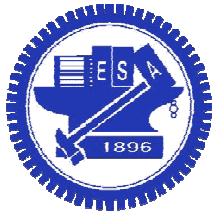
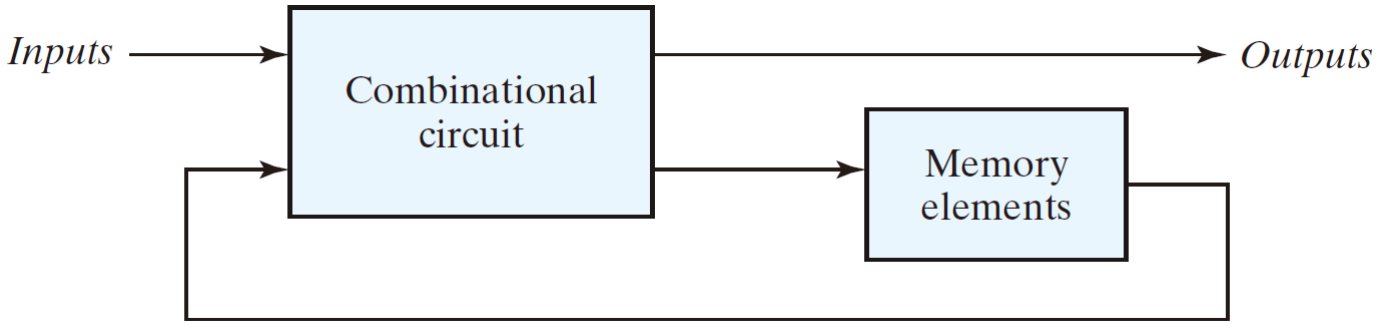
# Synchronous Sequential Circuits




Chun-Jen Tsai
National Chiao Tung University
11/01/2012

# Sequential Circuits

- A sequential circuit has the following features:
  - A feedback path
  - Memory of the state of the sequential circuit
  - The logic function maps (inputs, current state) to (outputs, next state)
- There are two types of signal changes
  - Synchronous: transition happens at discrete instants of time
  - Asynchronous: at any instant of time

