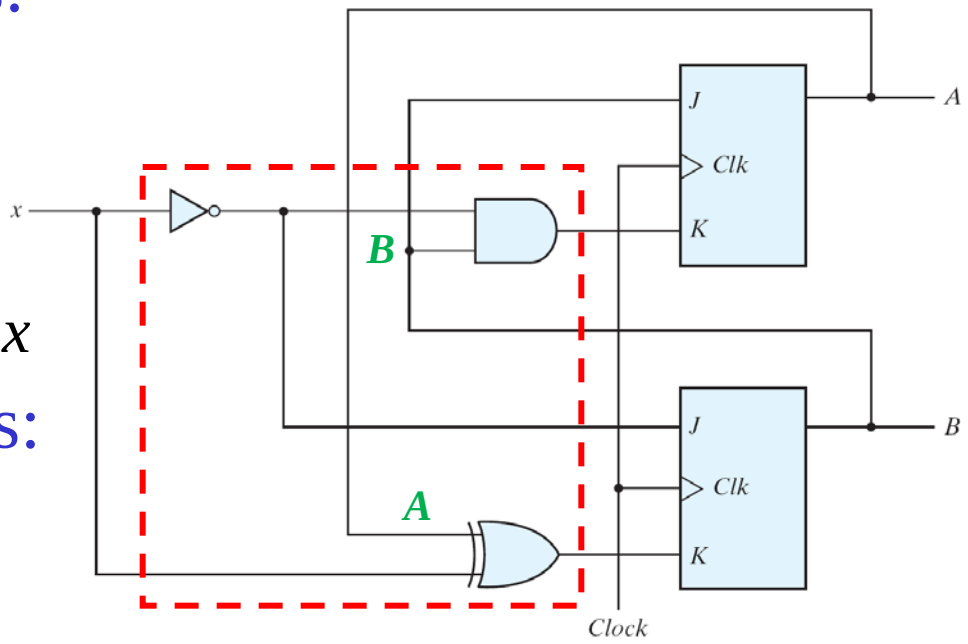
1. Flip-flop input equations & Output equations:

$J_A = B$

$K_A = B\,x'$

$J_B = x'$

$K_B = A \oplus x = Ax' + A'x$



2. Next state equations:

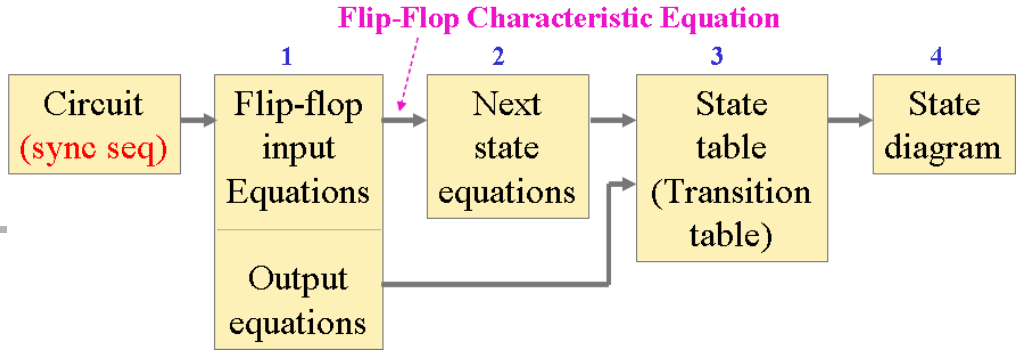
$(Q^+ = J\,Q' + K'\,Q)$

$A^+ = J_A\,A' + K_A'\,A$

$= (B)\,A' + (Bx')'\,A = A'B + AB' + Ax$

$B^+ = J_B\,B' + K_B'\,B$

$= (x')\,B' + (Ax' + A'x)'\,B = B'x' + ABx + A'Bx'$



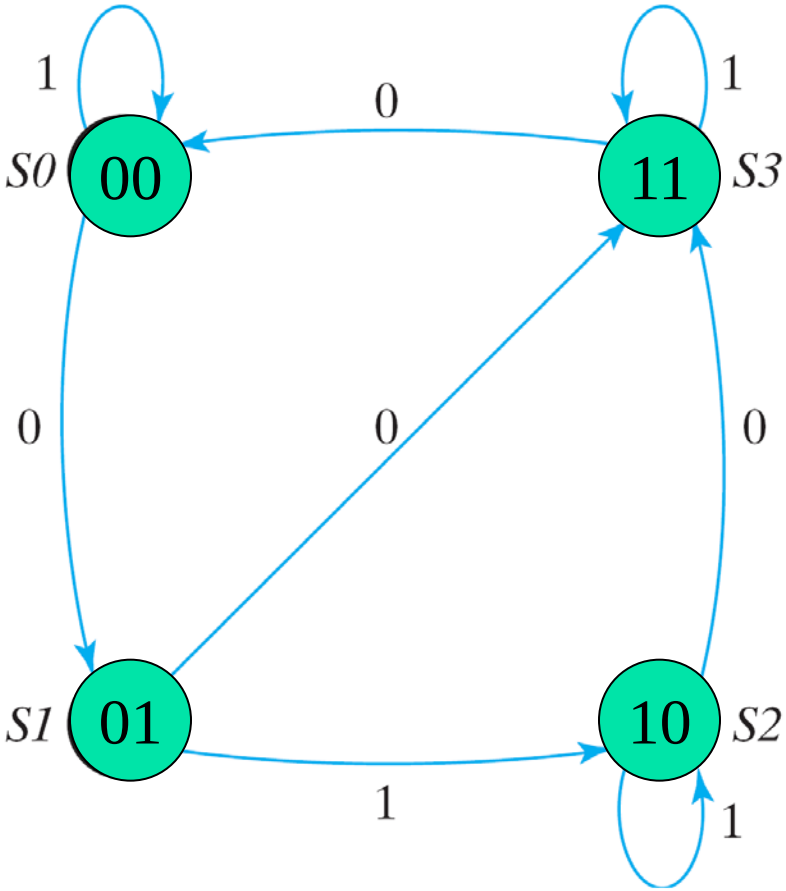
3. State table:

$$A^+ = A'B + AB' + Ax$$

$$B^+ = B'x' + ABx + A'Bx'$$

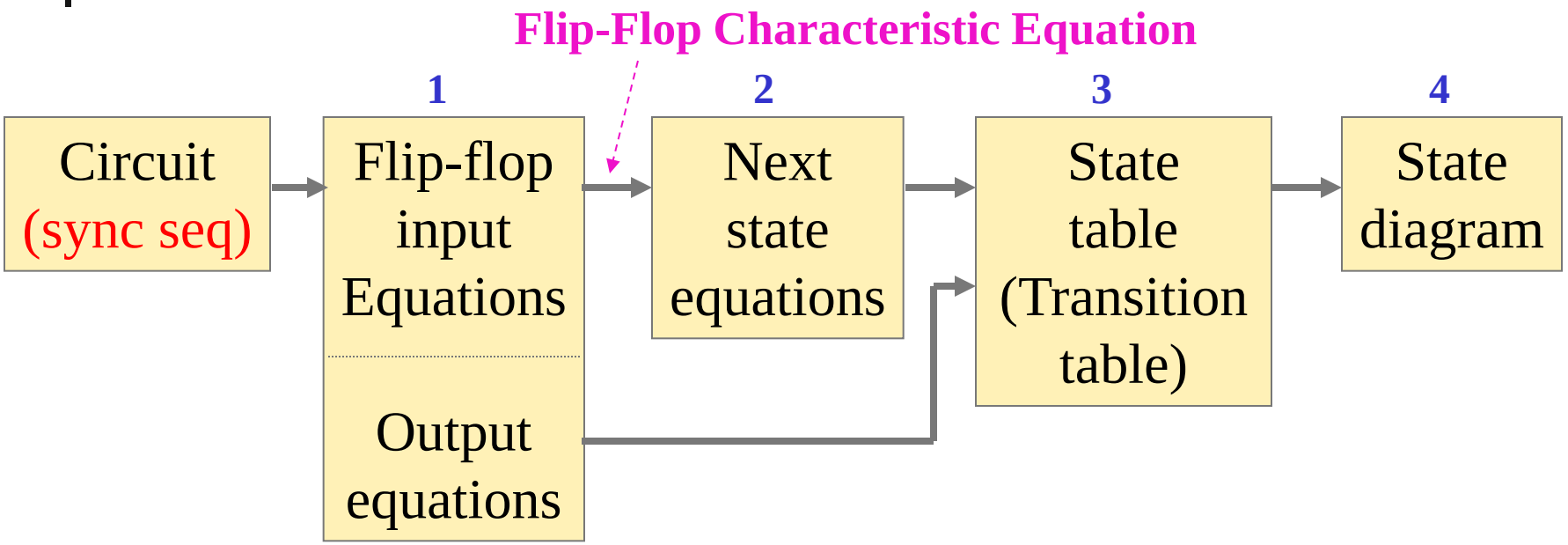
Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A^+	B^+	J_A	K_A	J_B	K_B
0	0	0			0	0	1	0
0	0	1			0	0	0	1
0	1	0			1	1	1	0
0	1	1			1	0	0	1
1	0	0			0	0	1	1
1	0	1			0	0	0	0
1	1	0			1	1	1	1
1	1	1			1	0	0	0

4. State diagram



Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A^+	B^+	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

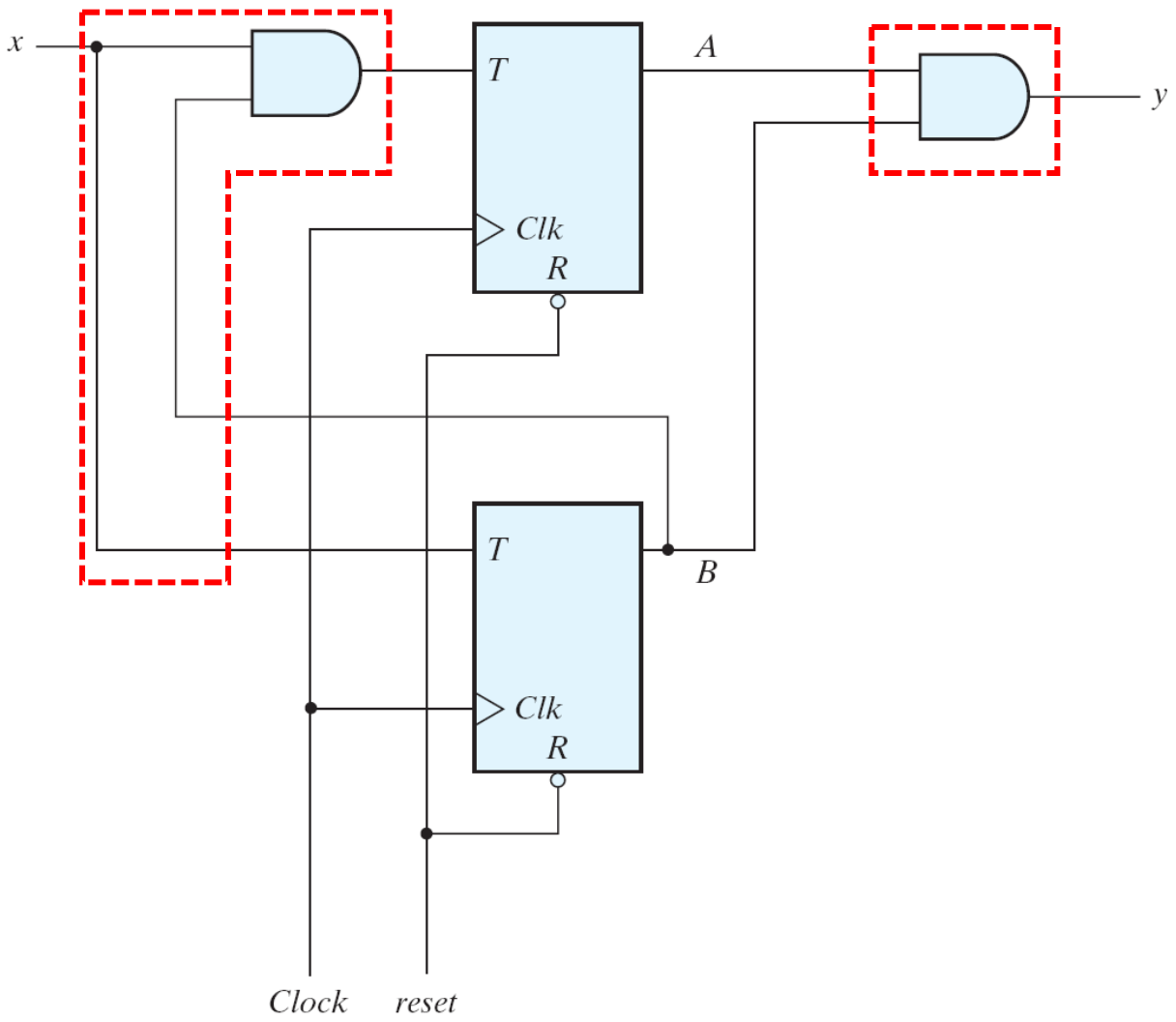
C. Analysis w/ T Flip-Flops

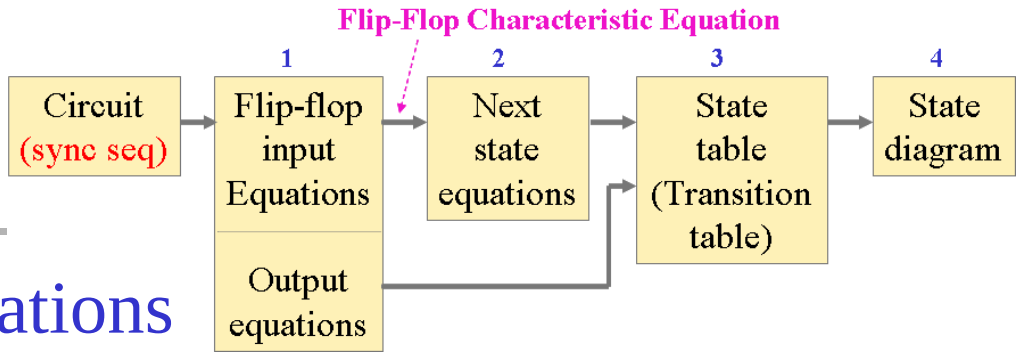


For T flip-flops: $Q^+ = T \oplus Q$

Example

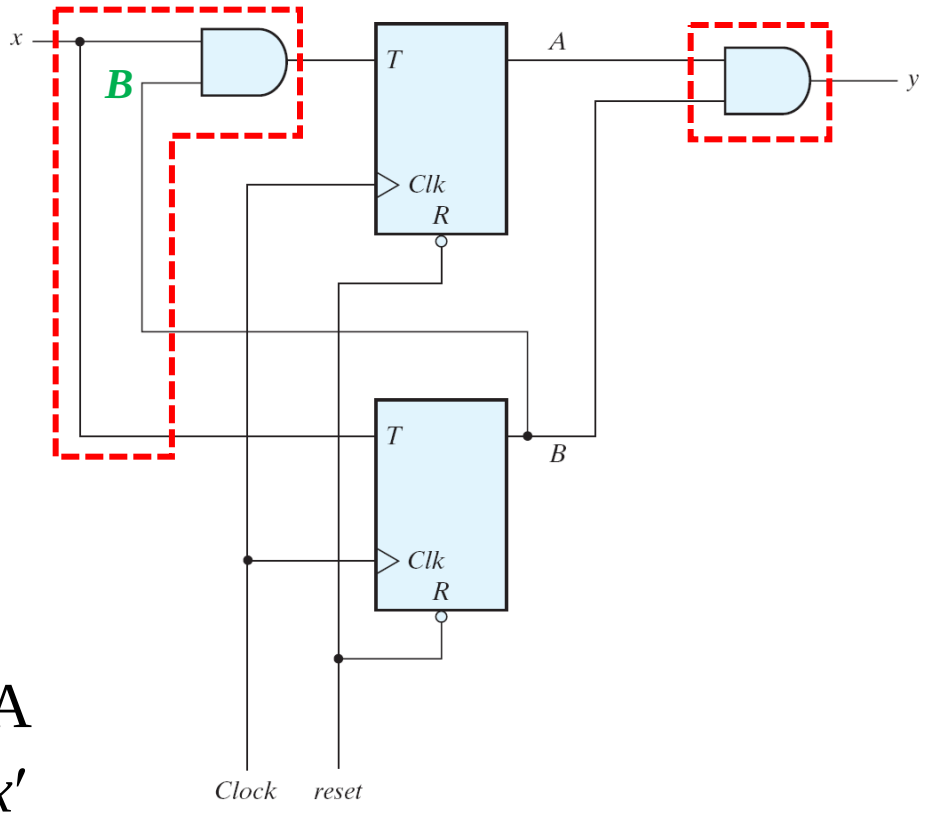
- Example:





1. Flip-flop input equations & Output equations:

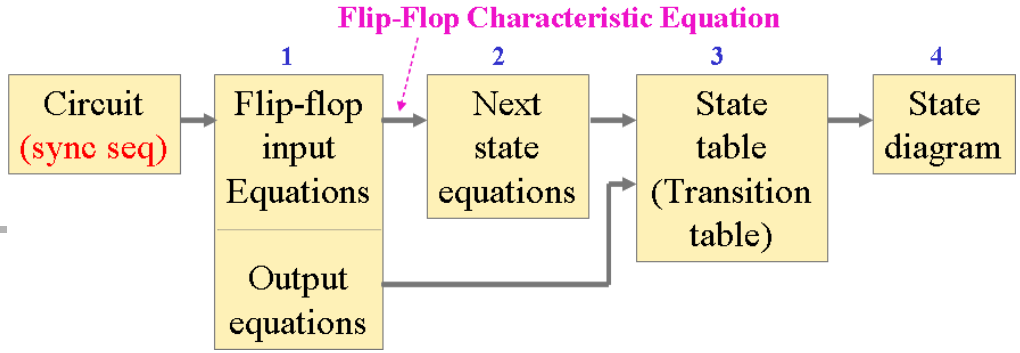
$$T_A = B\,x$$
$$T_B = x$$
$$y = A\,B$$



2. Next state equations:

$(Q^+ = T\,Q' + T'\,Q)$

$$A^+ = T_A\,A' + T_A'\,A$$
$$= (Bx)\,A' + (B\,x)'\,A$$
$$= A'Bx + AB' + Ax'$$
$$B^+ = T_B\,B' + T_B'\,B$$
$$= (x)\,B' + (x)'\,B = x \oplus B$$



3. State table:

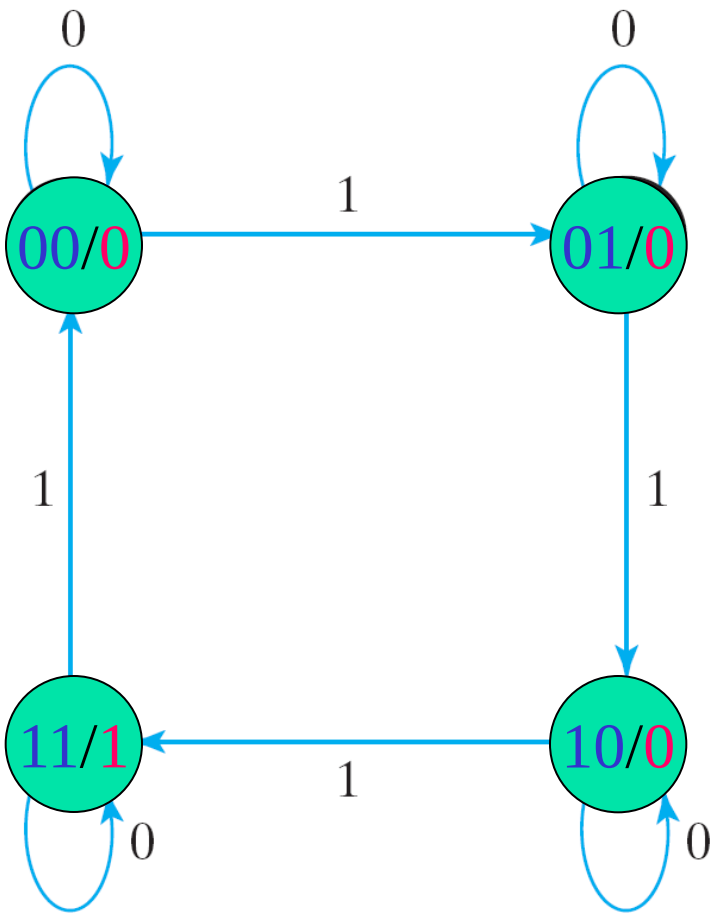
$A^+ = A' B x + A B' + A x'$

$B^+ = x \oplus B$

$y = A B$

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	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

4. 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	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

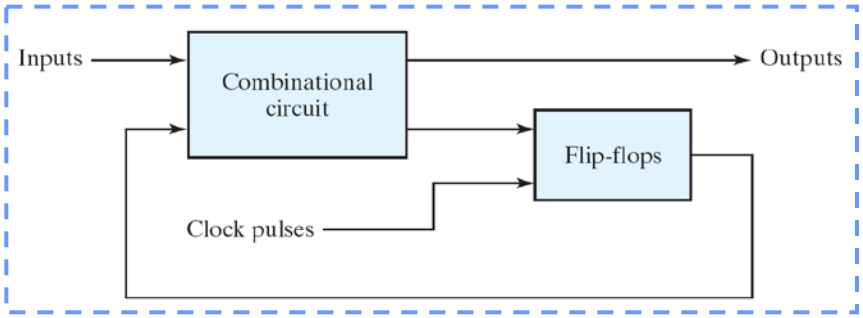
$$A^+ = A' B x + A B' + A x'$$

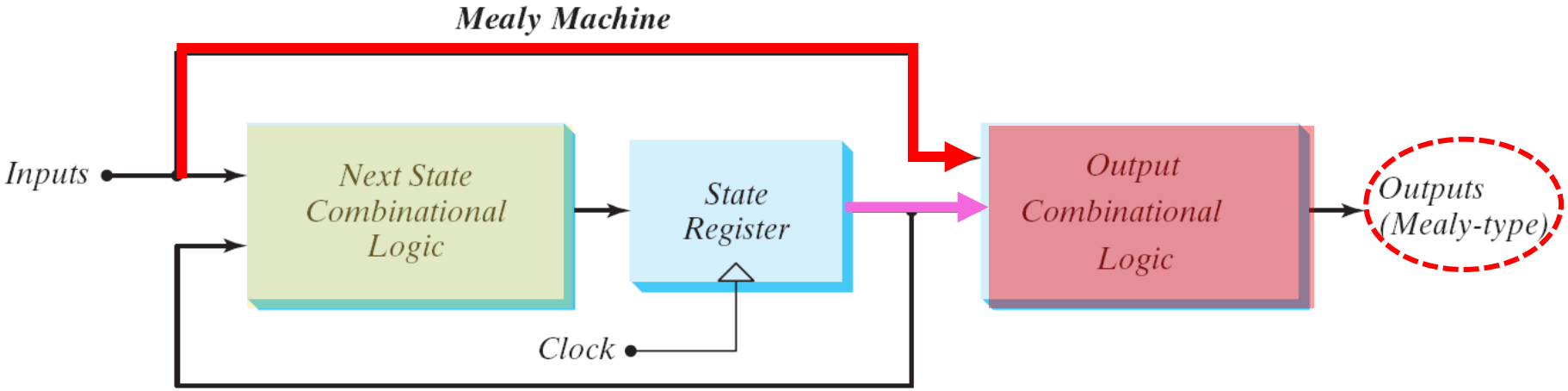
$$B^+ = x \oplus B$$

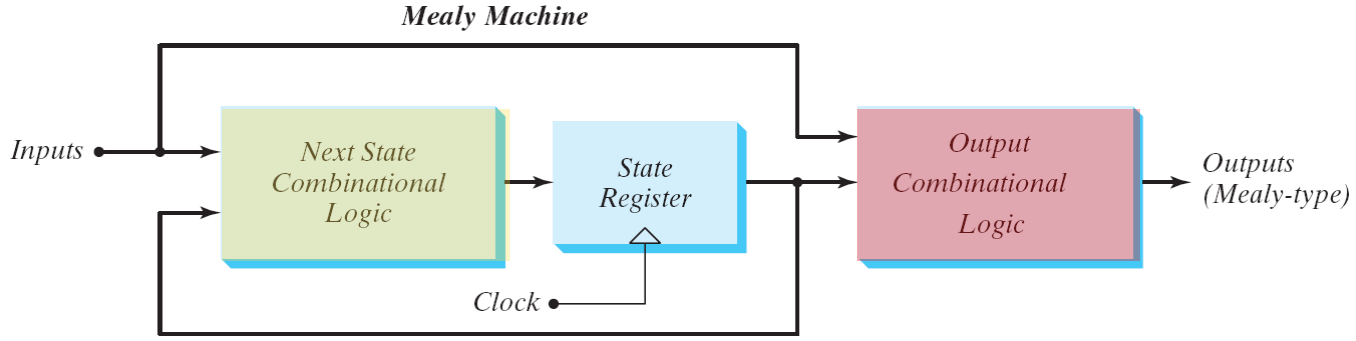
$$y = A B$$

D. Mealy & Moore Models of Finite State Machines

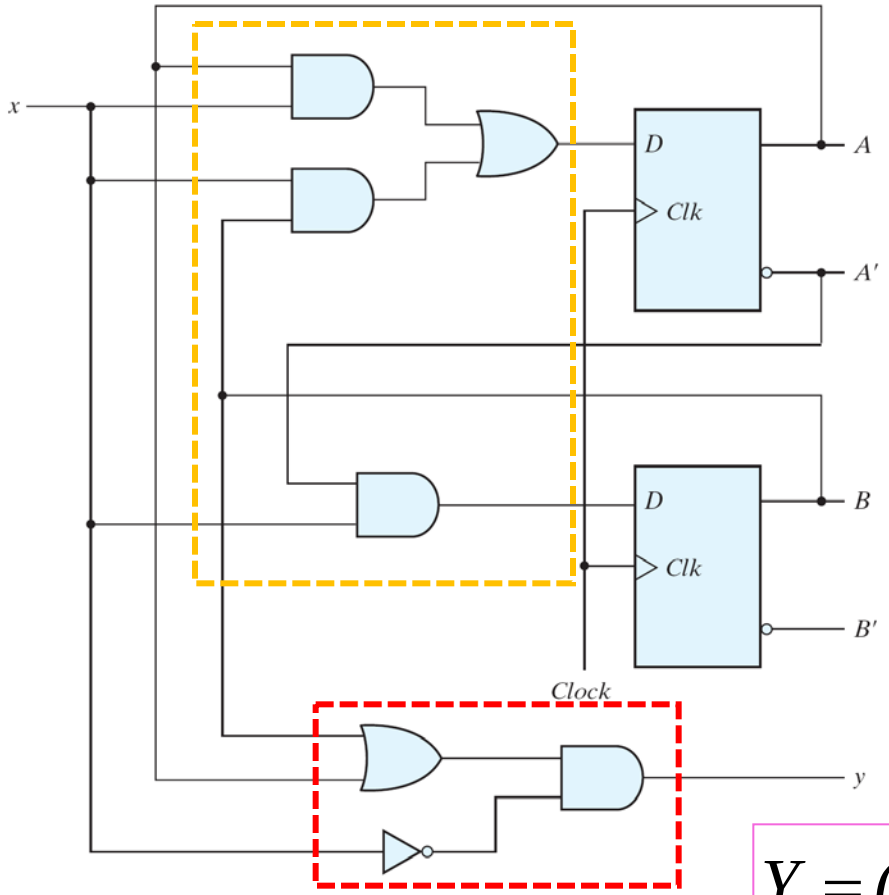
- Mealy model:
  - is a model of seq ckts in which the outputs depend on both the inputs and the present states.

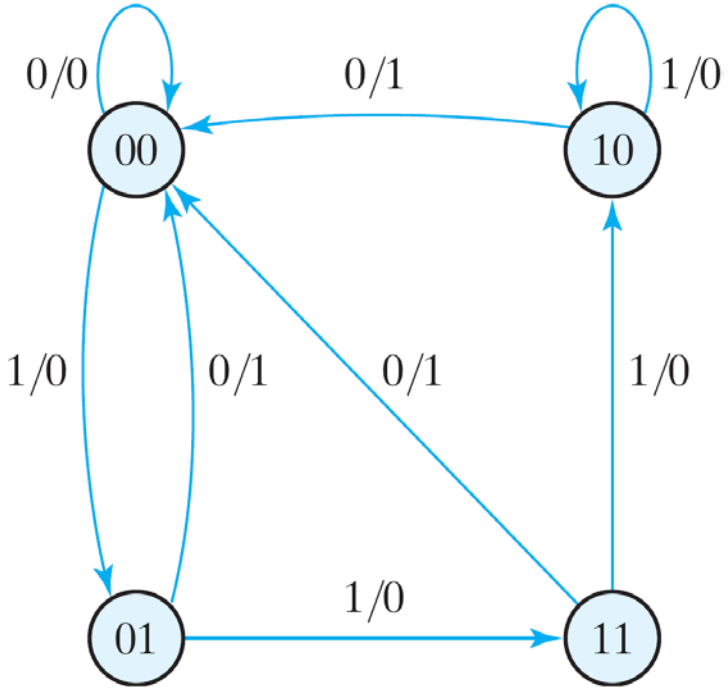






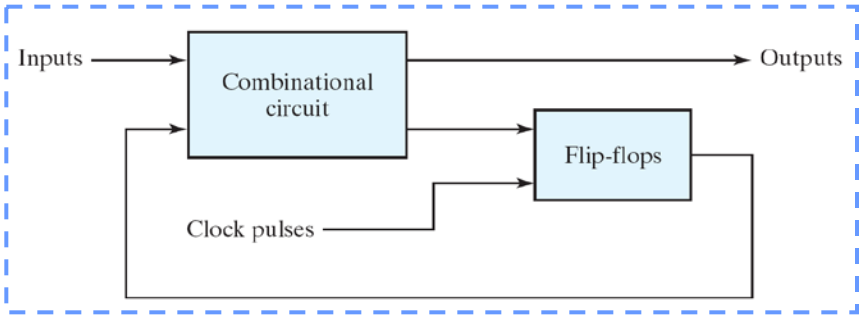
– E.g. 1:

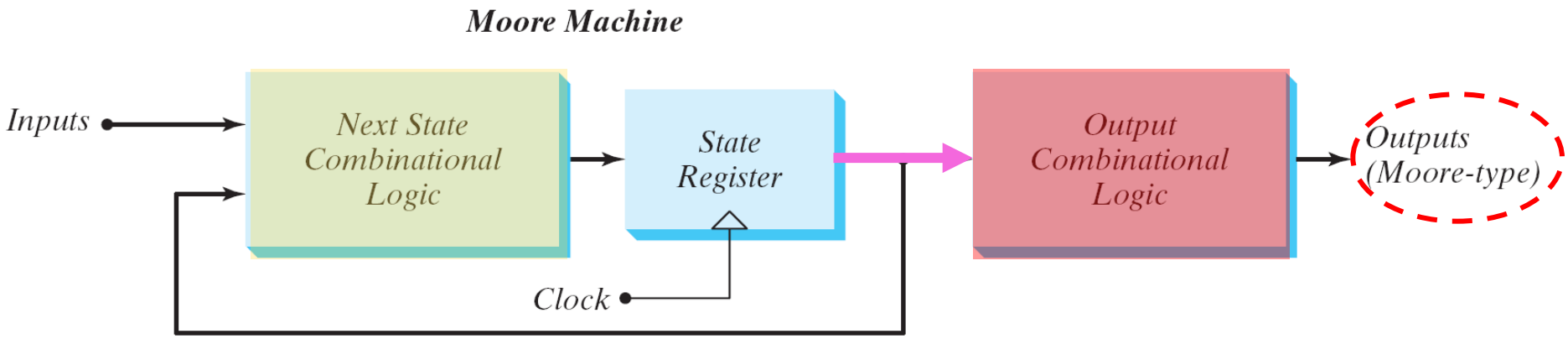


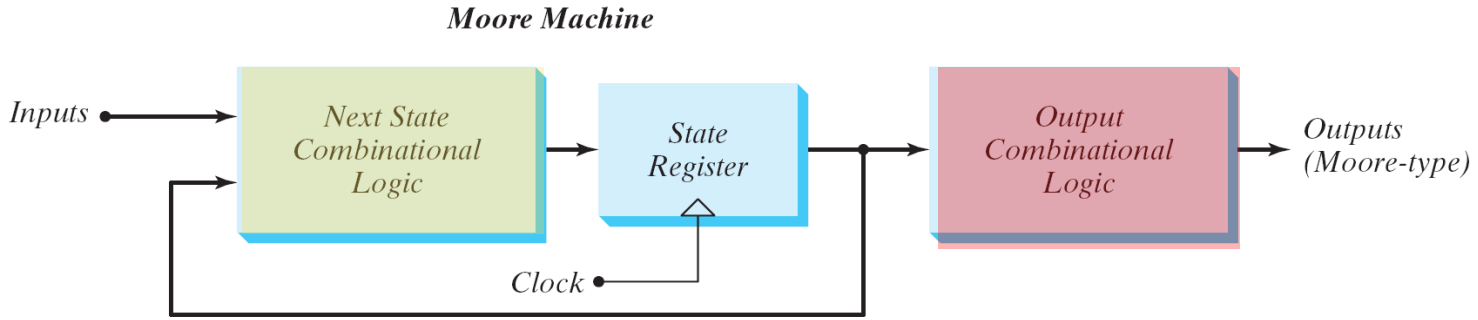


$$Y = (A + B)\overline{X}$$

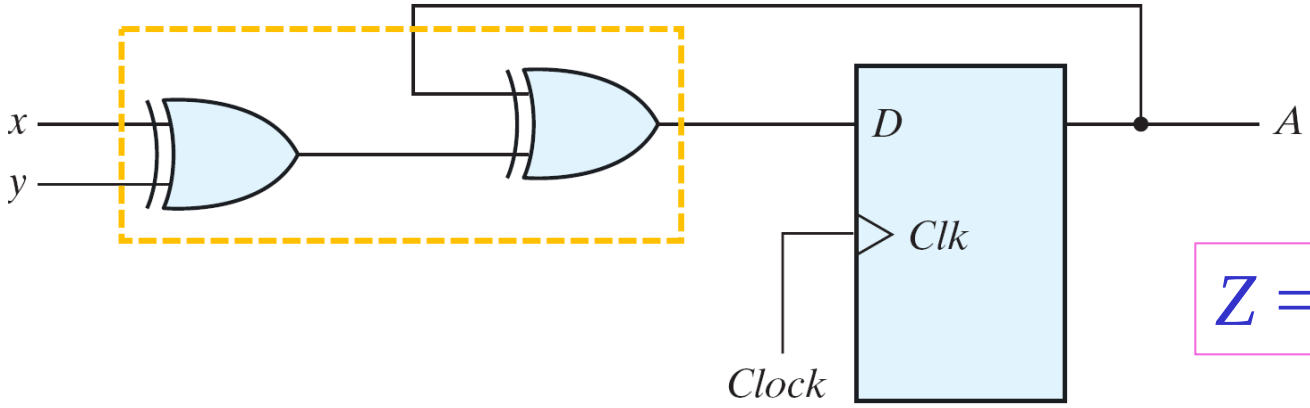
- Moore model:
  - is a model of seq ckts in which the outputs depend only on the present states.



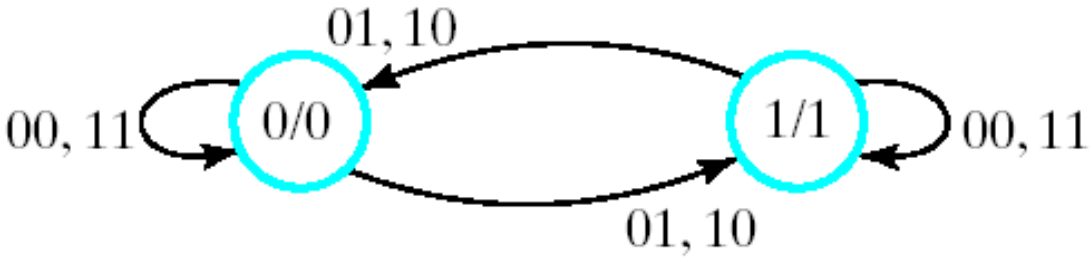



Moore Machine
Inputs
Next State Combinational Logic
State Register
Output Combinational Logic
Outputs (Moore-type)
Clock

– E.g. 2:

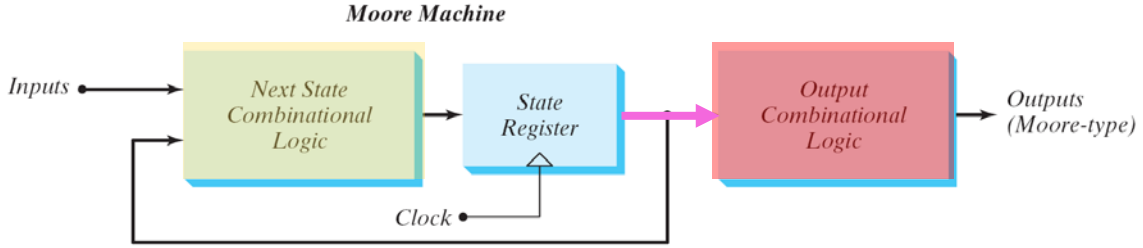

x
y
D
Clk
A
Clock

$Z = A$

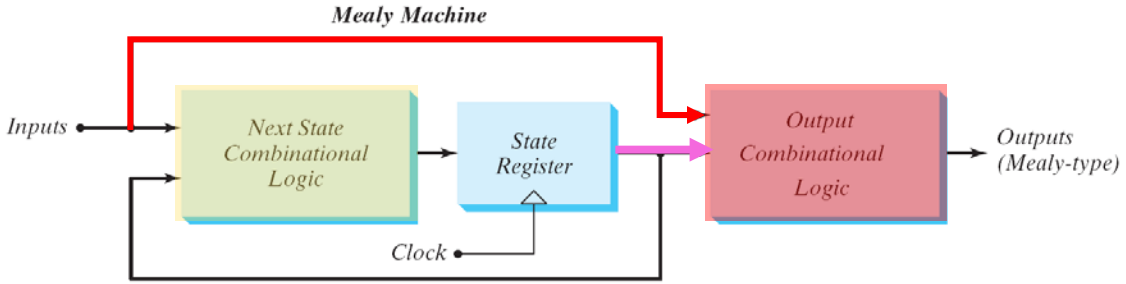

01, 10
00, 11
0/0
1/1
00, 11
01, 10

- Comparison:
  - Moore model: (state) → output
    - The outputs of the ckt are synchronized w/ the clock.

      (⇐ the outputs depend on only flip-flop outputs (states) that are synchronized w/ the clock)



  - Mealy model: (input, state) → output
    - The outputs may change if the inputs change during the clock cycle.

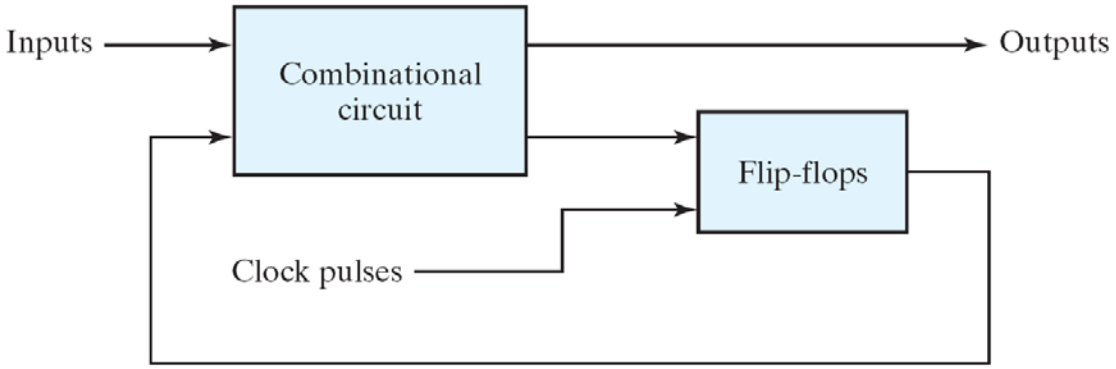


補充資料：Sequential Circuit Timing

- Analysis of a sync seq ckt:
  - analyze the function of the ckt $\Rightarrow$ State diagram
  - analyze the performance of the ckt
    - For comb. ckt: input-to-output delay
    - For sync seq ckt: the max clock frequency, f_{max} , at which the ckt can operate

      clock frequency $f = 1/t_p$, t_p: the clock period

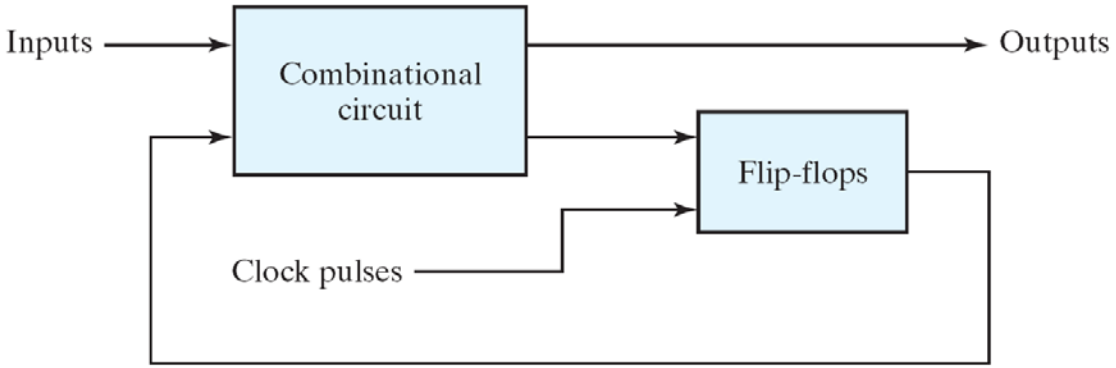
      $f_{max} = 1/t_{p\text{-}min}$, $t_{p\text{-}min}$: the min allowable clock period



(a) Block diagram

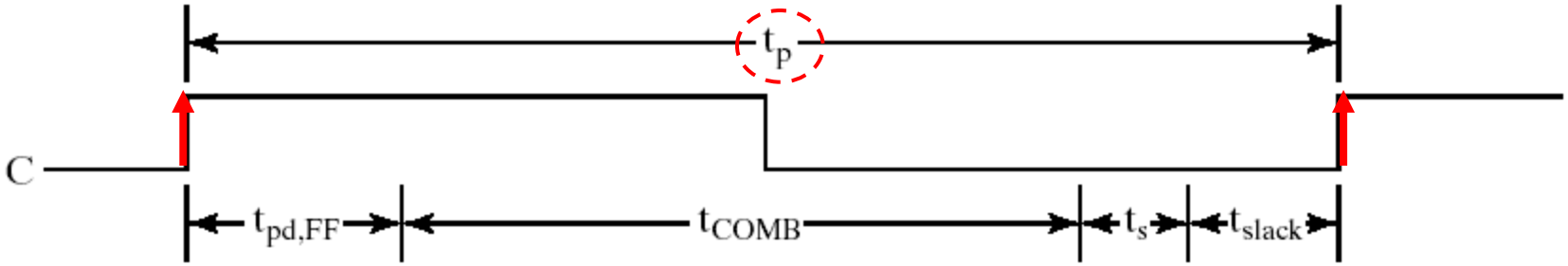
Sequential Circuit Timing Parameters

- Sequential ckt timing parameters:
  - i. f-f propagation delay: $t_{pd,\mathrm{FF}}$
  - ii. comb logic delay through the chain of gates along the path: $t_{pd,\mathrm{COMB}}$
  - iii. f-f setup time: t_s
  - (iv. slack time, t_{slack} : the extra time allowed in the clock period beyond that required by the path)

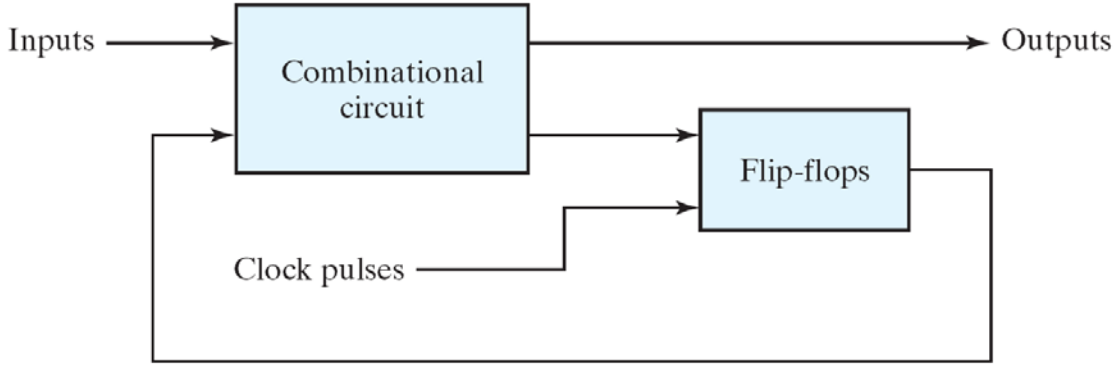


(a) Block diagram

- For edge-triggered flip-flops:
  - determine the longest delay from the triggering edge of the clock to the next triggering edge of the clock.



Edge-triggered (positive edge)



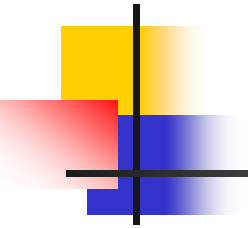
Block diagram

5-6 Synthesizable HDL Models of Sequential Circuits

- Behavioral Modeling of Sequential Circuit
- HDL Models of Flip-Flops and Latches
- State diagram-Based HDL Models
- Structural Description of Clocked Sequential Circuits

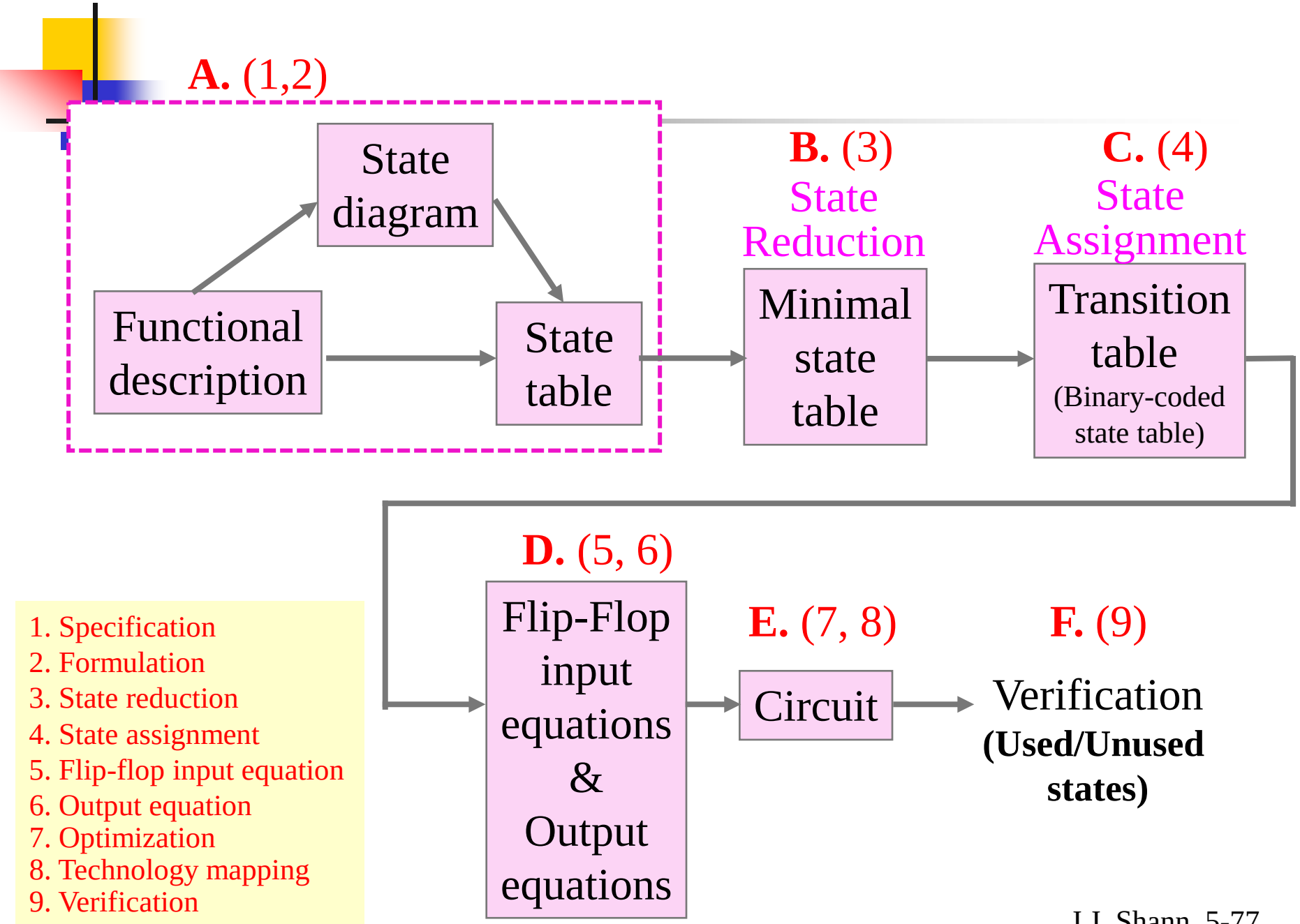
5-7 & 5-8 Design of Clocked Seq Circuits

- Design procedure:
  1. Specification: Write a specification for the ckt.
  2. **Formulation**: Obtain either a state diagram or state table from the statement of the problem. **(correctness)**
  3. State reduction: Reduce the # of states if necessary.
  4. State assignment: Assign binary codes to the states and obtain the binary-coded state table (or transition table).
  5. Flip-flop input equation determination: Select the flip-flop type or types. Derive the flip-flop input equations from the next-state entries in the encoded state table.
  6. Output equation determination: Derive output equations from the output entries in the encoded state table.