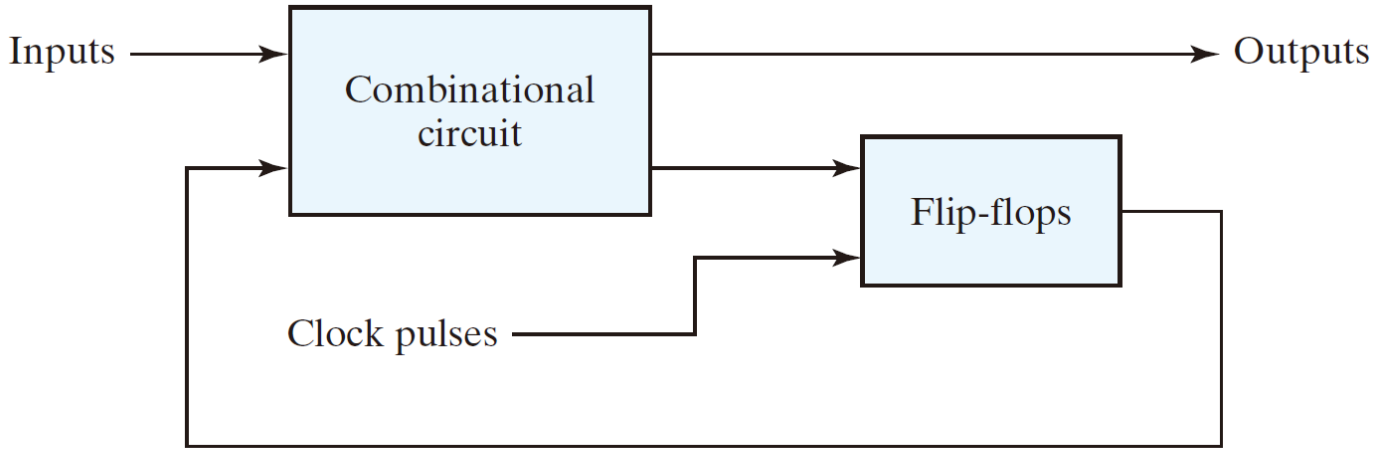


# Synchronous Sequential Circuits

- A synchronous sequential circuits use a master-clock generator to generate a periodic train of clock pulses
  - The clock pulses are distributed throughout the system; signals are only “used” when timing is right
- A memory element in a digital circuit keeps a binary state indefinitely, until it is directed to switch state
- There two types of memories: latches and flip-flops
  - Latches switches state according to input level; not practical for synchronous sequential circuits
  - Flip-flops are constructed using latches; and they switch states only at signal edges → make precise signal synchronization possible

# Clocked Sequential Circuit

- A synchronous clocked sequential circuit has the following basic architecture:

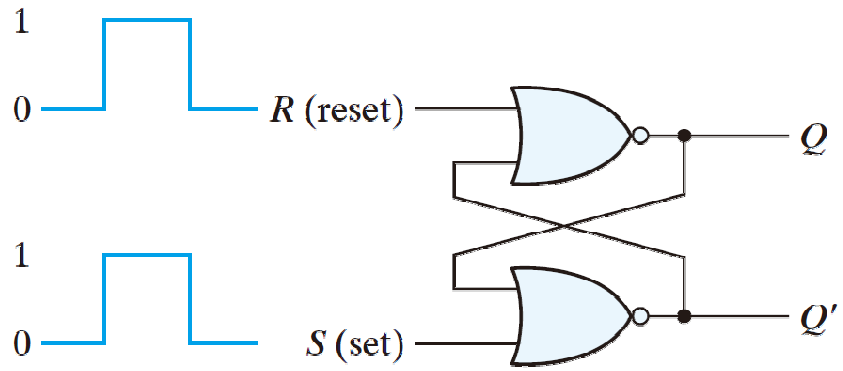



(a) Block diagram

(b) Timing diagram of clock pulses

# SR Latches (1/2)

- SR Latches built with two NOR gates:




(a) Logic diagram

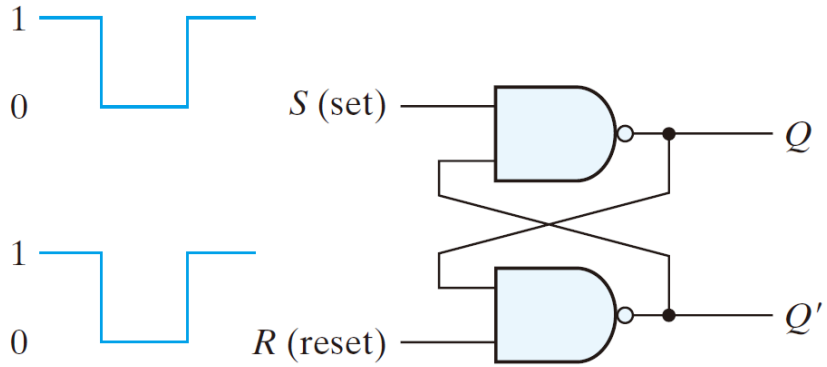
| $S$ | $R$ | $Q$ | $Q'$ | |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 0 | (after $S = 1, R = 0$) |
| 0 | 1 | 0 | 1 | |
| 0 | 0 | 0 | 1 | (after $S = 0, R = 1$) |
| 1 | 1 | 0 | 0 | (forbidden) |

(b) Function table

  - Under normal conditions, both inputs remain at 0 unless the state has to be changed, as follows:
    - $(S, R) = (0, 1)$: reset ($Q = 0$, the clear state)
    - $(S, R) = (1, 0)$: set ($Q = 1$, the set state)
  - The forbidden case put the latch into unpredictable states