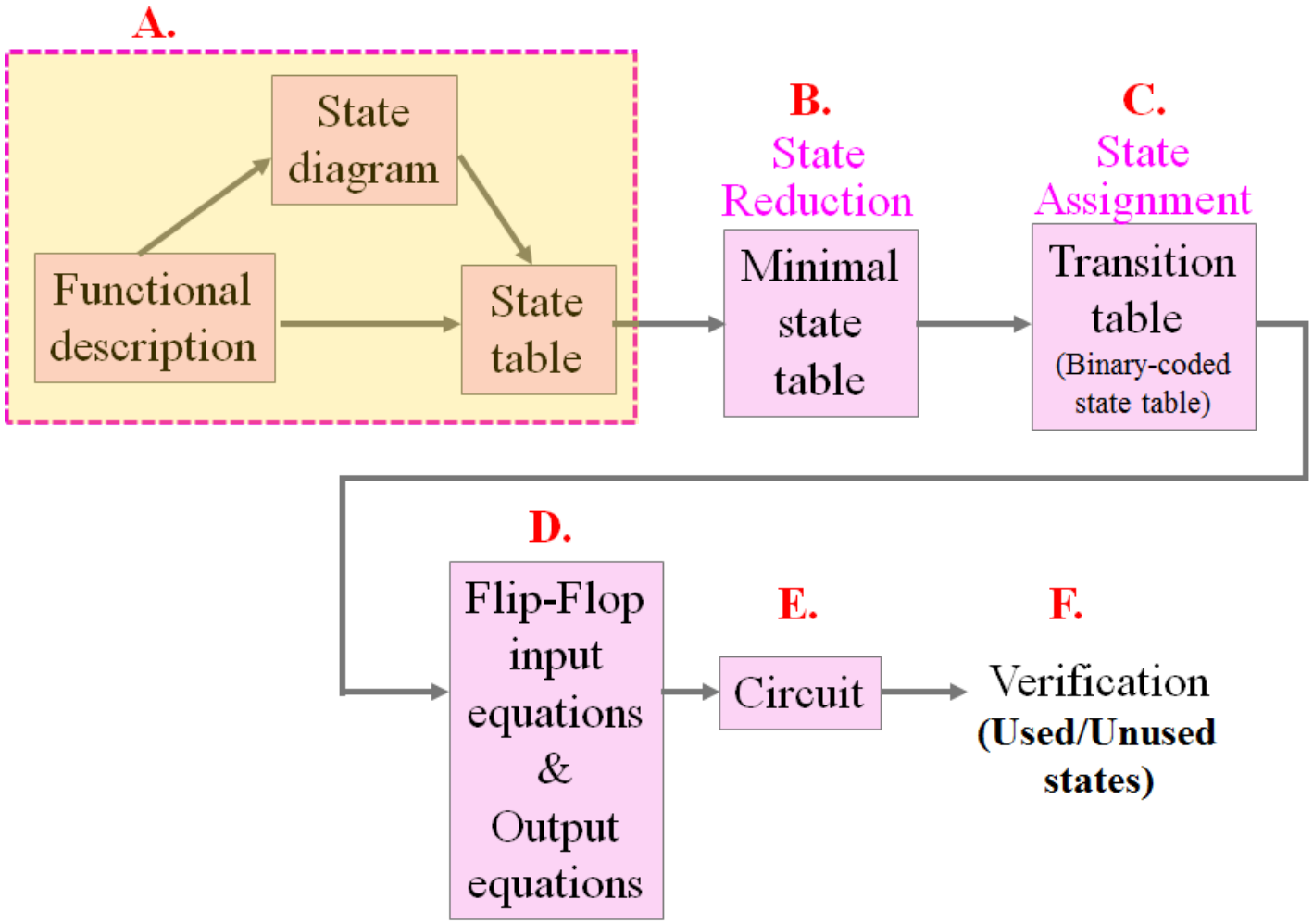


7. Optimization: Optimize the flip-flop input equations and output equations. (2-level or multiple-level)
8. Technology mapping: Draw a logic diagram of the ckt using flip-flops, ANDs, ORs, and inverters. Transform the logic diagram to a new diagram using the available flip-flop and gate technology.
9. Verification: Verify the correctness of the final design.

   (analysis of the sync seq circuit)


A. (1,2)
State diagram
Functional description
State table
B. (3)
State Reduction
Minimal state table
C. (4)
State Assignment
Transition table
(Binary-coded state table)
D. (5, 6)
Flip-Flop input equations & Output equations
E. (7, 8)
Circuit
F. (9)
Verification
(Used/Unused states)
1. Specification
2. Formulation
3. State reduction
4. State assignment
5. Flip-flop input equation
6. Output equation
7. Optimization
8. Technology mapping
9. Verification

A. Finding State Diagrams and State Tables



Example

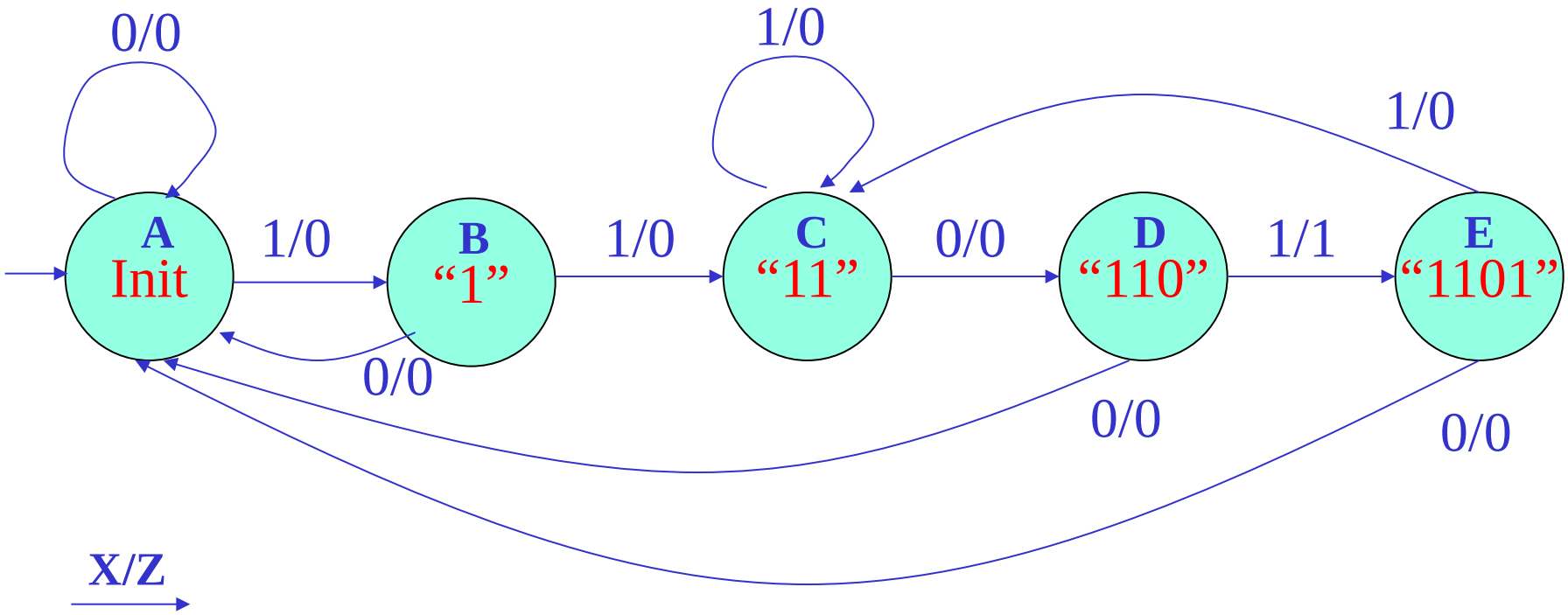
- E.g.: Sequence recognizer of "1101"
  - Problem description:
    - Recognize the occurrence of the sequence of bits 1101 on input X by making output Z equal to 1 when the previous three inputs to the circuit were 110 and current input is a 1, regardless of where it occurs in a sequence. Otherwise, Z equals 0.

      (Mealy-type FSM)
    - The circuit has ***direct resets*** on its flip-flops to initialize the state of the circuit to all zeros. (Initial state = 0…0)

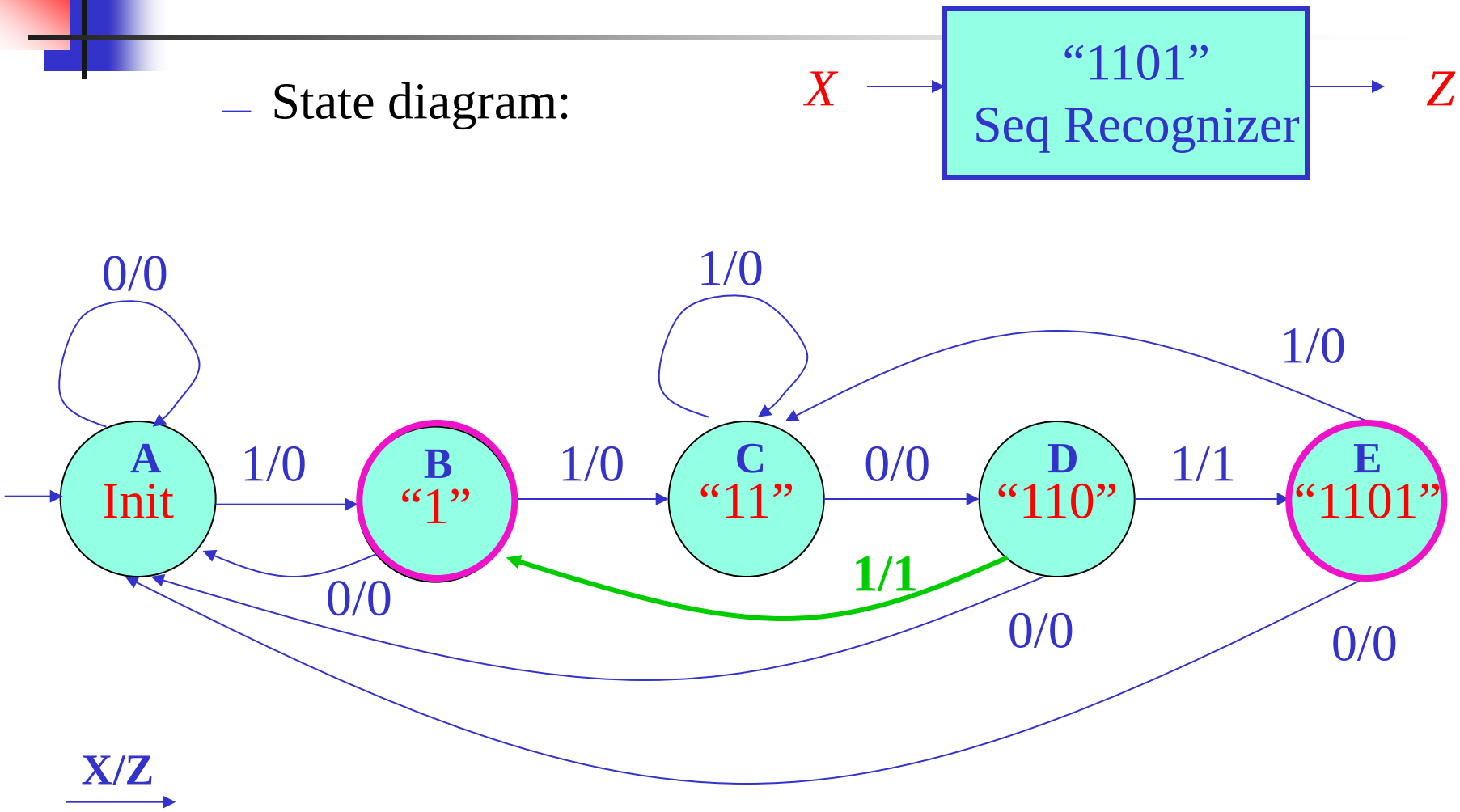
Mealy-type FSM

Mealy-type FSM

– State diagram:



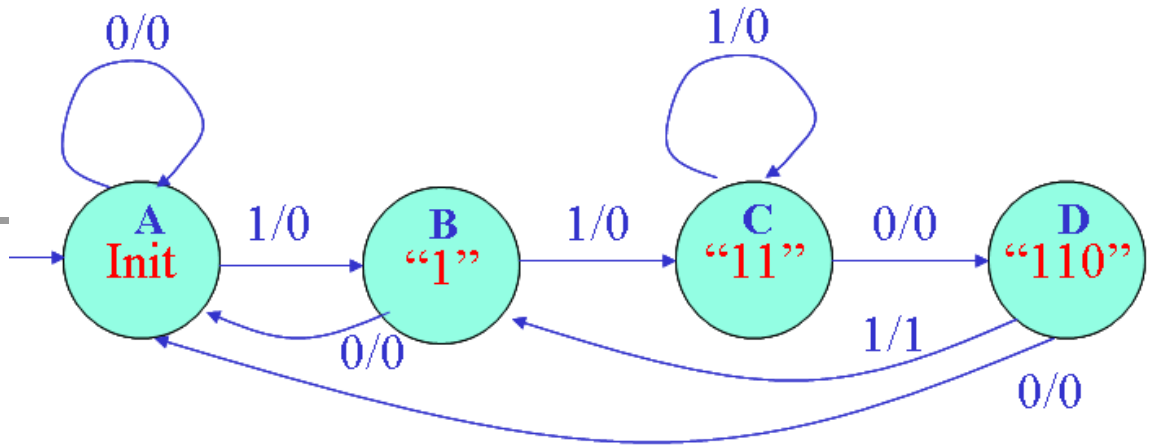
– State diagram:



State B = State E

– State table:

(2-D state table)



Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
A				
B				
C				
D				

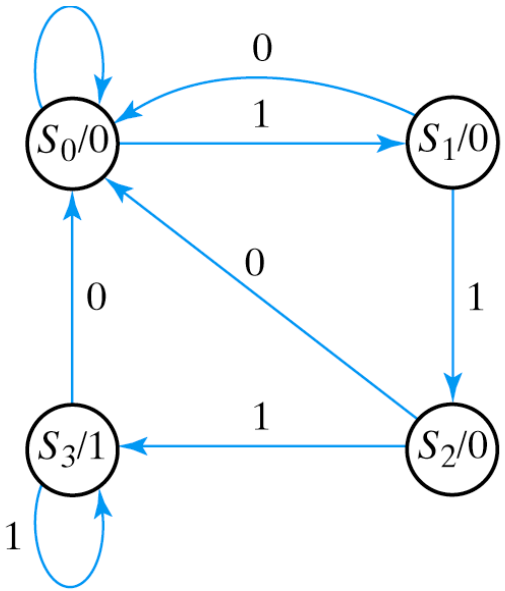
Example

- E.g.: Sequence detector of three or more consecutive 1's (Moore-type)

  Design a ckt that detects three or more consecutive 1's in a string of bits coming through an input line.

<Ans>

State diagram



State table:

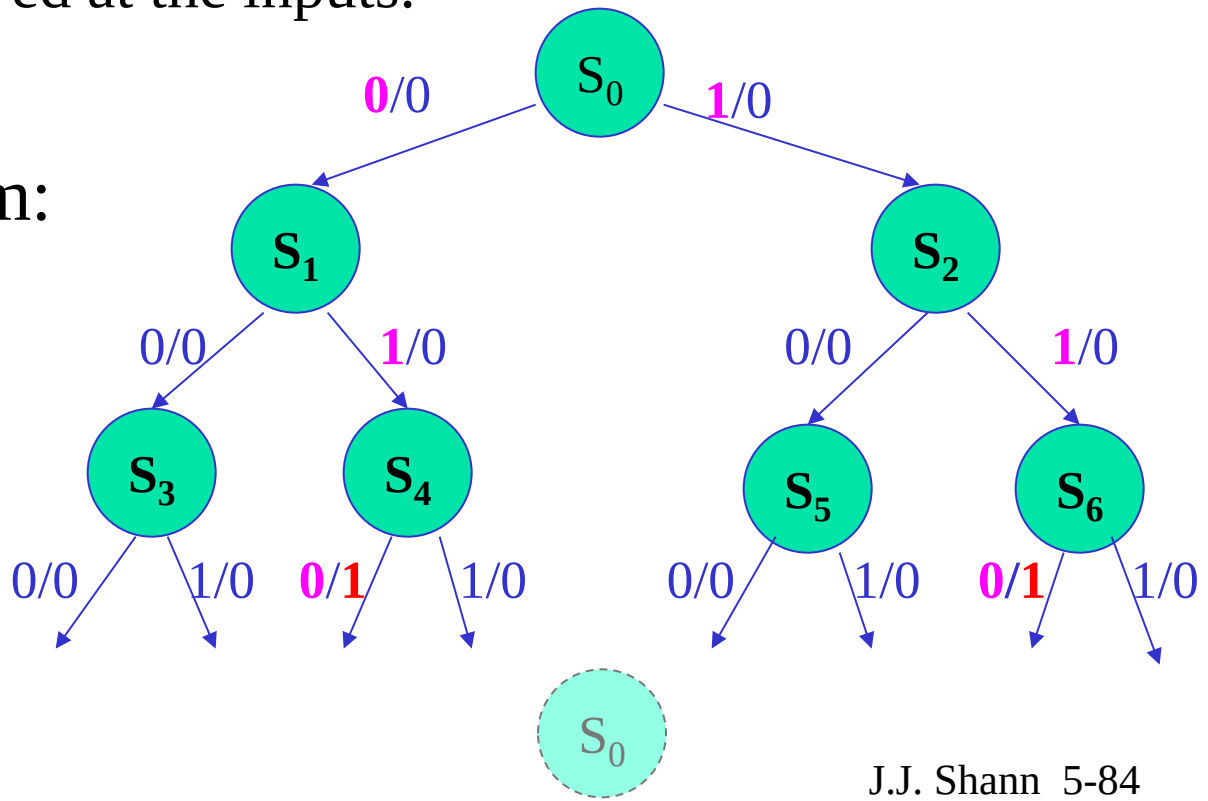
Present state	Input x	Next state	Output y
S_0	0	S_0	0
S_0	1	S_1	0
S_1	0	S_0	0
S_1	1	S_2	0
S_2	0	S_0	0
S_2	1	S_3	0
S_3	0	S_0	1
S_3	1	S_3	1

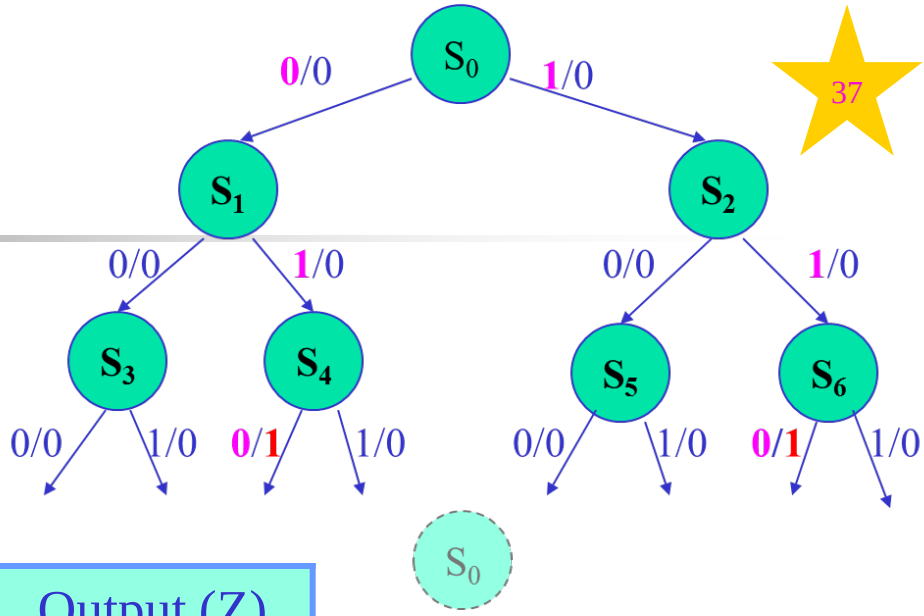
Example

- E.g.: 3-bit sequence detector (Mealy-type)
  - Single input X, Single output Z.
  - Output a 1 whenever the serial sequence 010 or 110 has been observed at the inputs.