# SR Latches (2/2)

- SR latches can also be built with two NAND gates:
  - Also called the S′R′ (or $\overline{S}\overline{R}$) latches

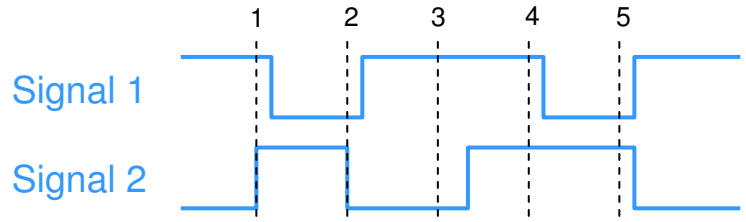



(a) Logic diagram

| $S$ | $R$ | $Q$ | $Q'$ | |
|---|---|---|---|---|
| 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) |

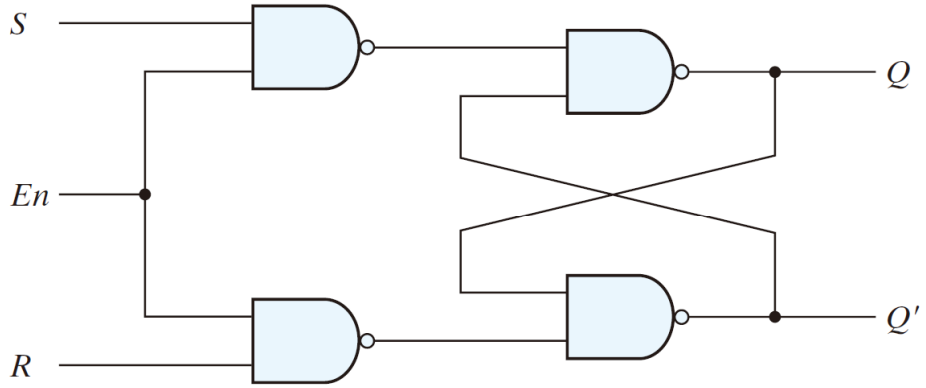
(b) Function table

- Note: variations in gate delays may cause jitters in digital signals



# SR Latches with Control Input

- SR latch with control input:
  - The *En* signal controls whether a state change can happen, regardless of what the levels of *S* and *R* are




(a) Logic diagram

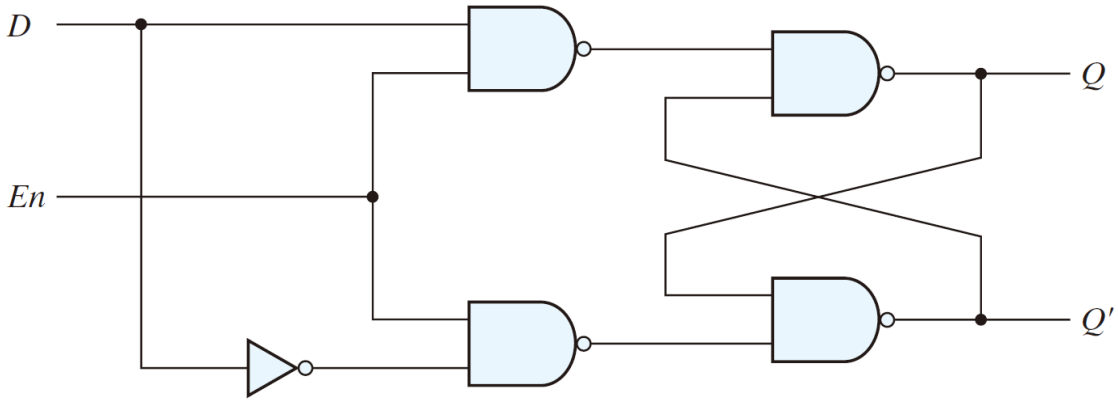
| *En* | *S* | *R* | Next state of *Q* |
|---|---|---|---|
| 0 | X | X | No change |
| 1 | 0 | 0 | No change |
| 1 | 0 | 1 | $Q = 0$; reset state |
| 1 | 1 | 0 | $Q = 1$; set state |
| 1 | 1 | 1 | Indeterminate |

(b) Function table

- In reality, an SR latch is difficult to use due to the existence of an indeterminate state

# D Latch (Transparent Latch)

- To eliminate the undesirable condition of the indeterminate state in the SR latch, we can use a single input to set the state:




(a) Logic diagram

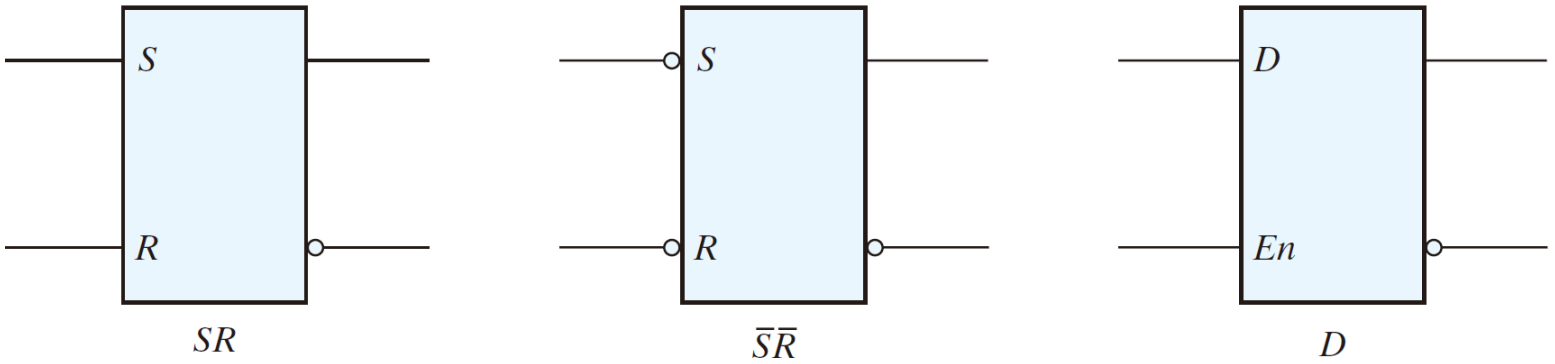
| *En* | *D* | Next state of *Q* |
|---|---|---|
| 0 | X | No change |
| 1 | 0 | $Q = 0$; reset state |
| 1 | 1 | $Q = 1$; set state |

(b) Function table

- When the *En* signal is active, the data line *D* is passed to the output transparently

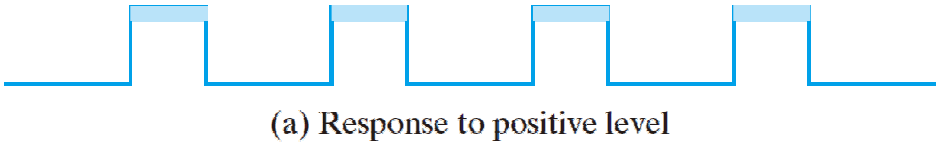
# Graphic Symbols for Latches

- The graphic symbols for latches are as follows:



# Flip-Flops

- For digital design, the state of a memory device only changes when a control signal triggers the switch
  - Level triggered – latches
  - Edge triggered – flip-flops