<Ans.>

State diagram:

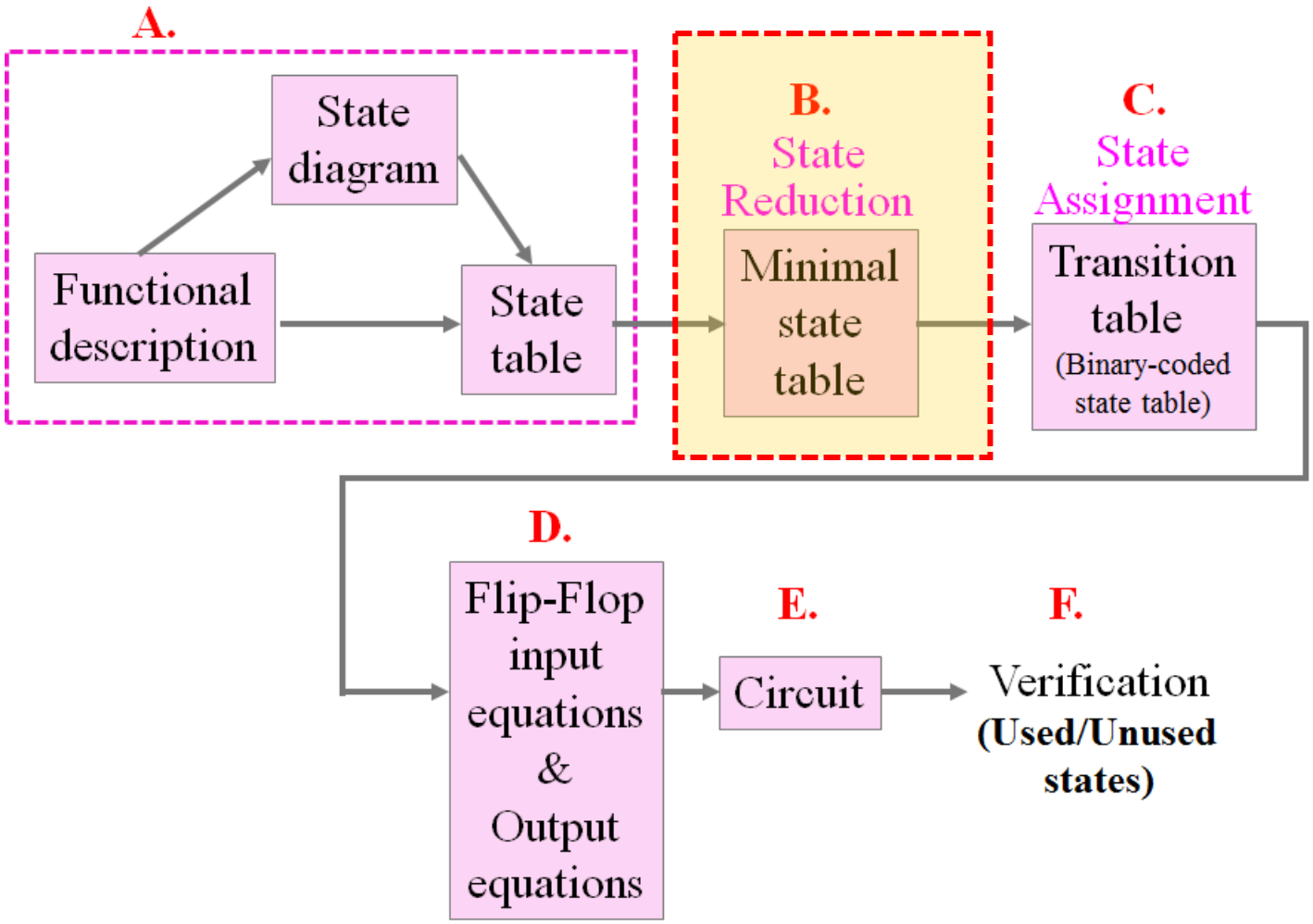




State table:

Input Sequence	Present State	Next State X = 0	Next State X = 1	Output (Z) X = 0	Output (Z) X = 1
Reset (Init)	S_0				
0	S_1				
1	S_2				
00	S_3				
01	S_4				
10	S_5				
11	S_6				

B. State Reduction → Minimal State Table

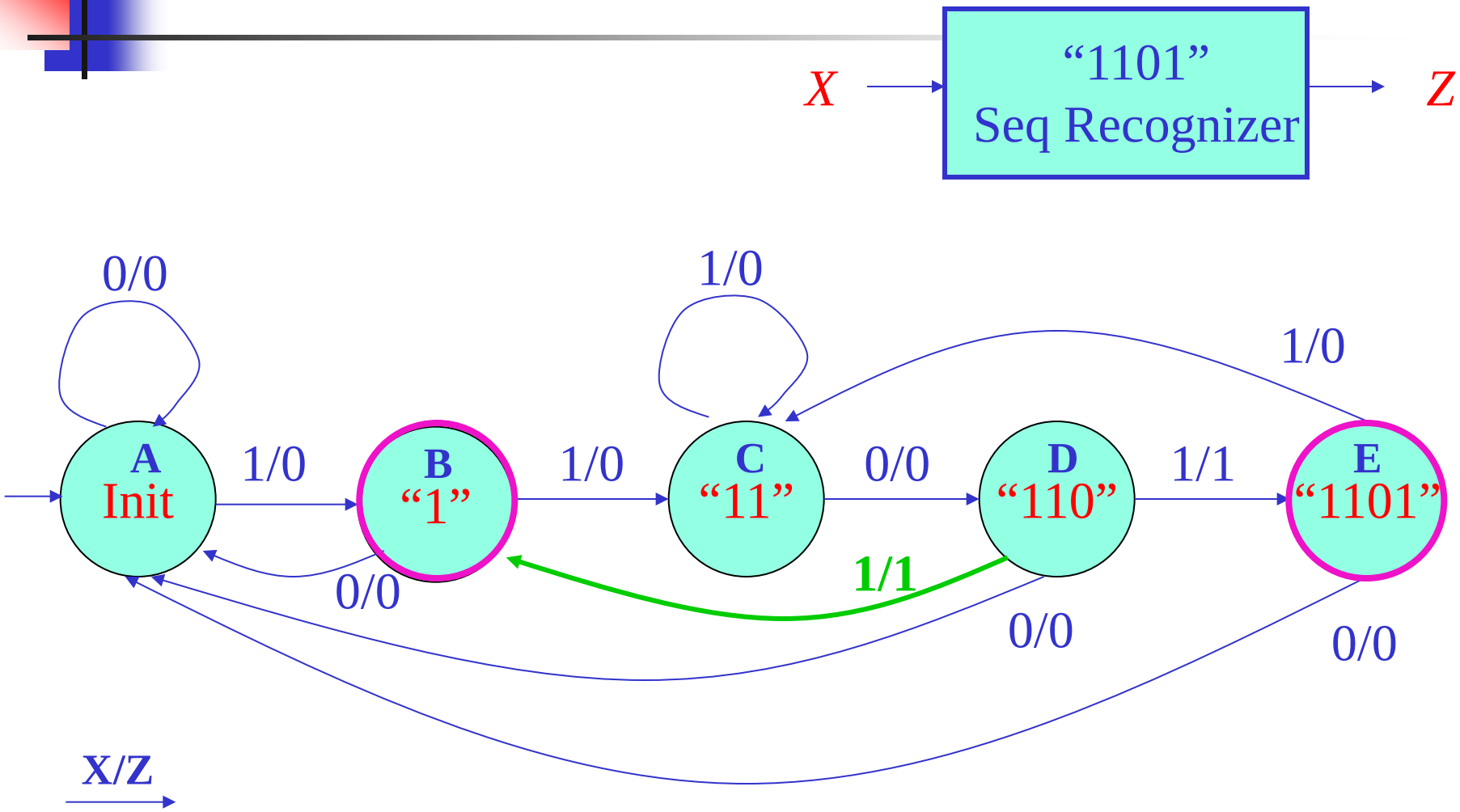


- State reduction:
  - Implement a seq ckt with fewest possible states

  ⇒ Least number of flip-flops
    - Boundaries are power of two number of states

  ⇒ Usually leads to more opportunities for don't cares
    - Reduce the number of gates needed for implementation (combinational ckt)
- Goal:
  - **Identify and combine states that have "*equivalent*" behavior**
  - ***Equivalent States*: for all input combinations, states transition to the *same* or *equivalent* states & outputs are the *same* or *compatible***

Example (p5-77~79)



State B = State E

State Reduction Methods (補充資料)

- State reduction methods:
  - Row matching method

    {equal states} ⇒ {equal states} ⇒ …
  - Implication chart method

    {unequivalent states} ⇒ {unequivalent states} ⇒ …

- Reference:
  - Randy H. Katz & Gaetano Borriello, *Contemporary Logic Design*, Prentice Hall.

(a) Row Matching Method

***Equivalent States*:**
for all input combinations,
states transition to the *same* or *equivalent* states &
outputs are the *same* or *compatible*

- Row matching method
  - Identify states w/ ***same*** output behavior.

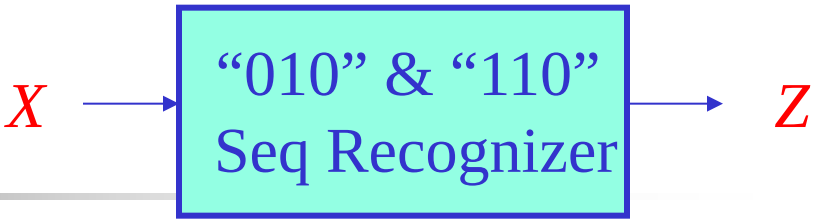
    If such states transition to the ***same*** next state, they are equivalent.

    Combine into a single new renamed state.

    Repeat until no new states are combined.
  - Adv.: Straightforward to understand and easy to implement
  - Problem: does not always yield the most reduced state table!

{equal states} ⇒ {equal states} ⇒ …

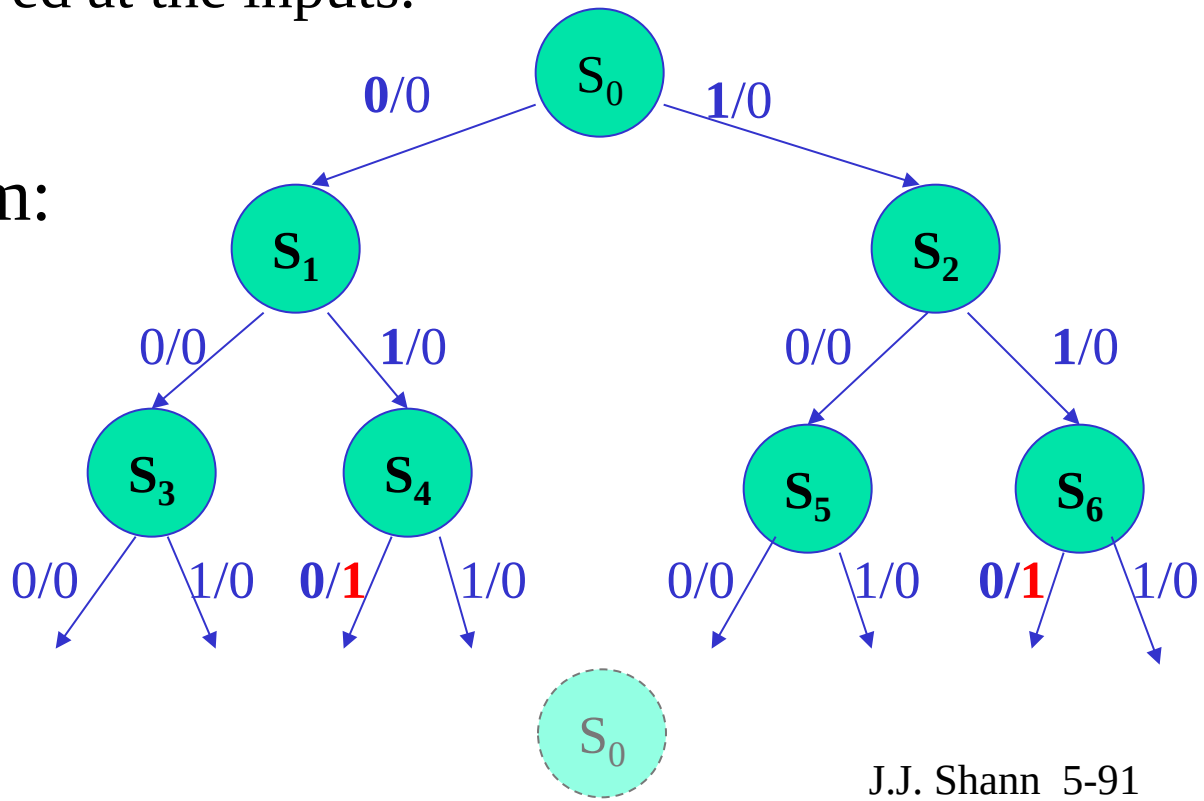
Example



- E.g.: 3-bit sequence detector
  - Single input X, Single output Z.
  - Output a 1 whenever the serial sequence 010 or 110 has been observed at the inputs.

<Ans.>

State diagram:



State table:

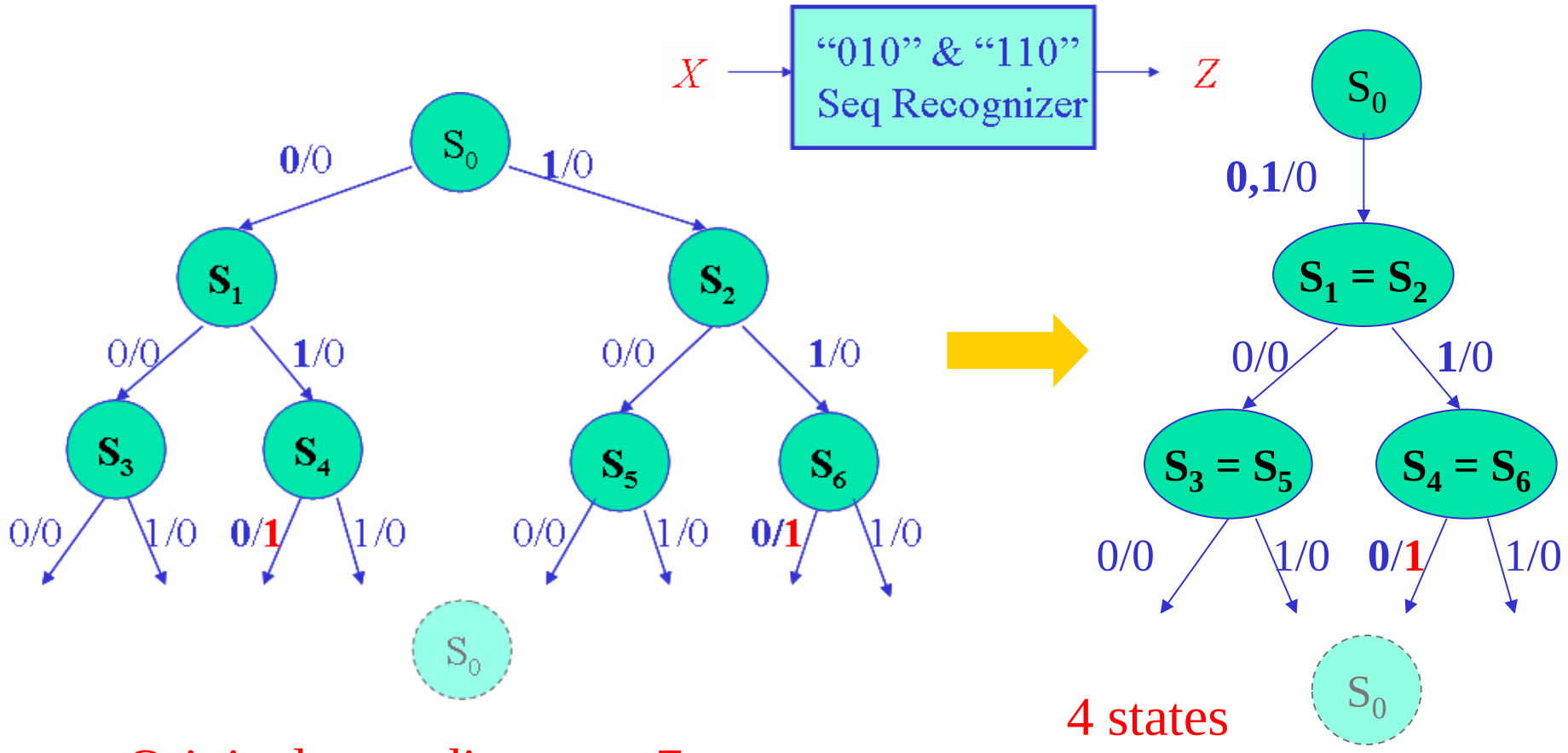
Input Sequence	Present State	Next State X = 0	Next State X = 1	Output (Z) X = 0	Output (Z) X = 1
Reset (Init)	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_3	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0

Reduced state table: (S1, S2) (S3, S5) (S4, S6)

	Input Sequence	Present State	Next State X = 0	Next State X = 1	Output (Z) X = 0	Output (Z) X = 1
	Reset (Init)	S_0				
(S1, S2)	0 or 1	S_1'				
(S3, S5)	00 or 10	S_3'				
(S4, S6)	01 or 11	S_4'				

Input Sequence	Present State	Next State X = 0	Next State X = 1	Output (Z) X = 0	Output (Z) X = 1
Reset (Init)	S_0	S_1'	S_1'	0	0
0 or 1	S_1'	S_3'	S_4'	0	0
00 or 10	S_3'	S_0	S_0	0	0
01 or 11	S_4'	S_0	S_0	1	0

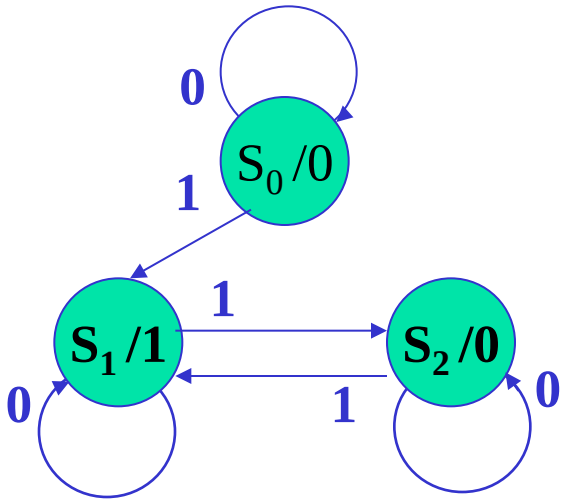
(S1, S2)
(S3, S5)
(S4, S6)



Original state diagram: 7 states

Example

- E.g.: 3-state odd parity checker (Moore type)



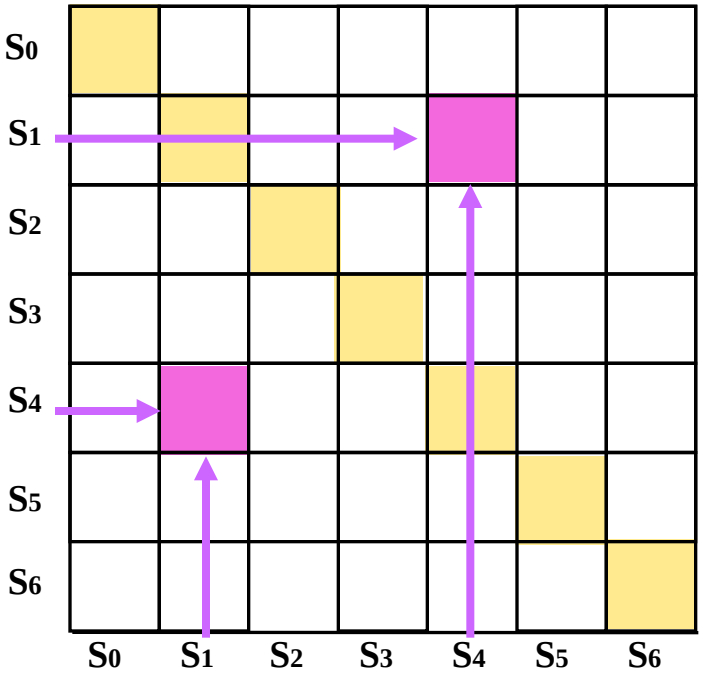
Present State	Next State X = 0	Next State X = 1	Output
S_0	S_0	S_1	0
S_1	S_1	S_2	1
S_2	S_2	S_1	0

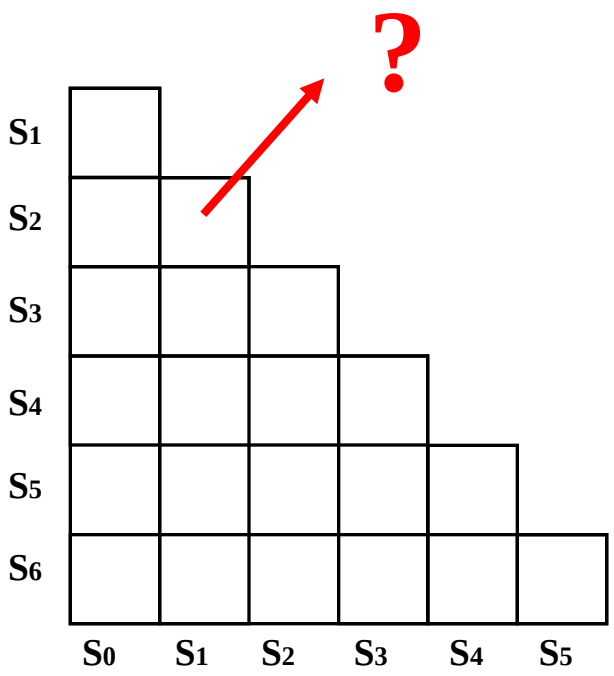
* None of the states may be reduced by the "*Row Matching*" Method.

(b) Implication Chart Method

- {unequivalent states} ⇒ {unequivalent states} ⇒ …
- Enumerate all possible combinations of states taken two at a time.

e.g.:



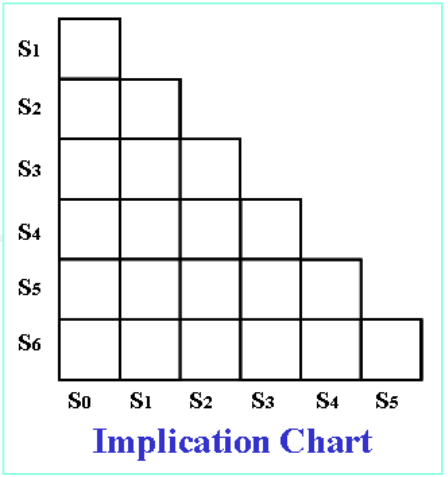


Implication Chart

x_{ii} can be eliminated (the diagonal)

x_{ij} will be the same as x_{ji}

{unequivalent states} ⇒ {unequivalent states} ⇒ …



- Filling in the implication chart:
  - x_{ij} : row is S_i, column is S_j
  - S_i is equivalent to S_j if their outputs are the same and next states are equivalent for each input combination
    - If S_i & S_j have different output behavior, then x_{ij} is crossed out.
    - x_{ij} contains the next states of S_i and S_j which must be equivalent if S_i and S_j are equivalent

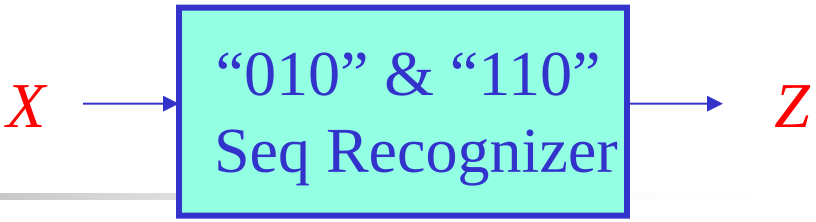
- Eg.:

Present State	Next State $x = 0$	Next State $x = 1$	Output
S_0	S_1	S_2	0
S_1	S_3	S_4	0
…	…	…	…

S_1 | S_1-S_3, S_2-S_4 | S_0

* Cross out x_{ij} if S_i and S_j is not possible to be equivalent.

Example



- E.g.: 3-bit sequence detector

  Single input X, Single output Z.

  Output a 1 whenever the serial sequence 010 or 110 has been observed at the inputs.