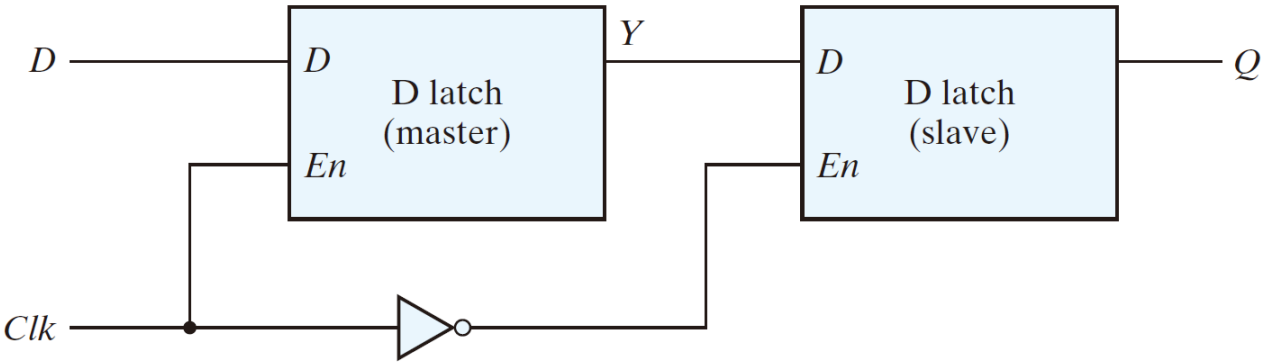
(a) Response to positive level

(b) Positive-edge response

(c) Negative-edge response

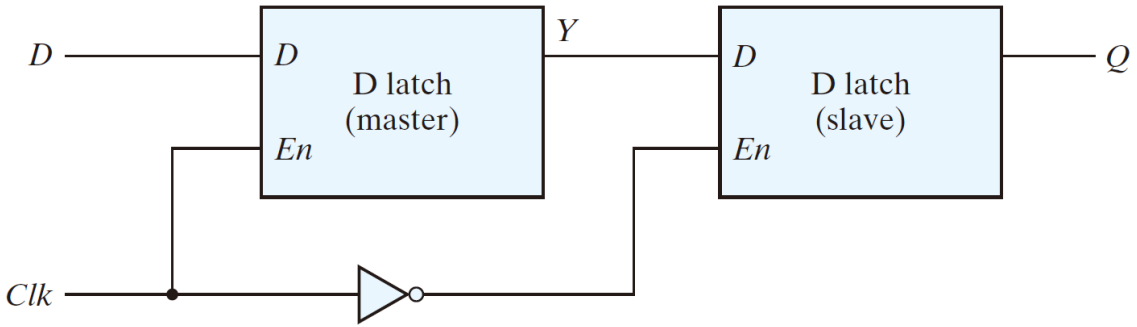
# Level Trigger vs. Edge Trigger

- If level-triggered flip-flops are used
  - the feedback path may cause instability problem
- Edge-triggered flip-flops
  - the state transition happens only at the edge
  - eliminate the multiple-transition problem