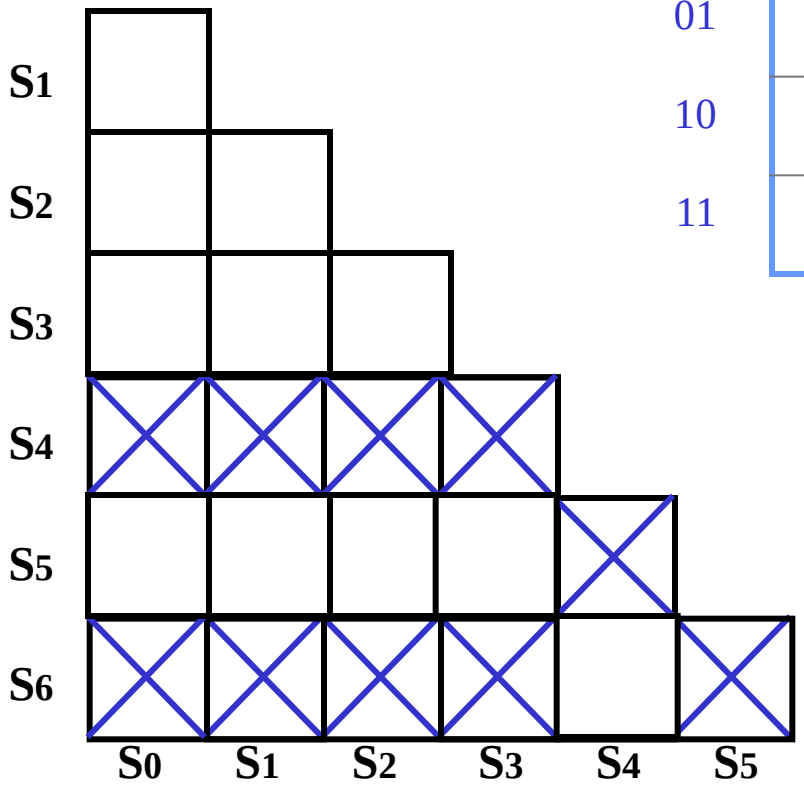
<Ans.>

State table:

Input Sequence	Prsent State	Next State X = 0	Next State X = 1	Output (Z) X = 0	Output (Z) X = 1
Reset (Init)	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0

Starting implication chart:

Input Sequence	Present State	Next State X = 0	Next State X = 1	Output (Z) X = 0	Output (Z) X = 1
Reset (Init)	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0

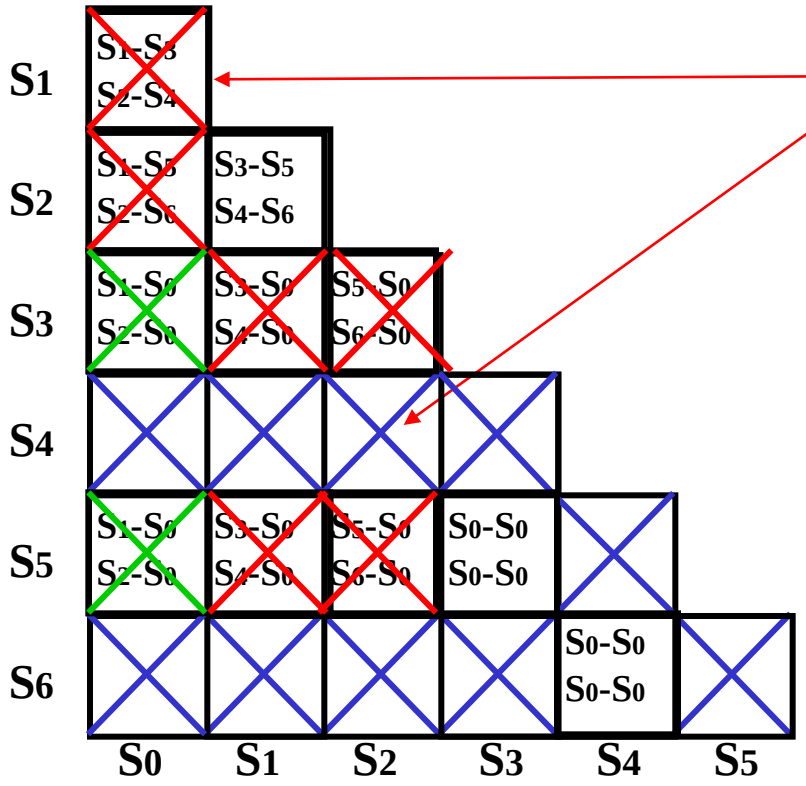


* If S_i & S_j have different output behavior, then x_{ij} is crossed out.

* x_{ij} contains the next states of S_i and S_j which must be equivalent if S_i and S_j are equivalent

Marking passes:

till no new "×" is added



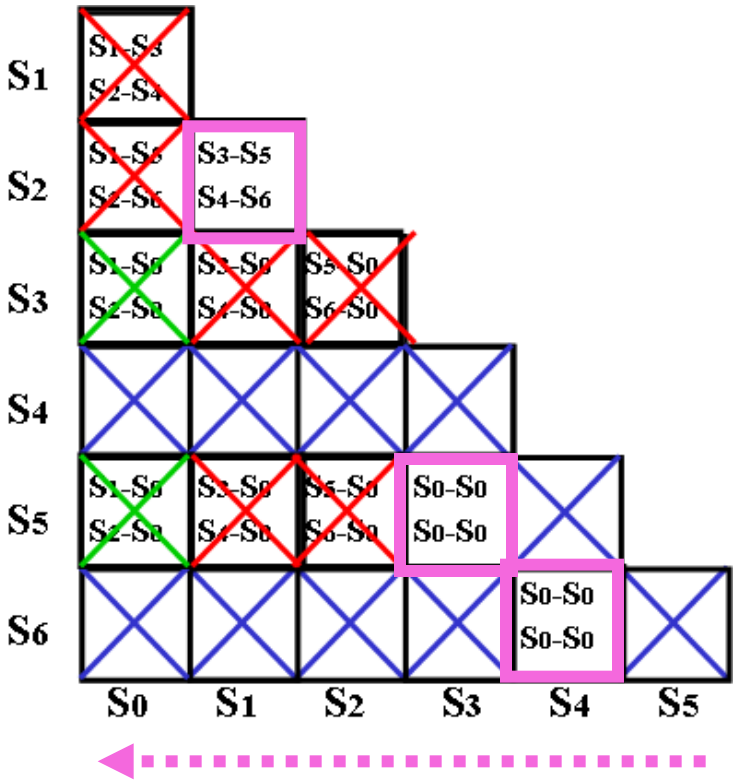
S_2 and S_4 have different output behavior

⇓

S_1 and S_0 cannot be combined

× ⇒ × ⇒ ×

Determining equivalent states:



S_4 and S_6 are equivalent

S_3 and S_5 are equivalent

$\Rightarrow$ S_1 and S_2 are too!

Original State Table

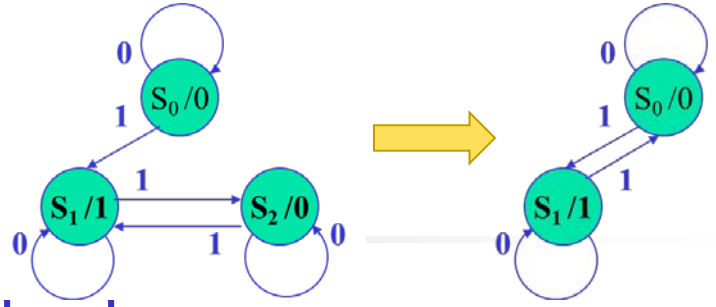
Input Sequence	Present State	Next State X = 0	Next State X = 1	Output (Z) X = 0	Output (Z) X = 1
Reset (Init)	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0

Reduced state table:

Equivalent states: (1,2), (3,5), (4,6)

	Input Sequence	Present State	Next State X = 0	Next State X = 1	Output (Z) X = 0	Output (Z) X = 1
	Reset (Init)	S_0	S_1'	S_1'	0	0
(S1, S2)	0 or 1	S_1'	S_3'	S_4'	0	0
(S3, S5)	00 or 10	S_3'	S_0	S_0	0	0
(S4, S6)	01 or 11	S_4'	S_0	S_0	1	0

Example

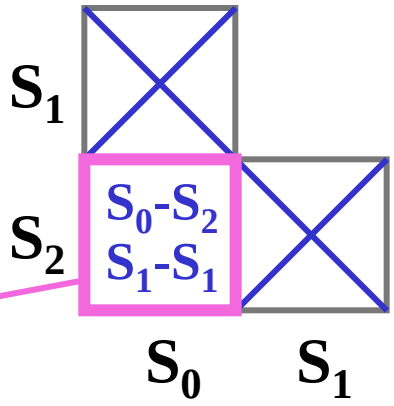


- E.g.: 3-state odd parity checker

* None of the states may be reduced by the *Row Matching Method*.

Present State	Next State X = 0	Next State X = 1	Output
S_0	S_0	S_1	0
S_1	S_1	~~S_2~~ 0	1
~~S_2~~	~~S_2~~	~~S_1~~	~~0~~

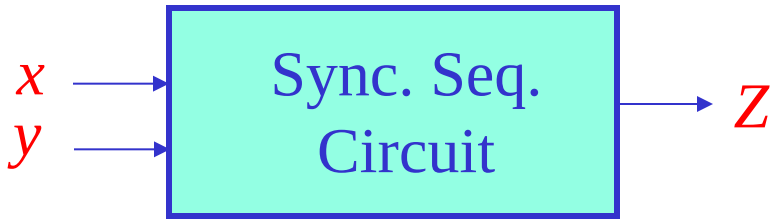
Implication Chart



S_0 is equivalent to S_2!

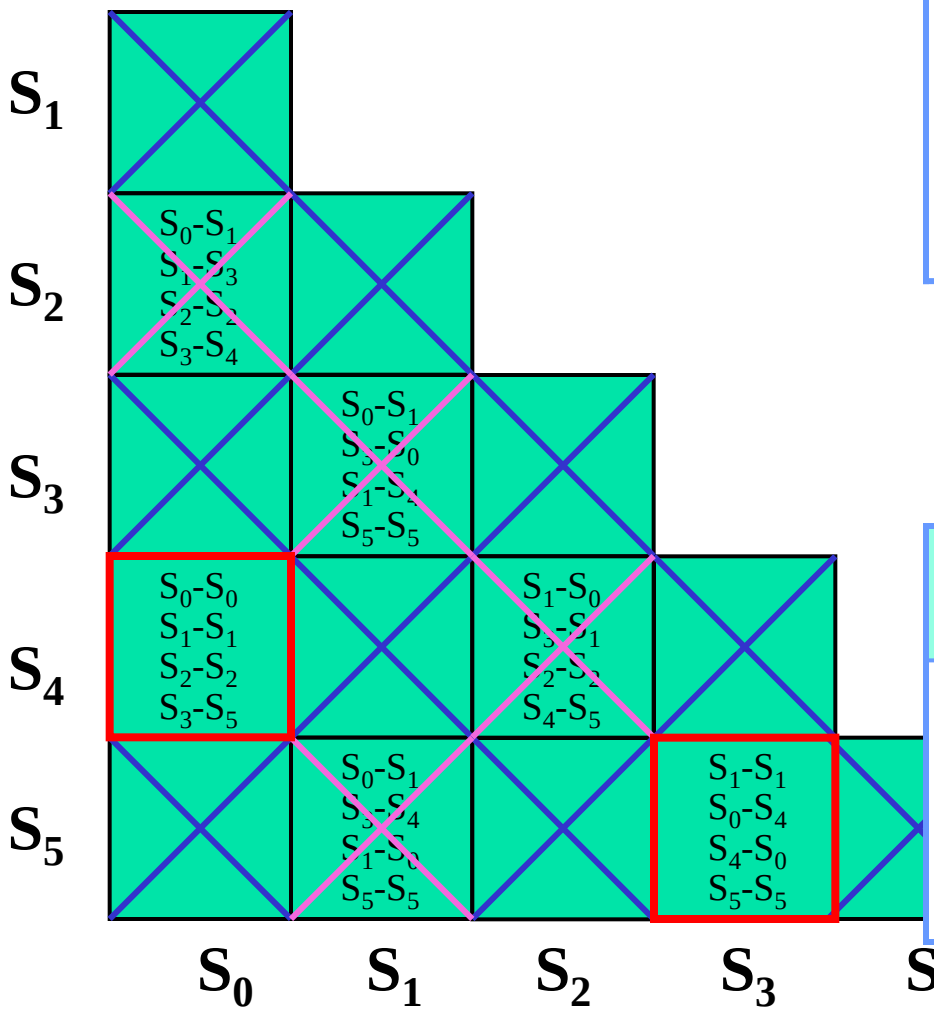
Multiple Input State Diagram Example

- E.g.: Multiple Input State Diagram Example



Prsent State	Next State xy = 00 01 10 11	Output (Z)
S_0	S_0 S_1 S_2 S_3	1
S_1	S_0 S_3 S_1 S_5	0
S_2	S_1 S_3 S_2 S_4	1
S_3	S_1 S_0 S_4 S_5	0
S_4	S_0 S_1 S_2 S_5	1
S_5	S_1 S_4 S_0 S_5	0

Symbolic State Table



Implication Chart

Prsent State	Next State xy = 00	01	10	11	Output (Z)
S_0	S_0	S_1	S_2	S_3	1
S_1	S_0	S_3	S_1	S_5	0
S_2	S_1	S_3	S_2	S_4	1
S_3	S_1	S_0	S_4	S_5	0
S_4	S_0	S_1	S_2	S_5	1
S_5	S_1	S_4	S_0	S_5	0

Symbolic State Table

Equivalent states:
(0,4), (3,5)

Prsent State	Next State xy = 00	01	10	11	Output (Z)
S_0	S_0	S_1	S_2	S_3	1
S_1	S_0	S_3	S_1	S_3	0
S_2	S_1	S_3	S_2	S_0	1
S_3	S_1	S_0	S_0	S_3	0

Minimized State Table

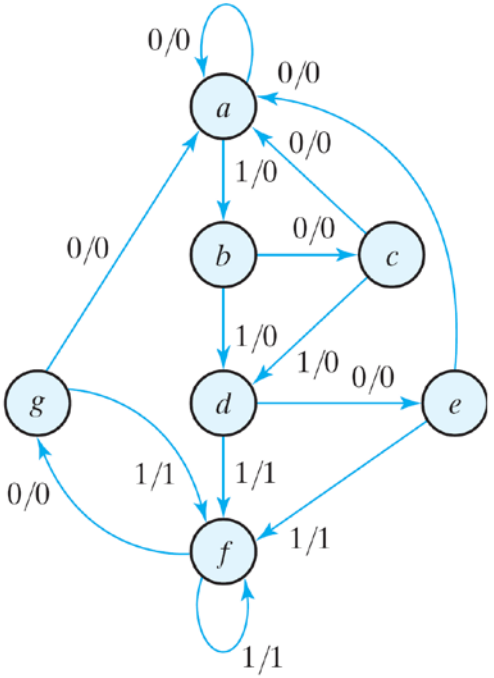
Detailed Algorithm of Imp. Chart Method

1. Construct implication chart, one square for each combination of states taken two at a time.
2. For each square labeled (S_i, S_j), if outputs differ then mark the square w/ "×". Otherwise write down implied state pairs for all input combinations
3. Advance through chart top-to-bottom and left-to-right. If square (S_i, S_j) contains an implied pair S_m-S_n and square (S_m, S_n) already labeled "×", then (S_i, S_j) is labeled "×".
4. Continue executing Step 3 until no new squares are marked with "×".
5. For each remaining unmarked square (S_i, S_j), then S_i and S_j are equivalent.

Example

State Table

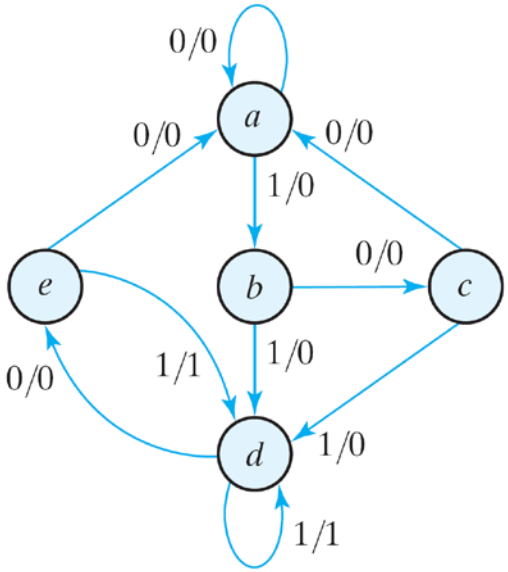
Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	~~f~~ d	0	1
e	a	~~f~~ d	0	1
~~f~~	~~g~~ e	~~f~~	~~0~~	~~1~~
~~g~~	~~a~~	~~f~~	~~0~~	~~1~~



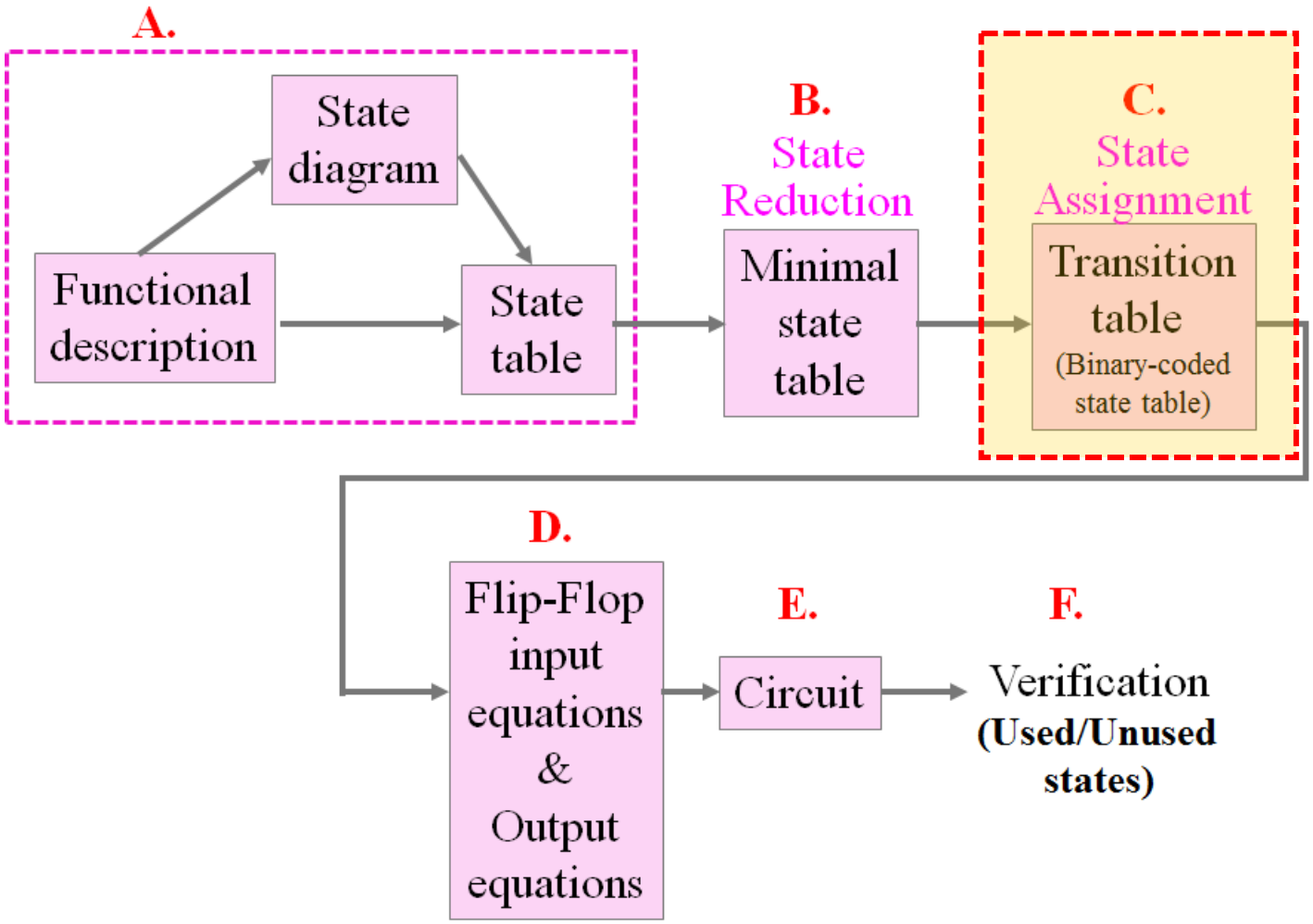
state reduction (d, f) (e, g)

Reduced State Table

Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1



C. State Assignment



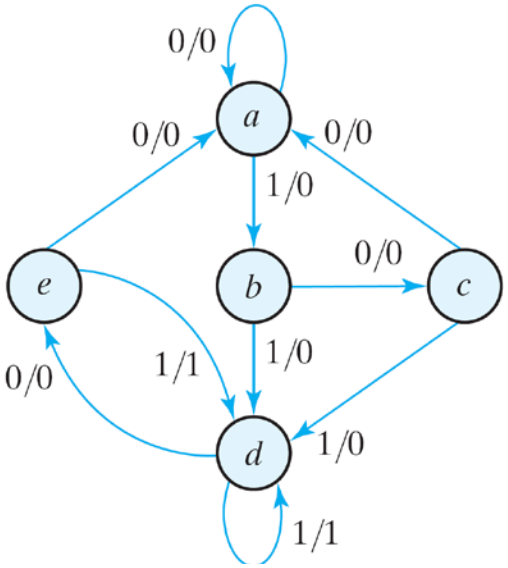
State Assignment

- State assignment:
  - assign unique coded binary values to the states
  - For a circuit with m states, the codes must contain n bits, $2^n \geq m$. ($n \geq \lceil \log_2 m \rceil$)
- Transition table: Binary coded state table
  - a state table w/ a binary assignment
- When a seq ckt implemented w/ gate logic, # gates will depend on mapping b/t symbolic state names and binary encodings.

Example

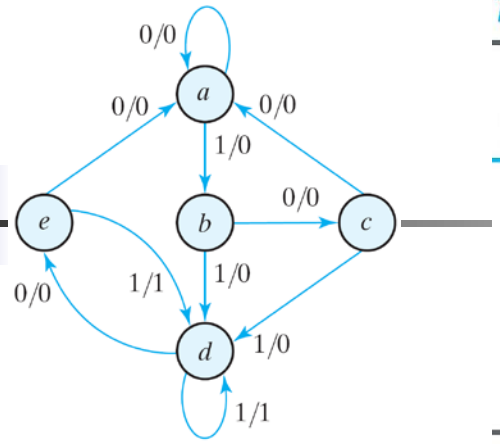
Reduced State Table

Present State	Next State $x = 0$	Next State $x = 1$	Output $x = 0$	Output $x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1



Three Possible Binary State Assignments

State	Assignment 1, Binary	Assignment 2, Gray Code	Assignment 3, One-Hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000



Three Possible Binary State Assignments

State	Assignment 1, Binary	Assignment 2, Gray Code	Assignment 3, One-Hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

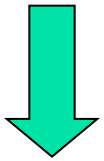
- Binary assignment
- Gray code assignment:
  - only one bit in the code group changes when going from one number to the next
  - makes it easier for the Boolean functions to be placed in the map for simplification
- One-hot assignment: one flip-flop per state
  - uses as many bits as there are states in the ckt

    At any given time, only one bit is equal to 1 while all others are kept at 0.
  - usually leads to simpler decoding logic for the next state and output

Reduced State Table

Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

State	Assignment 1, Binary
a	000
b	001
c	010
d	011
e	100



transition table

Reduced State Table with Binary Assignment 1

Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

* Effect of Different Assignments on Next State Map

$Q_2Q_1Q_0$	$Q_2^+Q_1^+Q_0^+$	
Current State	Next State $x=0$	$x=1$
(*a*) 000	000	001
(*b*) 001	010	011
(*c*) 010	000	011
(*d*) 011	100	011
(*e*) 100	000	011

(Assignment 1)

Q_2^+

Q_0X \ Q_2Q_1	00	01	11	10
00	0	0	×	0
01	0	0	×	0
11	0	0	×	×
10	0	1	×	×

Q_1^+

Q_0X \ Q_2Q_1	00	01	11	10
00	0	0	×	0
01	0	1	×	1
11	1	1	×	×
10	1	0	×	×

Q_0^+

Q_0X \ Q_2Q_1	00	01	11	10
00	0	0	×	0
01	1	1	×	1
11	1	1	×	×
10	0	0	×	×

$Q_2Q_1Q_0$	$Q_2^+Q_1^+Q_0^+$	
Current State	Next State $x=0$	$x=1$
(*a*) 000	000	001
(*b*) 001	011	010
(*c*) 011	000	010
(*d*) 010	110	010
(*e*) 110	000	010

(Assignment 2)

Q_2^+

Q_0X \ Q_2Q_1	00	01	11	10
00	0	1	0	×
01	0	0	0	×
11	0	0	×	×
10	0	0	×	×

Q_1^+

Q_0X \ Q_2Q_1	00	01	11	10
00	0	1	0	×
01	0	1	1	×
11	1	1	×	×
10	1	0	×	×

Q_0^+

Q_0X \ Q_2Q_1	00	01	11	10
00	0	0	0	×
01	1	0	0	×
11	0	0	×	×
10	1	0	×	×

Construct the Binary-coded State Table

- Minimal state table —State assignment→ Transition table (Binary-coded state table)
  - Unused states: don't-care conditions
- Eg.:

Present State	Next State X = 0	Next State X = 1	Output Z X = 0	Output Z X = 1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

⇓ **A = 00, B = 01, C = 11, D = 10**

Transition table (Binary-coded state table)

Present State	Next State X = 0	Next State X = 1	Output Z X = 0	Output Z X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

Example

- E.g.: 4 states — S_0, S_1, S_2, S_3

  4 coded binary values — 00, 01, 10, 11

4 states $\Rightarrow$ 4 choices for first state, 3 for second, 2 for third, 1 for last

$\Rightarrow$ 24 different encodings (4!)