# Edge-triggered D-type Flip-Flop

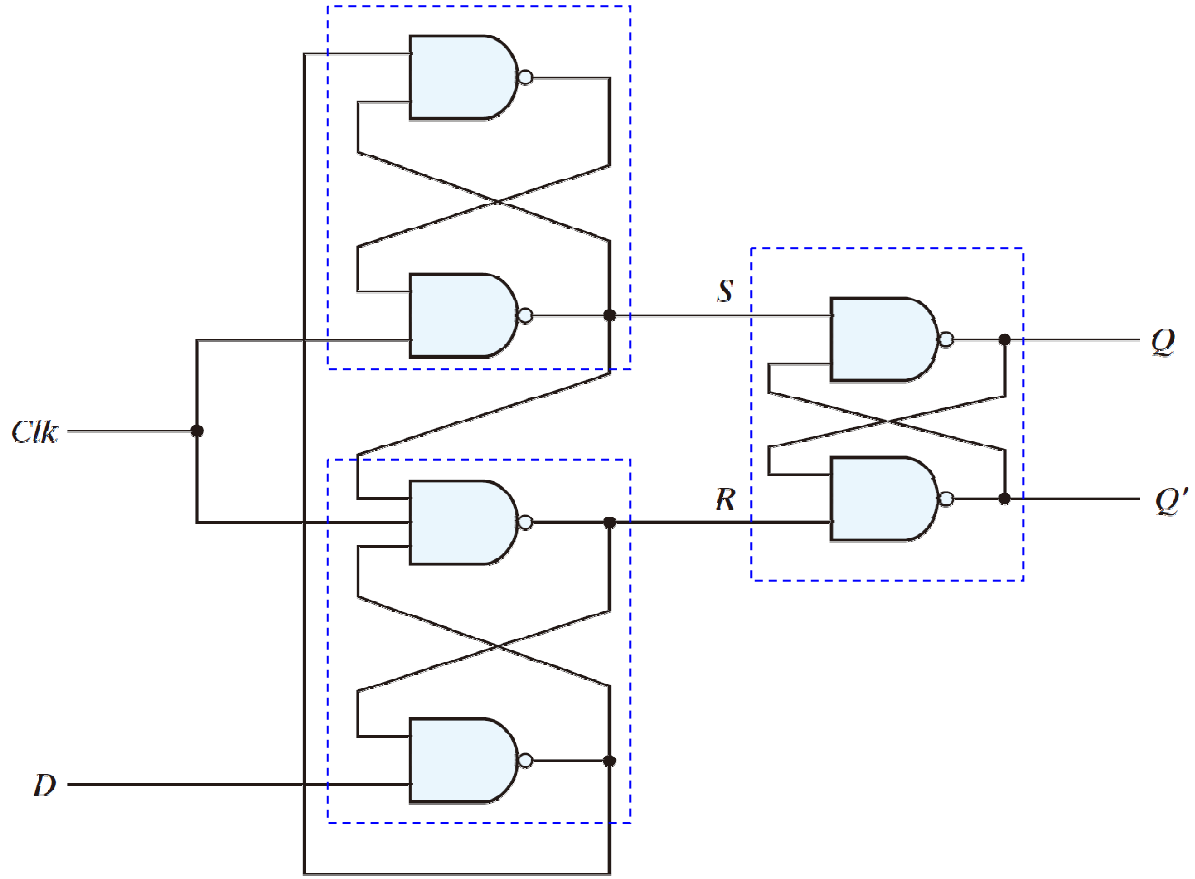
- A D-type flip-flop is constructed with two D latches arranged in master-slave configuration
  - A positive edge-triggered D flip-flop changes $Q$ at the negative edge $(1 \rightarrow 0)$ of the $Clk$ signal
    - The master latch samples $D$ at positive-level of $Clk$
    - The slave latch samples $Y$ at negative-level of $Clk$




  - How to design a D flip-flip triggered at the negative edge?

# Alternative Design of D Flip-flop

- A D-type positive-edge-triggered flip-flop can also be implemented using three SR latches:

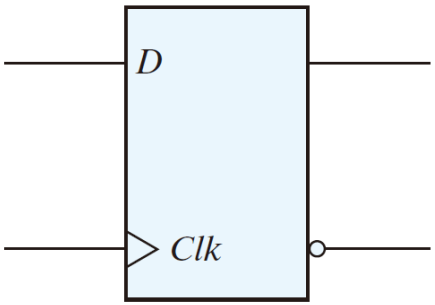


# D Flip-Flop Properties

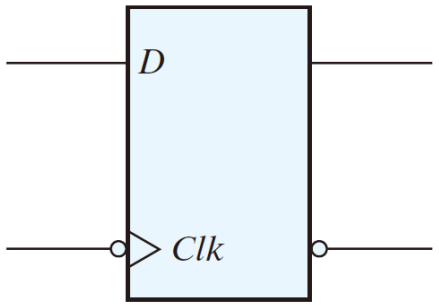
- There are several important parameters of a flip-flip:
  - The *setup time* is the time period the $D$ input must be maintained at a constant value prior to the occurrence of the clock transition
  - The *hold time* is the time period the $D$ input must not change after the trigger edge of the clock
  - The *propagation delay time* is defined as the interval between the trigger edge and the stabilization of the output to a new state
- In a synchronous digital system, all flip-flops belongs to the same clock domain must make their state transition at the same time