S_0	S_1	S_2	S_3
00	01	10	11
00	01	11	10
00	10	01	11
00	10	11	01
00	11	01	10
00	11	10	01
01	00	10	11
01	00	11	10
01	10	00	11
01	10	11	00
01	11	00	10
01	11	10	00

S_0	S_1	S_2	S_3
10	00	01	11
10	00	11	01
10	01	00	11
10	01	11	00
10	11	00	01
10	11	01	00
11	00	01	10
11	00	10	01
11	01	00	10
11	01	10	00
11	10	00	01
11	10	01	00

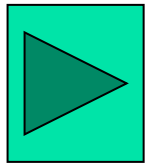
Symbolic State Names:
S_0, S_1, S_2, S_3

24 state assignments

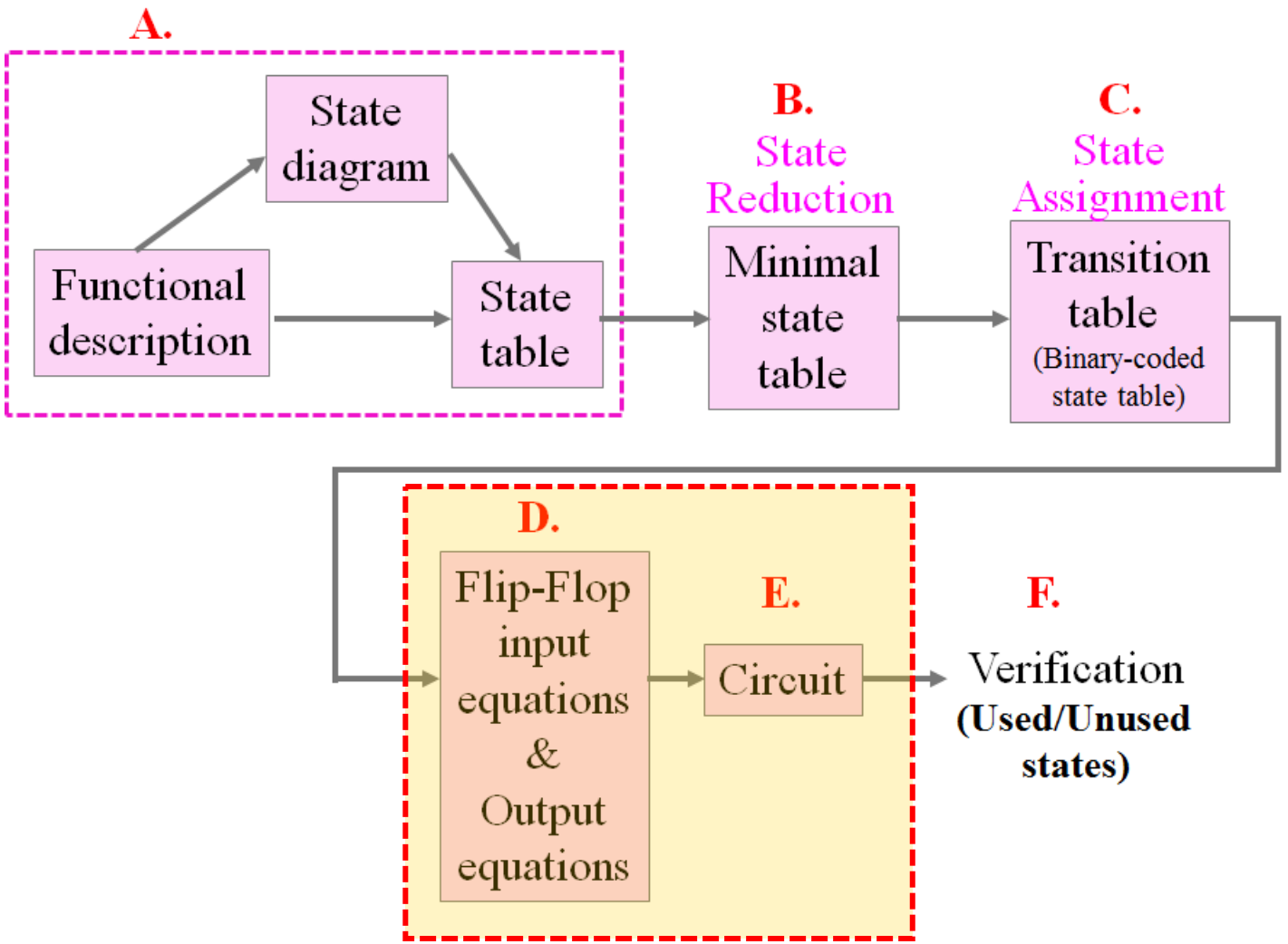
State Assignment (補充資料)

- When a seq ckt implemented w/ gate logic, # gates will depend on mapping b/t symbolic state names and binary encodings.

- Reference:
  - Randy H. Katz & Gaetano Borriello, *Contemporary Logic Design*, Prentice Hall.



D. Derive Flip-Flop Input Eqs & Output Eqs
E. Circuit



Flip-Flop Excitation Tables

- Select the flip-flop type or types

  $\Rightarrow$ Flip-flop excitation tables: (p.5-45)

(a) *JK* Flip-Flop

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

(b) *SR* Flip-Flop

$Q(t)$	$Q(t+1)$	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

(c) *D* Flip-Flop

$Q(t)$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

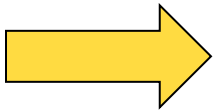
(d) *T* Flip-Flop

$Q(t)$	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table of D Flip-Flop

Characteristic table

D	Q^+
0	0
1	1



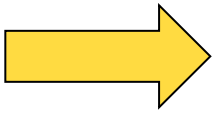
Excitation table

Q	Q^+	D
0	0	
0	1	
1	0	
1	1	

Excitation Table of JK Flip-Flop

Characteristic table

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Q′



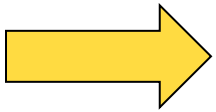
Excitation table

Q	Q^+	J	K	
0	0			⇒ 0 ×
0	1			⇒ 1 ×
1	0			⇒ × 1
1	1			⇒ × 0

Excitation Table of T Flip-Flop

Characteristic table

T	Q^+
0	Q
1	Q′



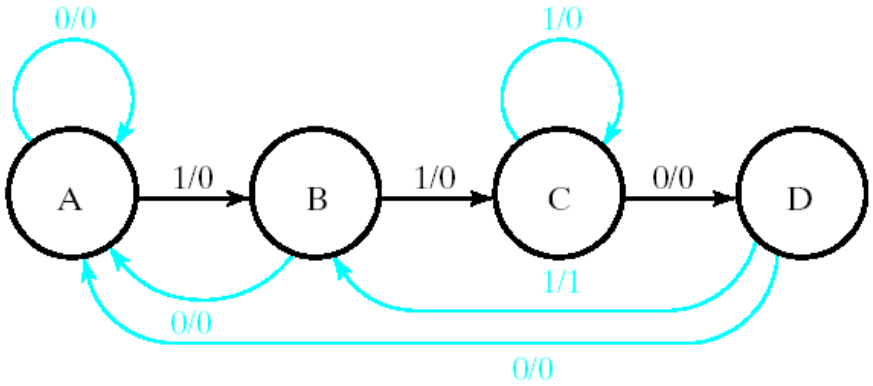
Excitation table

Q	Q^+	T
0	0	
0	1	
1	0	
1	1	

(a) Designing Using D Flip-Flops

- Example: Sequence detector (p.5-79)

State diagram:



State table:

Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

Present State	Next State		Output Z	
	X = 0	X = 1	X = 0	X = 1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

State assignment:

A = 00, B = 01,

C = 11, D = 10

Binary coded state table:

Present State	Next State		Output *Z*	
AB	X = 0	X = 1	X = 0	X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

Excitation table of D f-f

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

Present State AB	Next State A^+B^+ X = 0	Next State A^+B^+ X = 1	Output Z X = 0	Output Z X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

D f-f input values

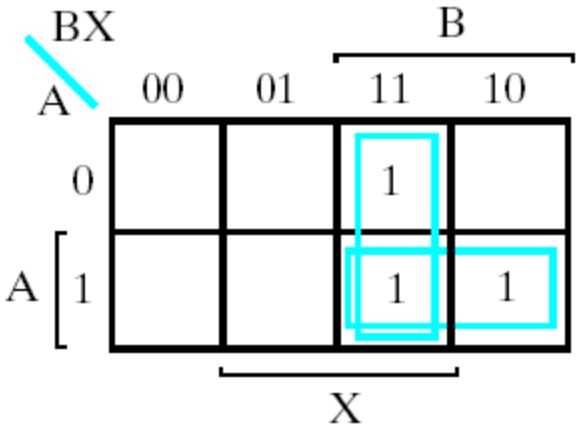
$A^+ \Rightarrow D_A$ $B^+ \Rightarrow D_B$ Z

F-F input eqs & output eq:

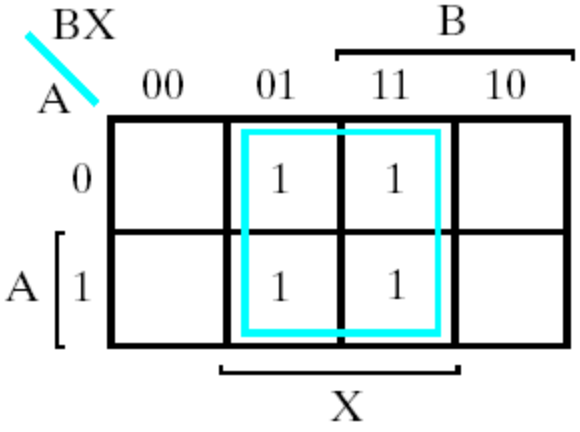
$$A(t+1) = D_A(A, B, X) = \Sigma m(3,6,7)$$

$$B(t+1) = D_B(A, B, X) = \Sigma m(1,3,5,7)$$

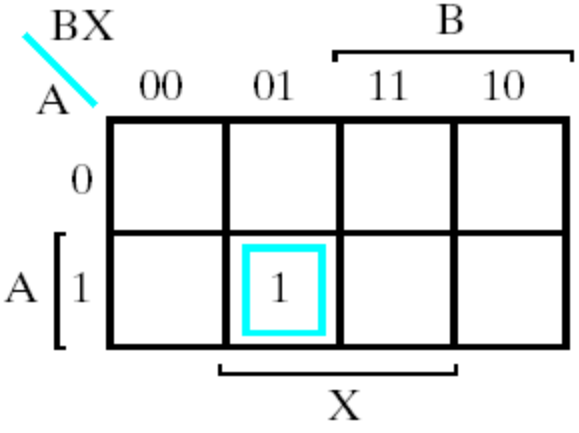
$$Z(A, B, X) = \Sigma m(5)$$



$D_A = AB + BX$



$D_B = X$



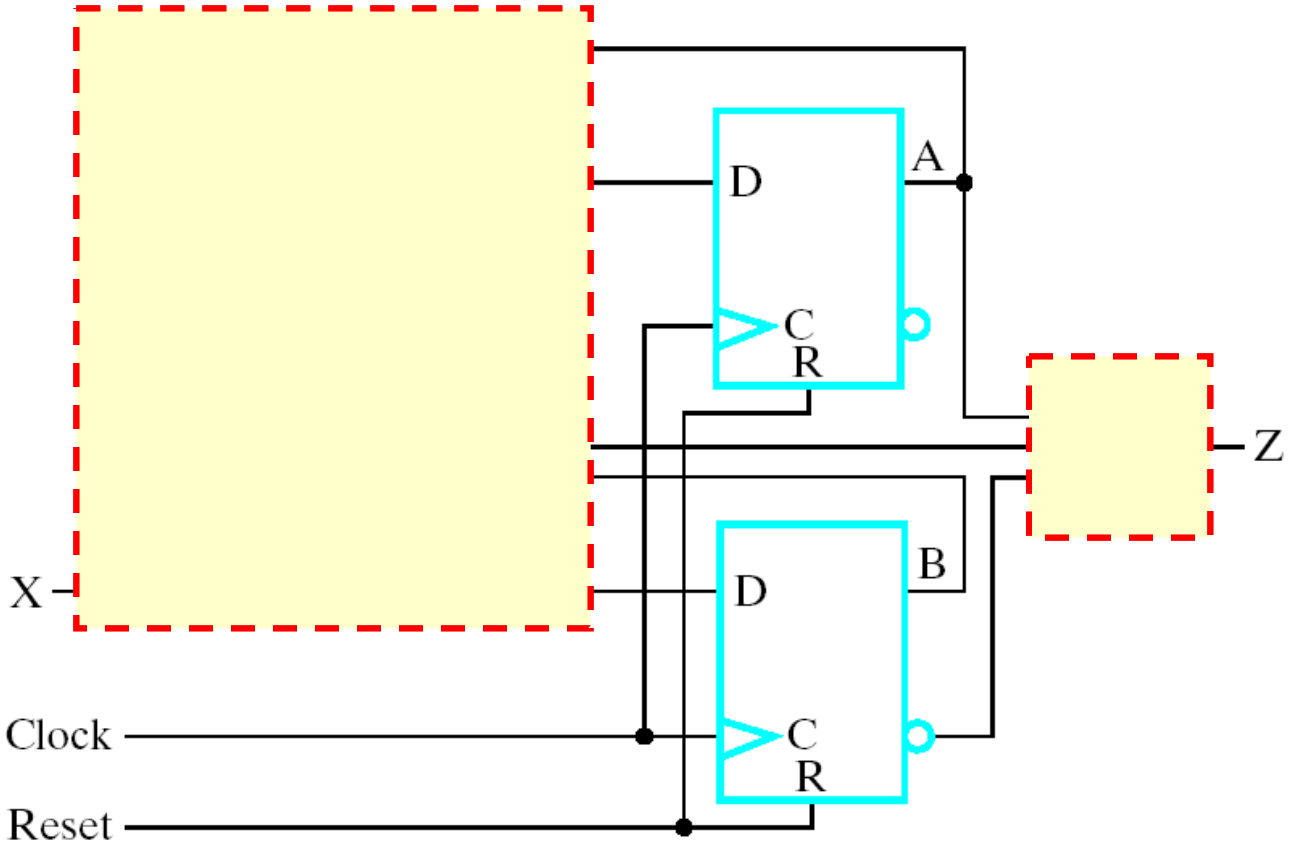
$Z = A\bar{B}X$

Logic diagram:

$$D_A = AB + BX$$

$$D_B = X$$

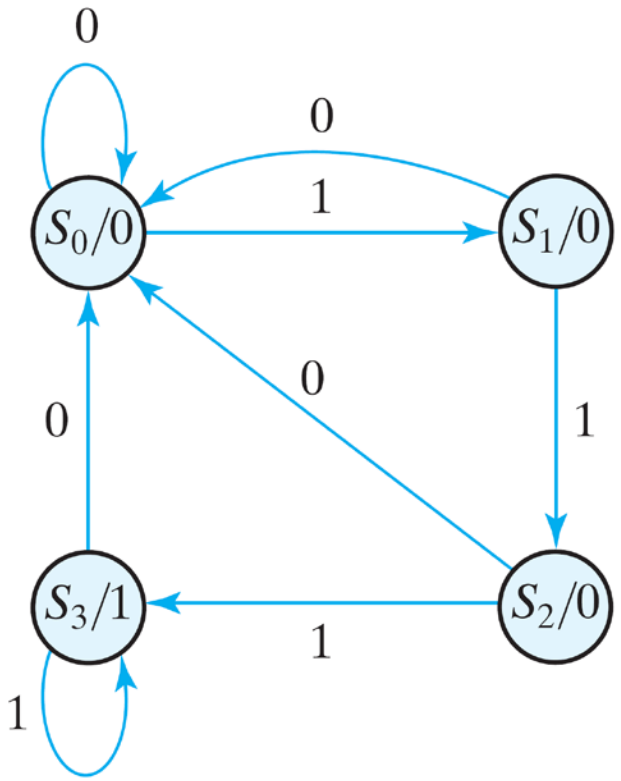
$$Z = A\overline{B}X$$



Example

- E.g.: Sequence detector

State diagram:



State assignment:

$S_0 = 00$, $S_1 = 01$,

$S_2 = 10$, $S_3 = 11$

Binary coded state table:

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

Excitation table of D f-f

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

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

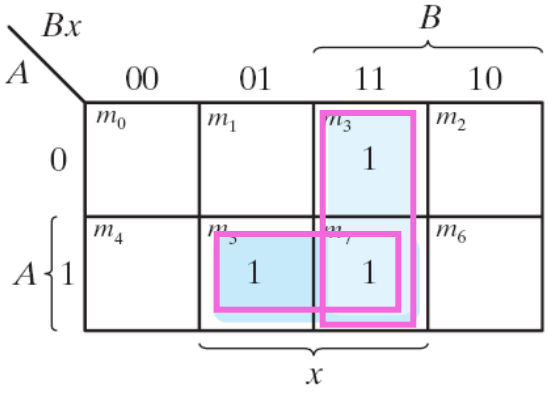
F-F input eqs & output eq:

$$A(t+1) = D_A(A, B, x) = \Sigma m(3,5,7)$$

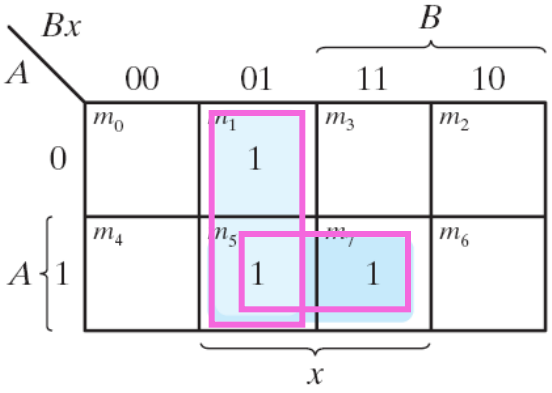
$$B(t+1) = D_B(A, B, x) = \Sigma m(1,5,7)$$

$$y(A, B, x) = \Sigma m(6,7)$$

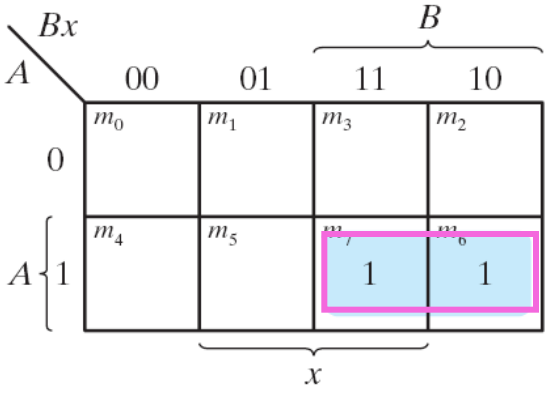
≡ D f-f input values



$D_A = Ax + Bx$



$D_B = Ax + B'x$



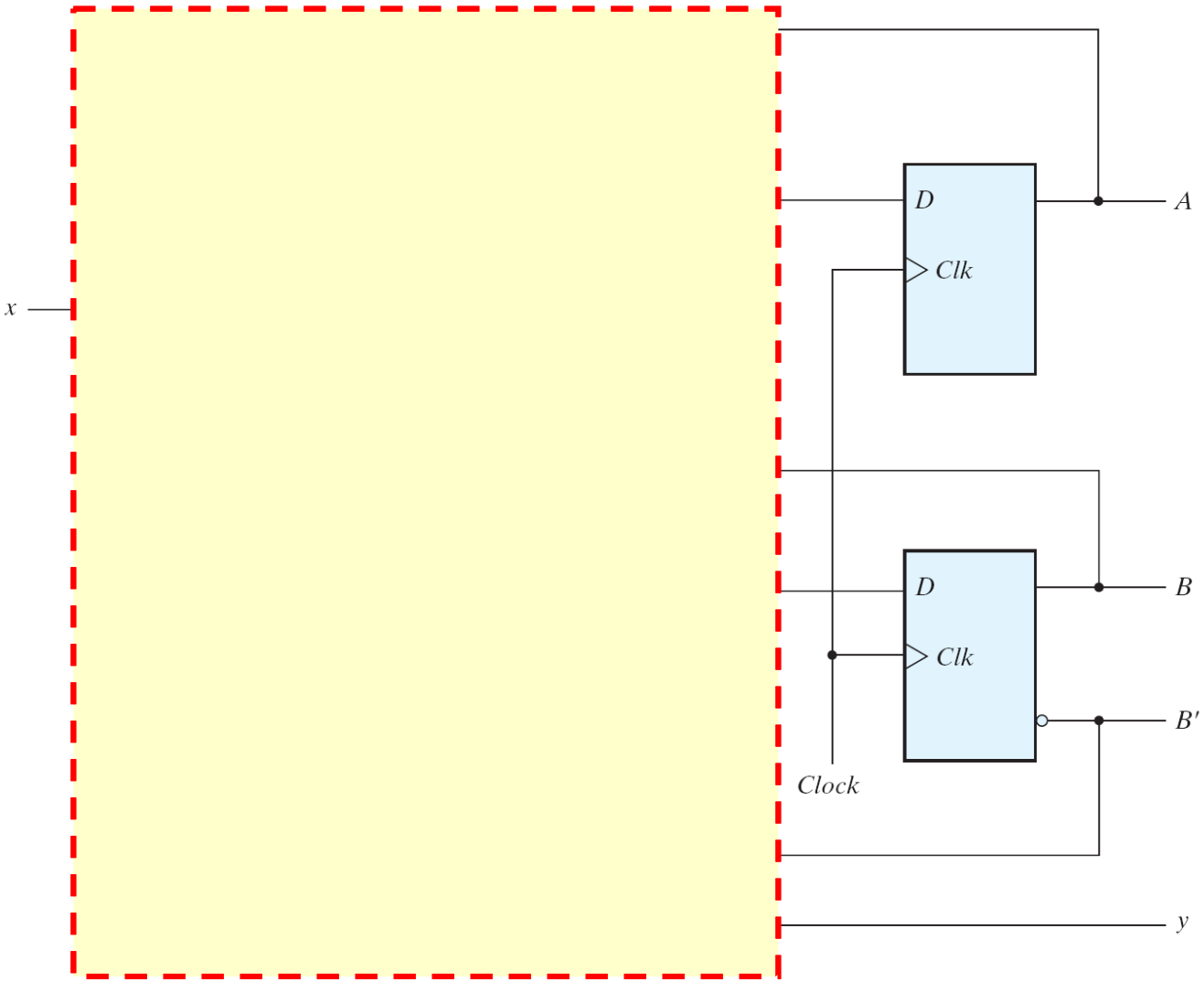
$y = AB$

Logic diagram:

$$D_A = Ax + Bx$$

$$D_B = Ax + B'x$$

$$y = AB$$



Designing w/ Unused States

- Example:
  - State table:

3 unused states

000

110

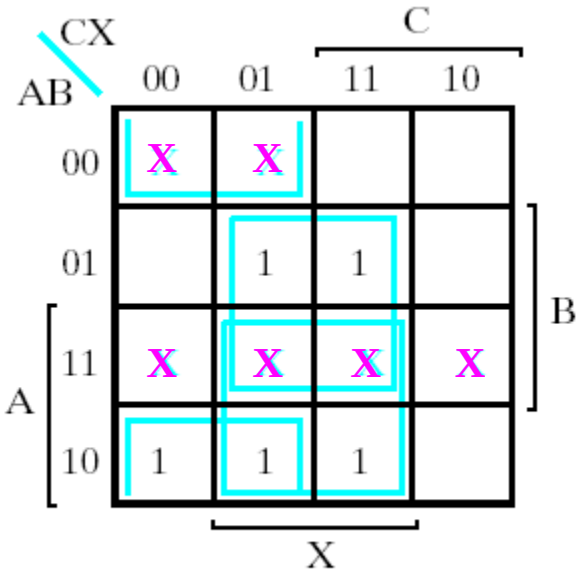
111

Present State			Input	Next State		
A	B	C	X	A^+	B^+	C^+
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	1	0	0
1	0	0	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	1	0	0