# Graphic Symbols for D Flip-Flop

- For a D flip-flop, the characteristic equation is:
  - $Q(t+1) = D$
- There are two symbols for D flip-flop, one for positive-edge trigger and one for negative edge trigger
  - The arrowhead symbol designates a dynamic input that triggers the flip-flop at edge transisitons




(a) Positive-edge



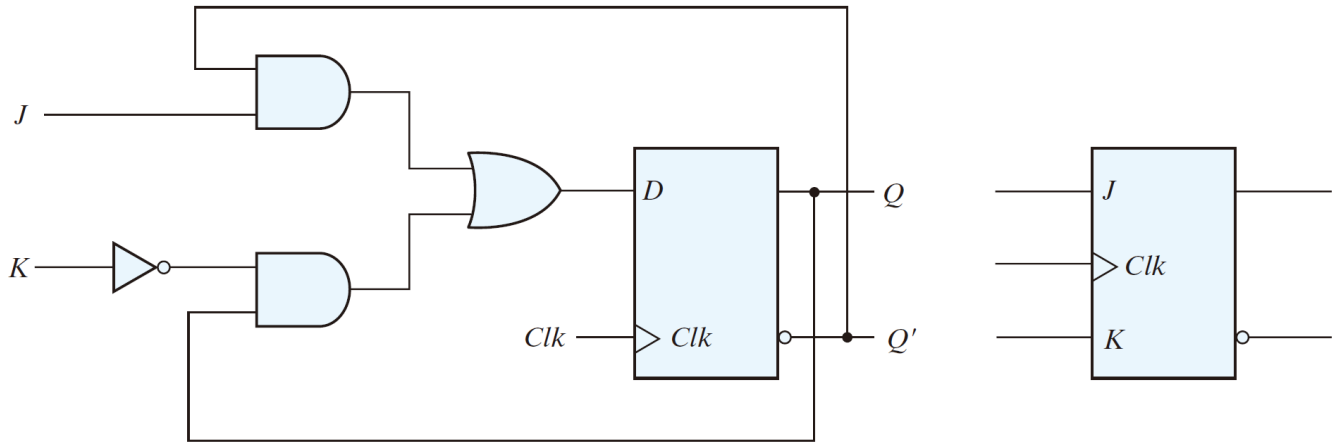

(a) Negative-edge

# JK Flip-Flops

- A JK flip-flop has the following characteristic equation:
  - $Q(t+1) = JQ' + K'Q$