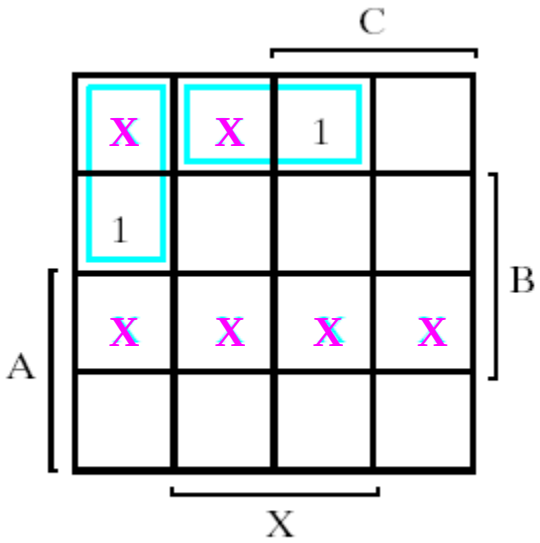
3 unused states: 000, 110,111

– Maps for optimizing flip-flop input equations:

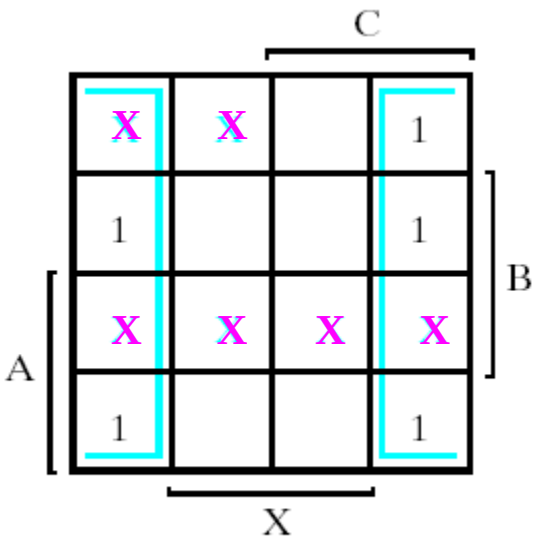
Present State			Input	Next State		
A	B	C	X	A^+	B^+	C^+
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	1	0	0
1	0	0	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	0	1	1	1	0	0



$D_A = AX + BX + \overline{B}\,\overline{C}$



$D_B = \overline{A}C\overline{X} + \overline{A}B\overline{X}$



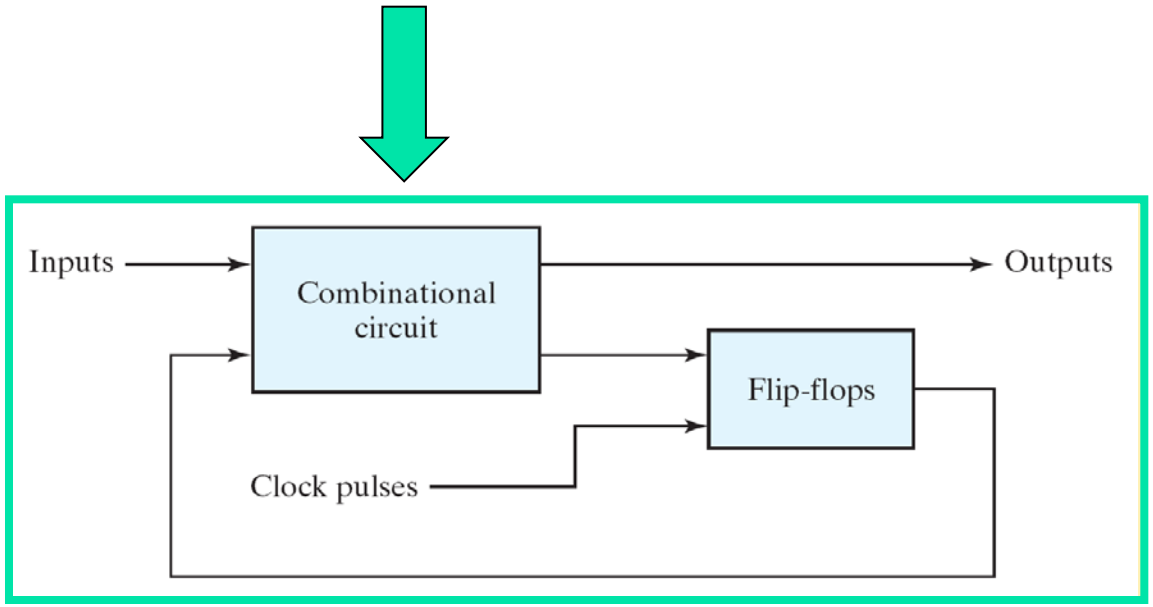
$D_C = \overline{X}$

– Logic diagram:

$$D_A = AX + BX + \overline{B}\,\overline{C}$$

$$D_B = \overline{A}\,\overline{C}\,\overline{X} + \overline{A}\,\overline{B}\,X$$

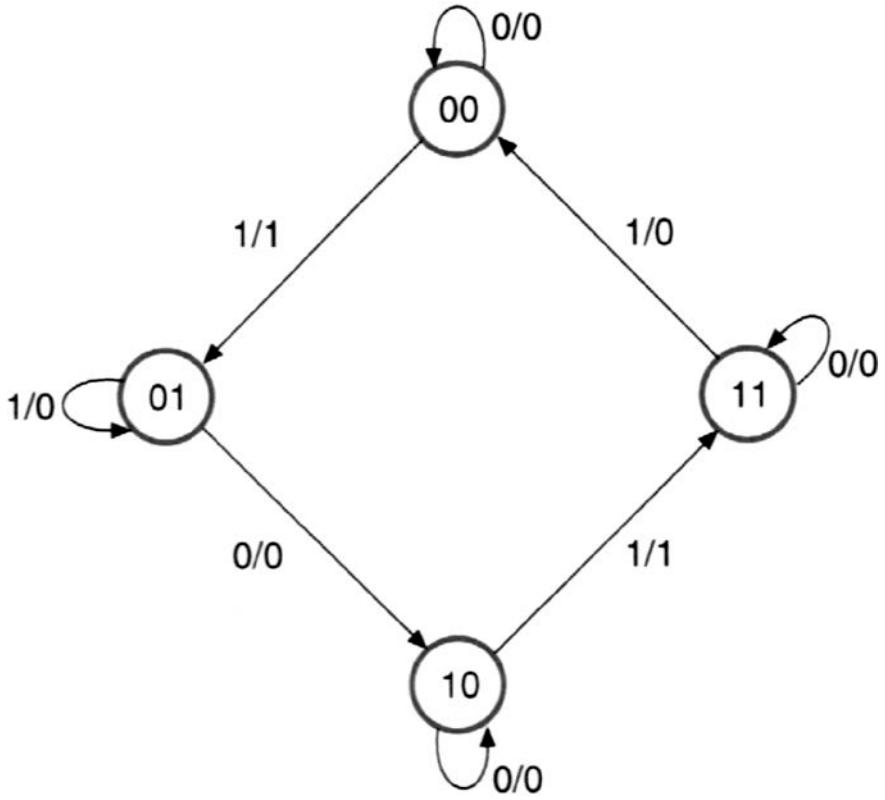
$$D_C = \overline{X}$$



(b) Synthesis Using JK Flip-Flops

- Example:

State diagram:



State table:

Present State		Input	Next State	
A	B	x	A^+	B^+
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

Excitation table of JK f-f

Q	Q^+	J	K
0	0		
0	1		
1	0		
1	1		

– State table and JK flip-flop inputs:

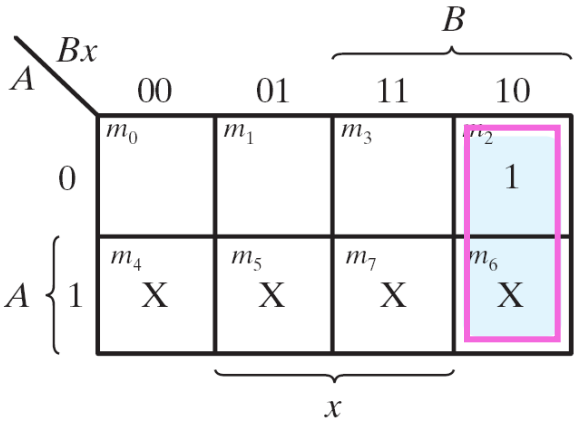
State Table and JK Flip-Flop Inputs

Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A^+	B^+	J_A	K_A	J_B	K_B
0	0	0	0	0	0	×		
0	0	1	0	1			1	×
0	1	0	1	0			×	1
0	1	1	0	1			×	0
1	0	0	1	0				
1	0	1	1	1				
1	1	0	1	1				
1	1	1	0	0				

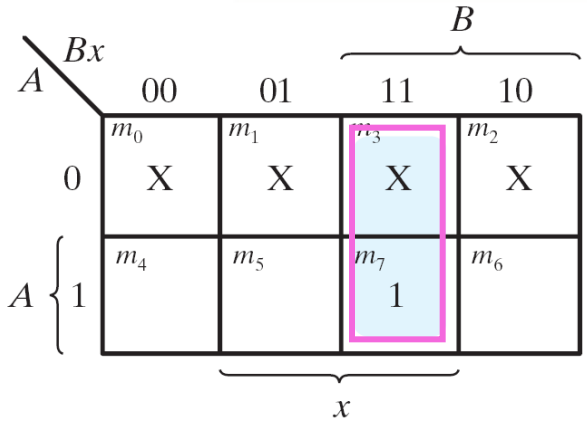
State Table and JK Flip-Flop Inputs

Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

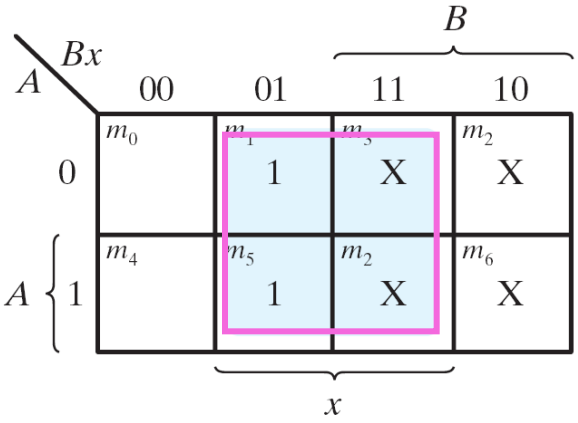
– Maps for flip-flop input equations:



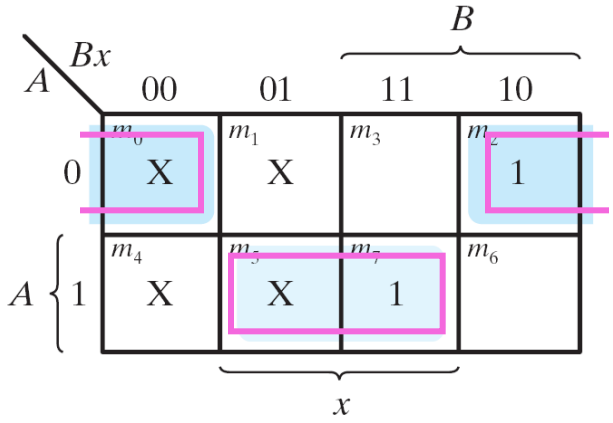
$J_A = Bx'$



$K_A = Bx$



$J_B = x$

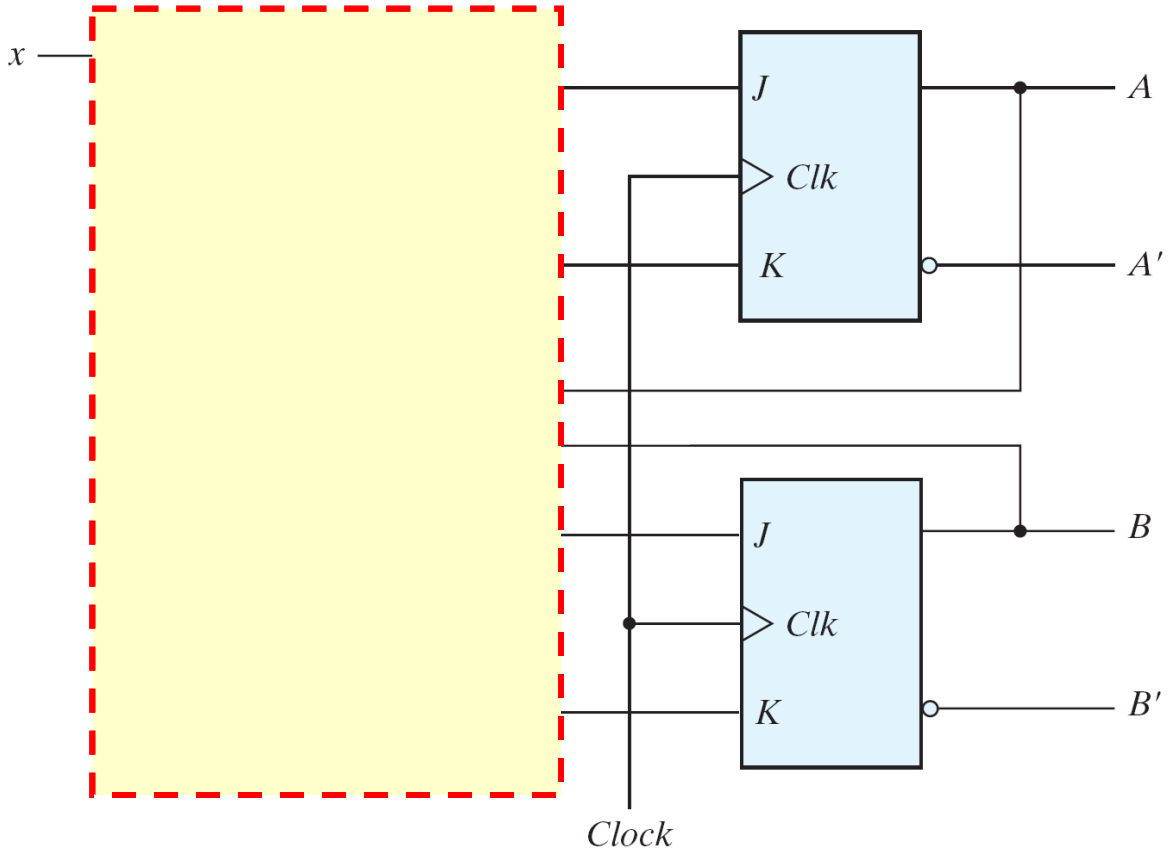


$K_B = (A \oplus x)'$

- Circuit:

$J_A = B\,x'$ $\quad K_A = B\,x$

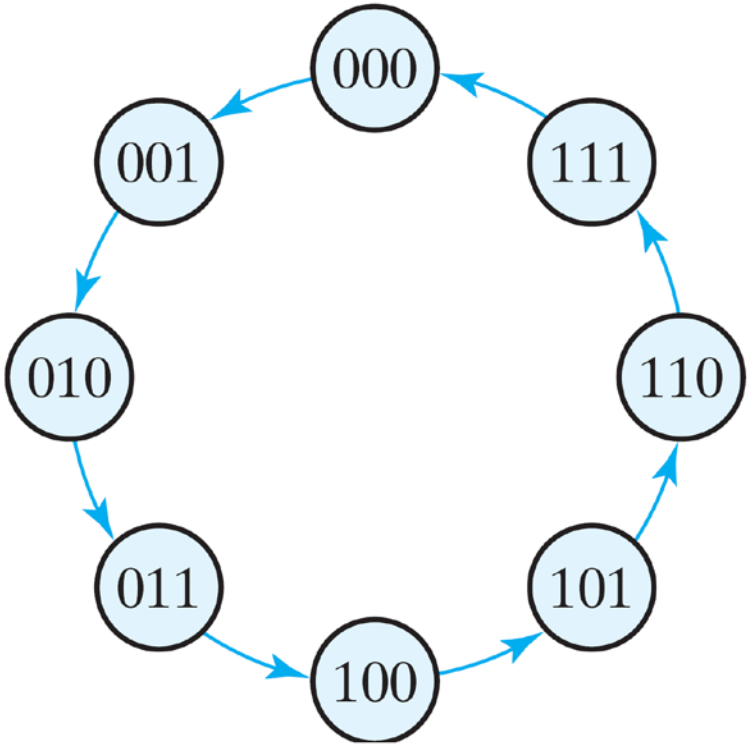
$J_B = x$ $\quad K_B = Ax + A'\,x' = (A \oplus x)'$



(c) Synthesis Using T Flip-Flops

- Example: 3-bit binary counter

State diagram:



State table:

Present State			Next State		
A_2	A_1	A_0	A_2	A_1	A_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

Excitation table of T f-f

Q	Q^+	T
0	0	
0	1	
1	0	
1	1	

– State table and T flip-flop inputs:

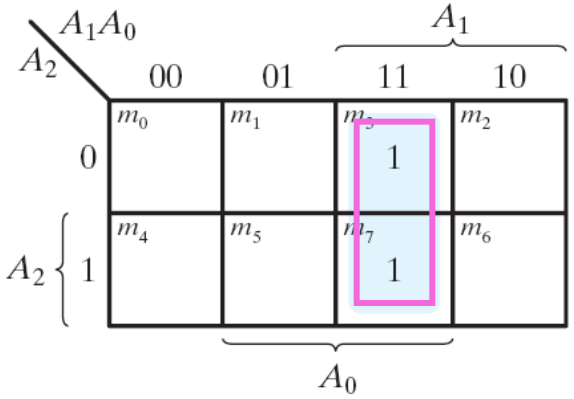
State Table for Three-Bit Counter

Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0		1
0	0	1	0	1	0			1
0	1	0	0	1	1		0	
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			

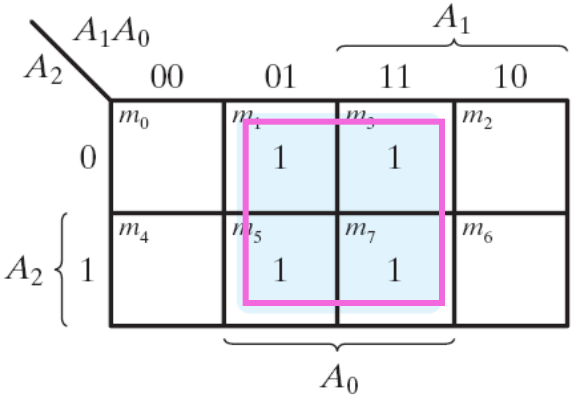
State Table for Three-Bit Counter

Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	**0**	1
1	1	1	0	0	0	1	1	1

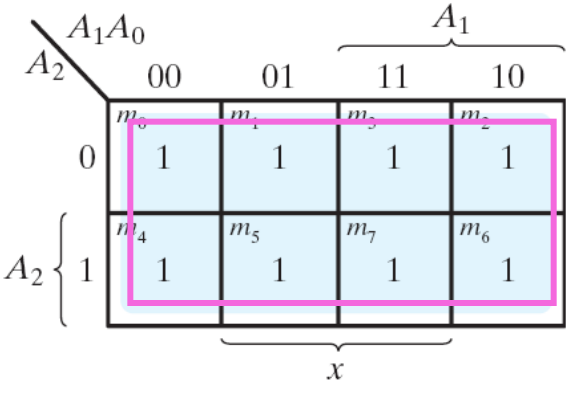
- Maps for flip-flop input equations:



$T_{A2} = A_1A_0$

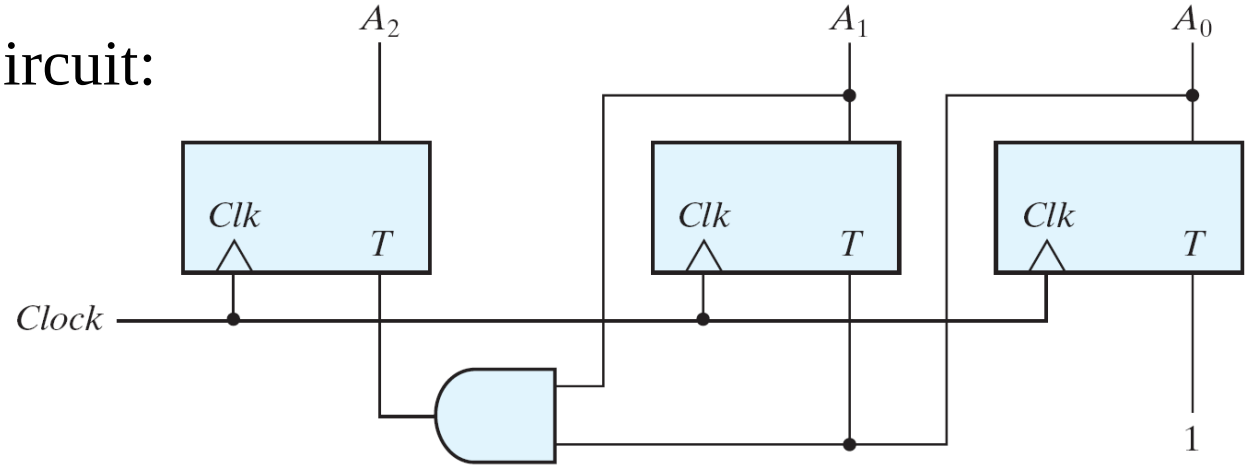


$T_{A1} = A_0$

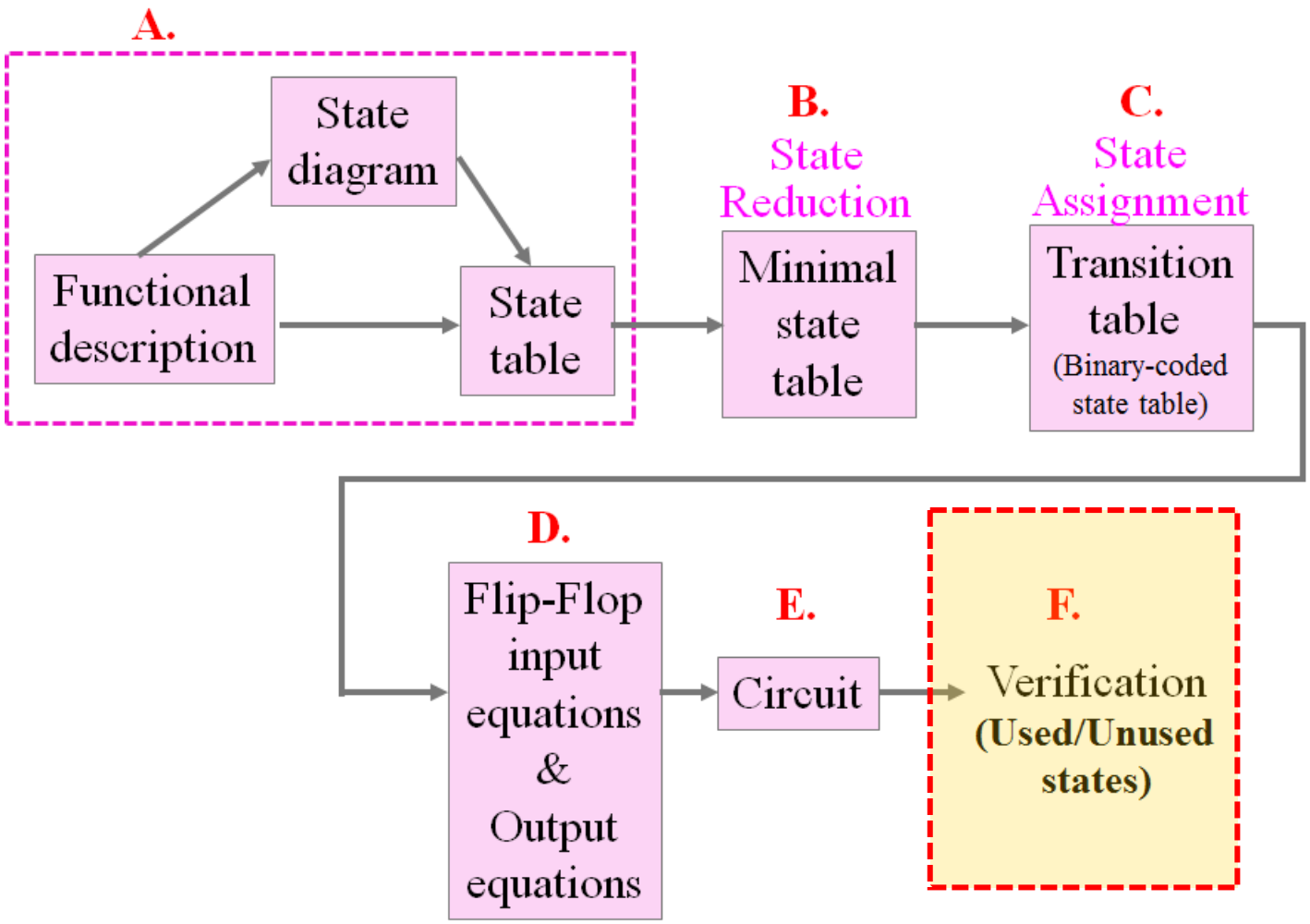


$T_{A0} = 1$

- Circuit:



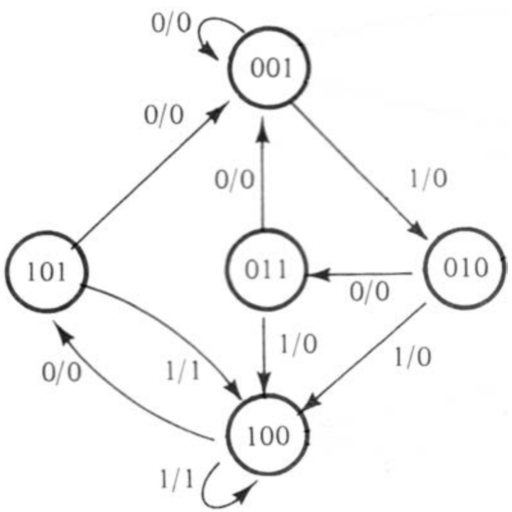
F. Verification



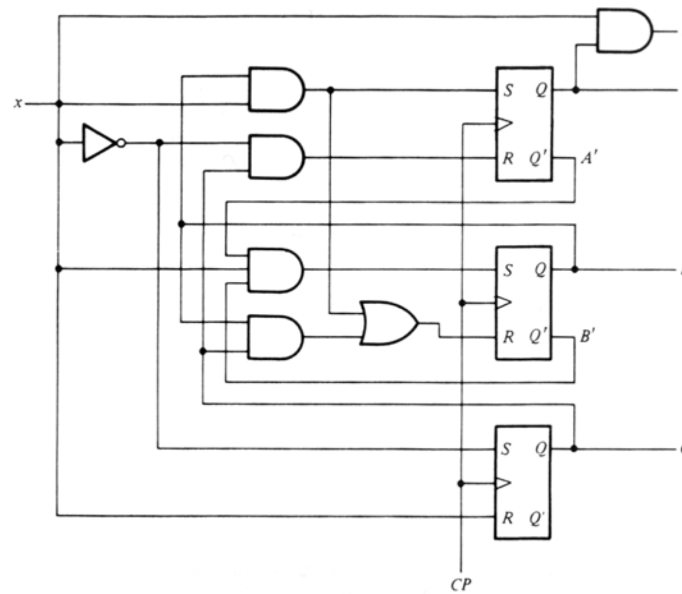
- Verification of sync seq ckts:
  - show that the ckt produces the original state diagram or state table
    - used states
    - unused states
  - Manual verification: for small ckts
    - analyze the ckt
  - Verification w/ simulation
    - requires a sequence of input combinations and applied clocks

Example: Manual Verification

- E.g.:

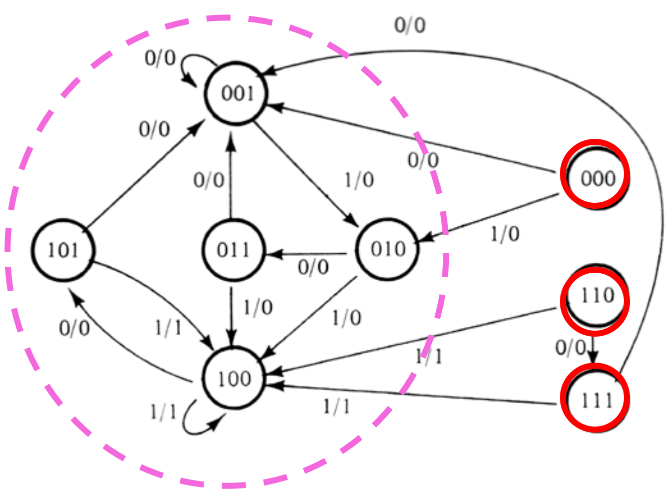


Synthesis
(A ~ F)



Verify
Analysis
(G)

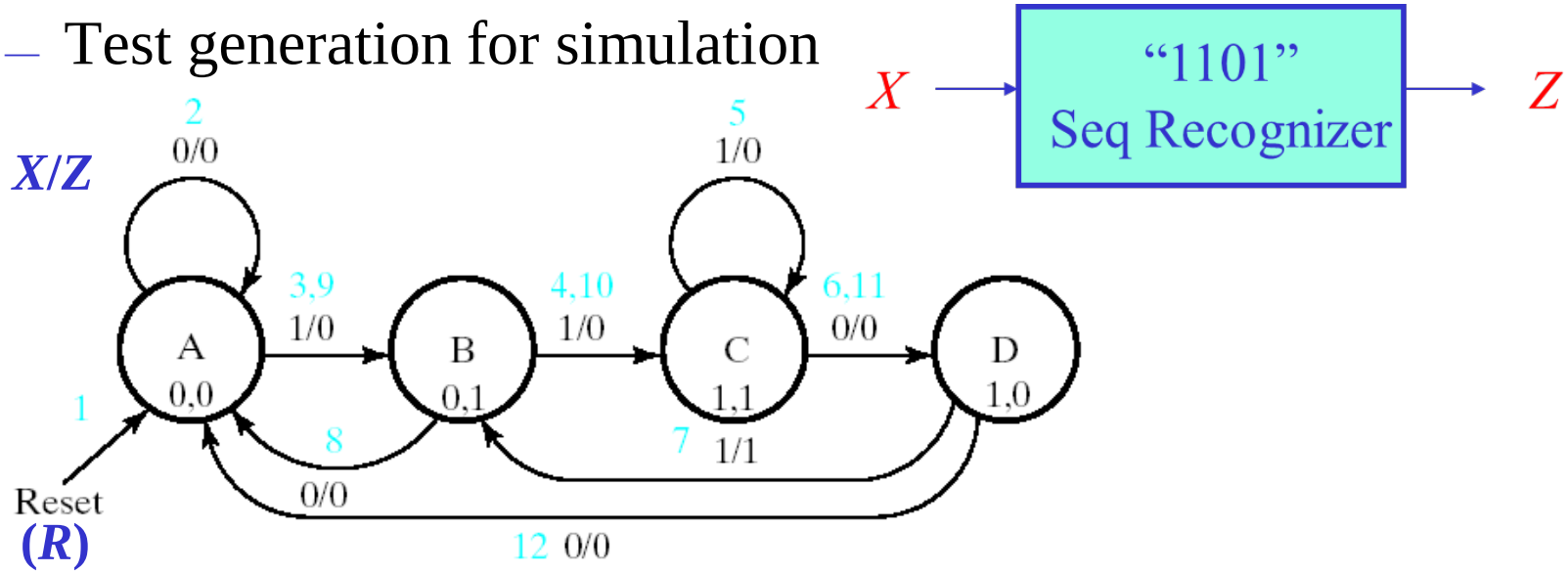
Used
state



Unused
state

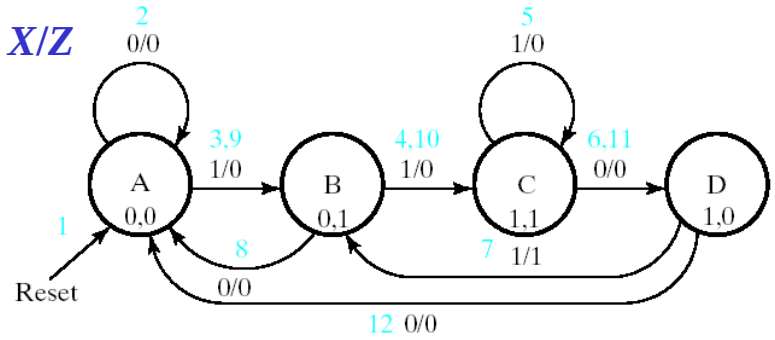
Example: Verification w/ simulation

- E.g.: Verifying the sequence recognizer
  - Test generation for simulation

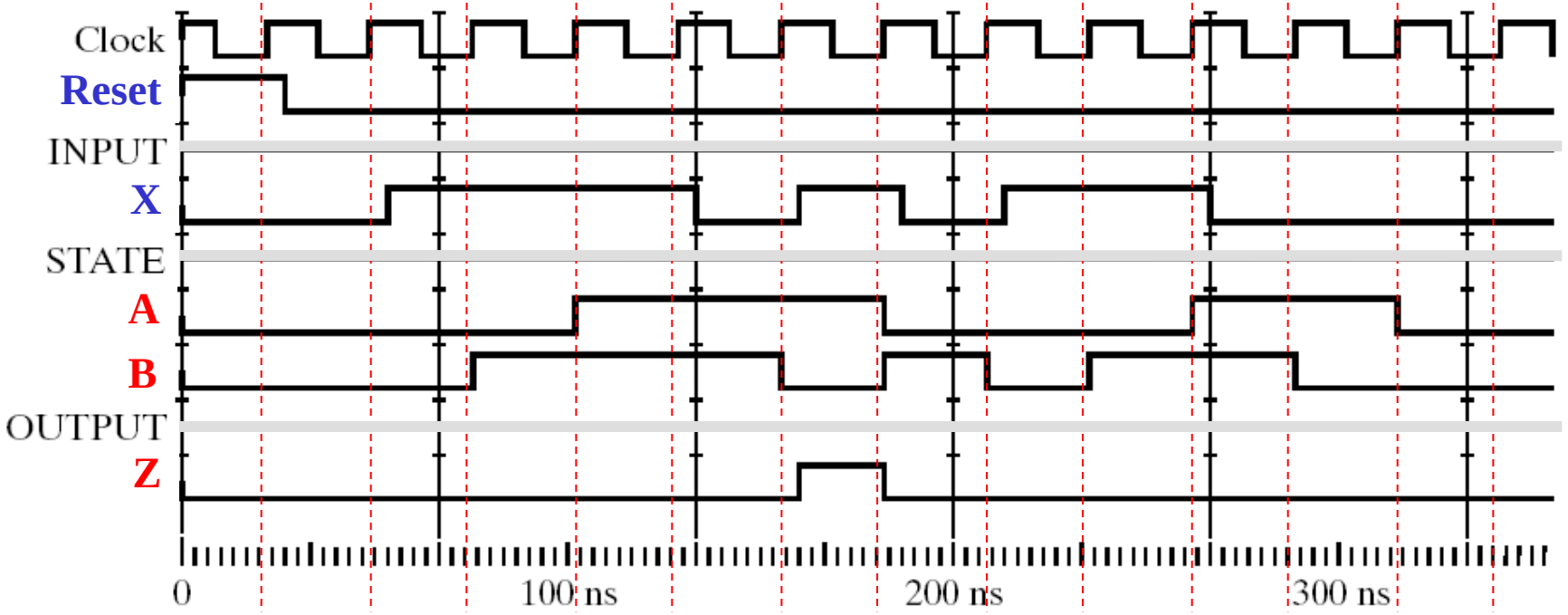


Clock Edge:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Input R:	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	
Input X:	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	
State (A,B):	×,×													
Output Z:	×													

– Simulation



Clock Edge:		0	1	2	3	4	5	6	7	8	9	10	11	12	13
Input R:	X	1	0	0	0	0	0	0	0	0	0	0	0	0	
Input X:	X	0	0	1	1	1	0	1	0	1	1	0	0	0	
State (A,B):		X,X	0,0*	0,0	**0,1**	**1,1**	1,1	**1,0**	**0,1**	**0,0**	**0,1**	**1,1**	**1,0**	**0,0**	0,0
Output Z:	X	0	0	0	0	0	0	1	0	0	0	0	0	0	



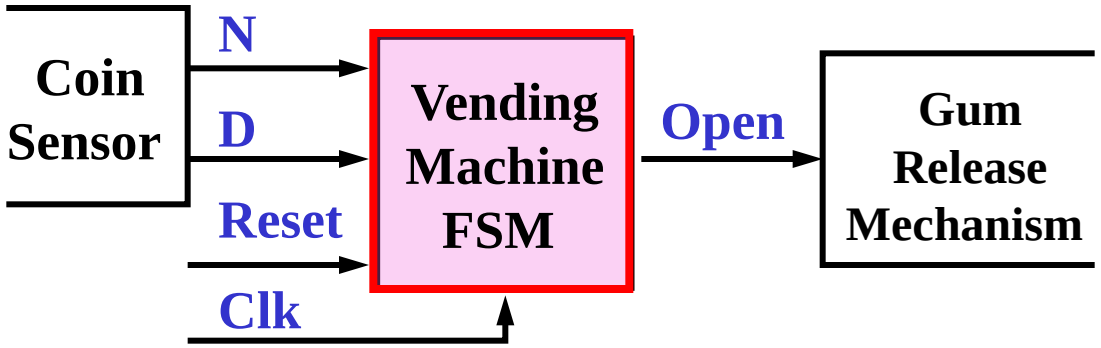
補充資料： Word Problems

Example: Vending Machine

Deliver a package of gum after 15 cents received in coins.
Has a single coin slot for dimes and nickels, one coin at a time.
No change.

<Ans.>

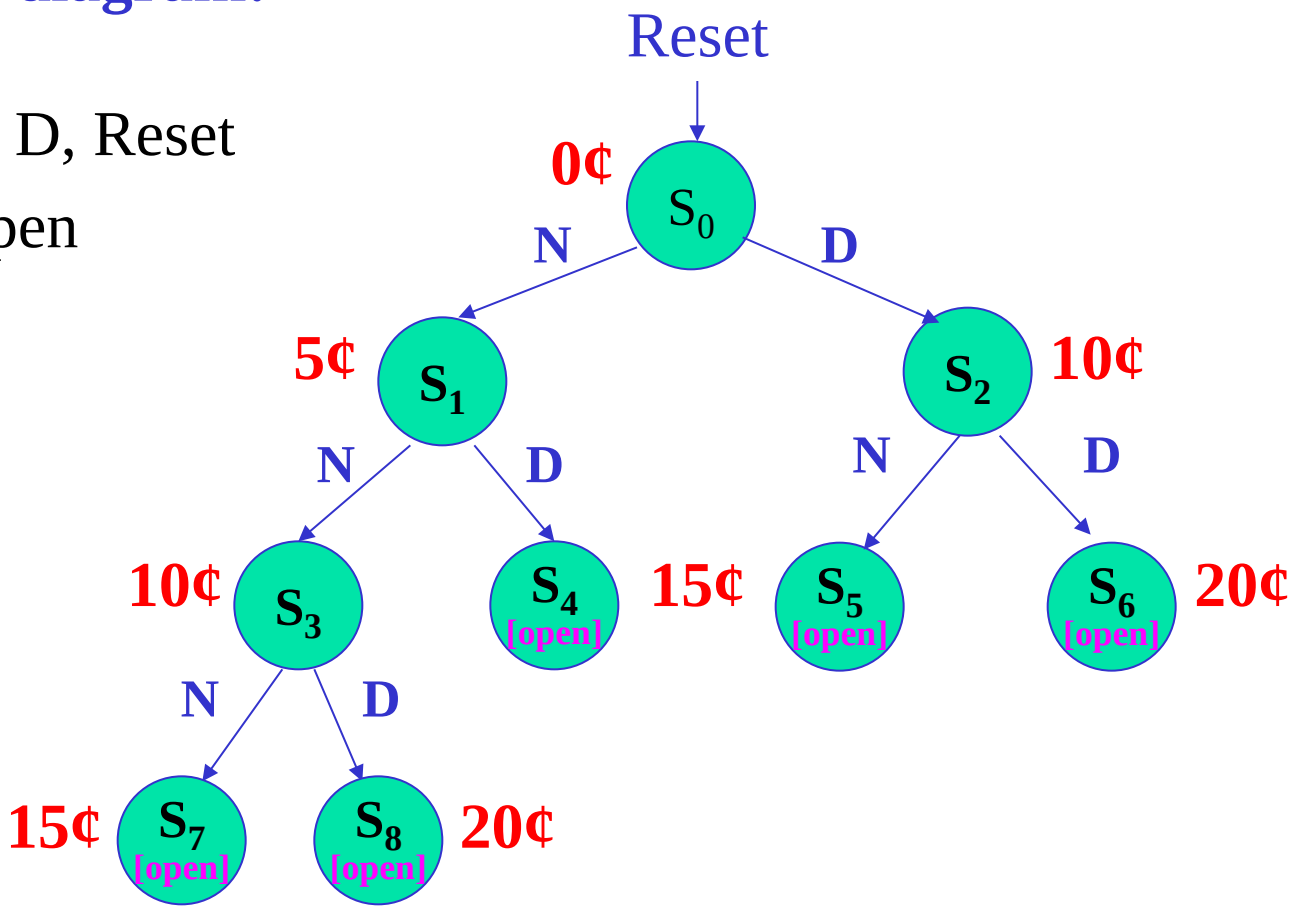
Understand the problem:



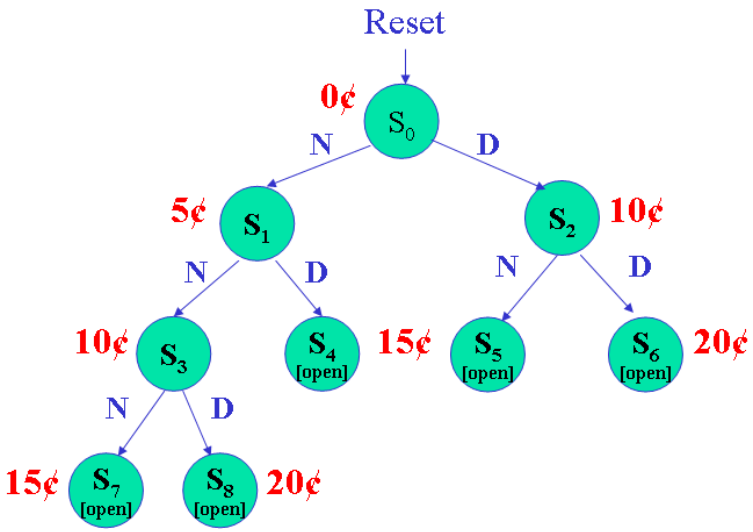
Draw state diagram:

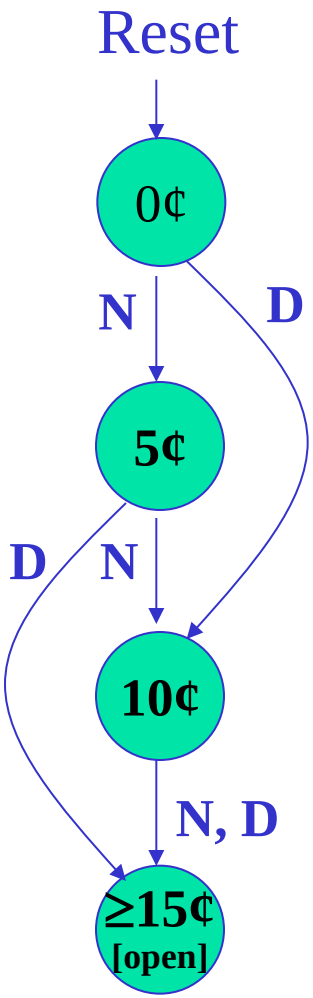
Inputs: N, D, Reset

Output: open



State Minimization:



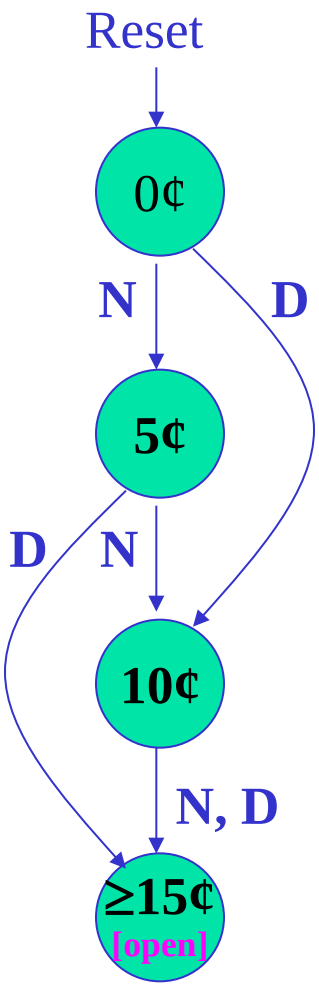


9 states

$\Rightarrow$ 5 states: 0¢, 5¢, 10¢, 15¢, 20¢

$\Rightarrow$ 4 states: 0¢, 5¢, 10¢, ≥15¢

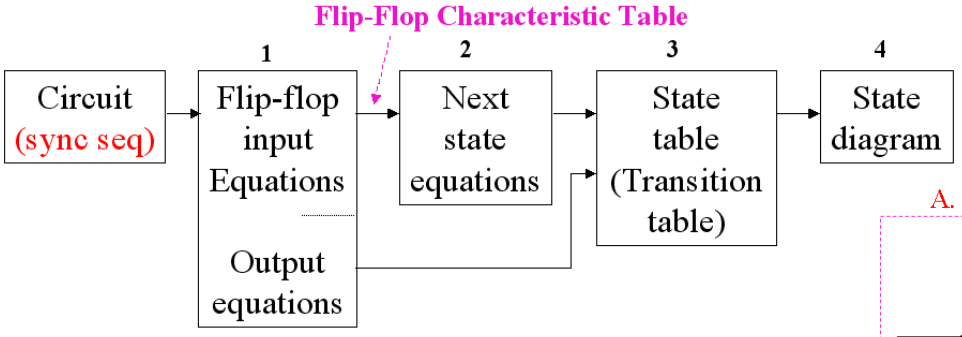
State Table:



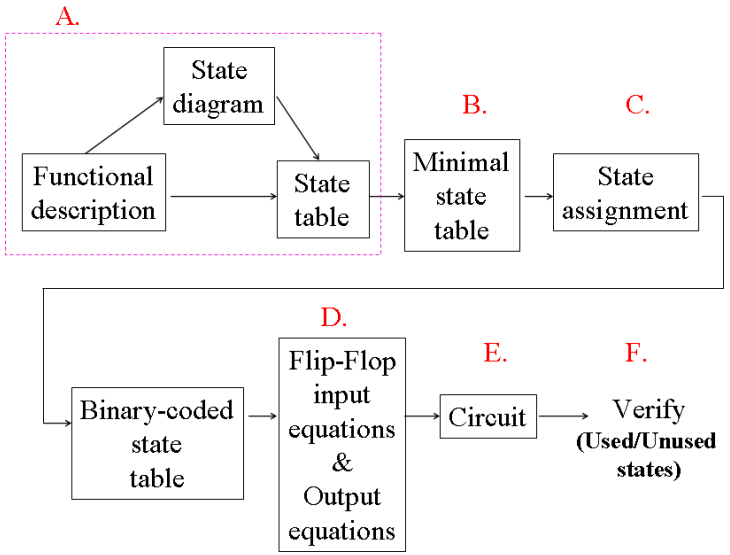
Present State	Input D	Input N	Next State	Output Open
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	×	×
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	≥15¢	0
	1	1	×	×
10¢	0	0	10¢	0
	0	1	≥15¢	0
	1	0	≥15¢	0
	1	1	×	×
≥15¢	×	×	≥15¢	1

Chapter Summary

- Sequential Circuits
- Latches
- Flip-Flops: master-slave, edge-triggered
- Analysis of Clocked Seq Ckts



- Design of Clocked Seq Ckts
  - State Reduction
  - State Assignment



Problems & Homework (6th ed)

Sections	Exercises	Homework
§5-3	5.1	5.1(a)
§5-4	5.2~5.5	5.4
§5-5	5.6~5.10	5.6
§5-7	5.11~5.14	5.11 (by implication chart method)
§5-8	5.15~5.20	5.18*, 5.19(a)*
HDL	5.21~5.60	