**JK Flip-Flop characteristic table**

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




(a) Circuit diagram (b) Graphic symbol

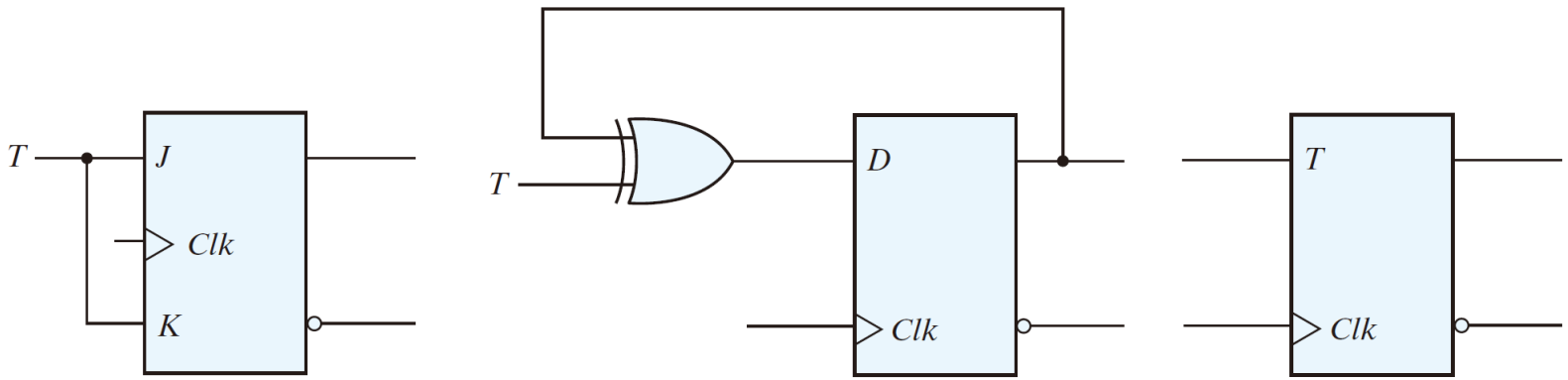
# T Flip-Flops

- A T flip-flop has following characteristic equation:
  - $Q(t+1) = T \oplus Q = TQ' + T'Q$

**T Flip-Flop characteristic table**

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

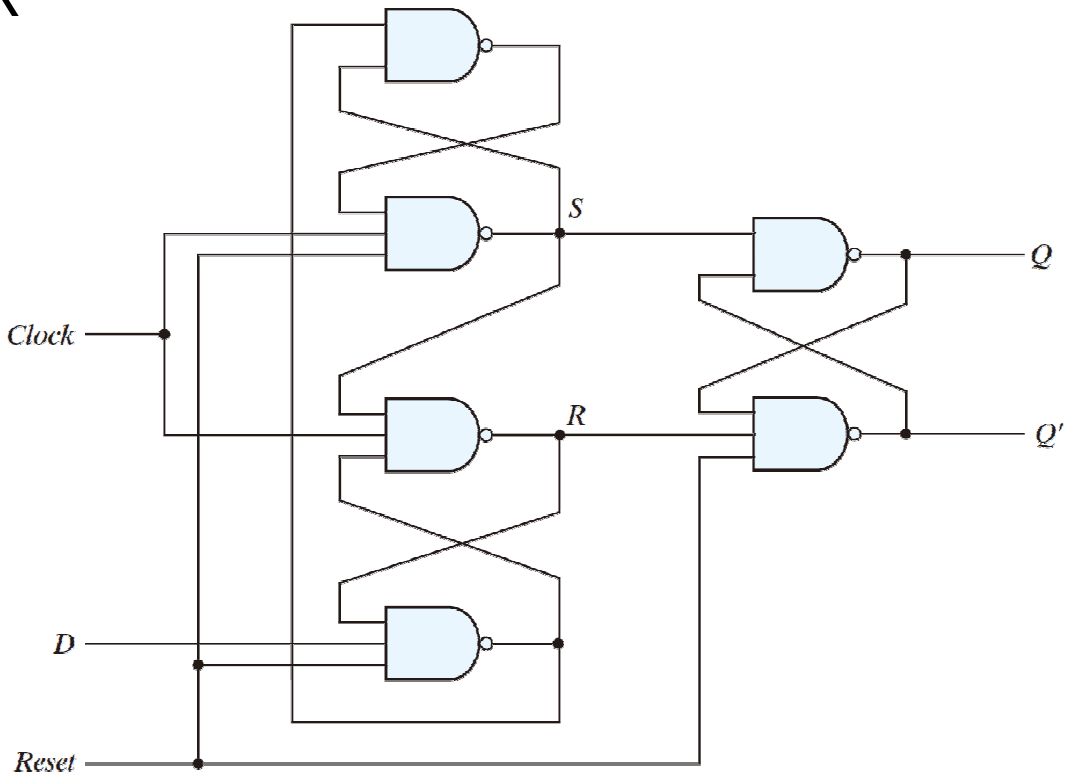
  - Circuit diagram:




(a) From *JK* flip-flop (b) From *D* flip-flop (c) Graphic symbol

# Asynchronous Reset/Set (1/2)

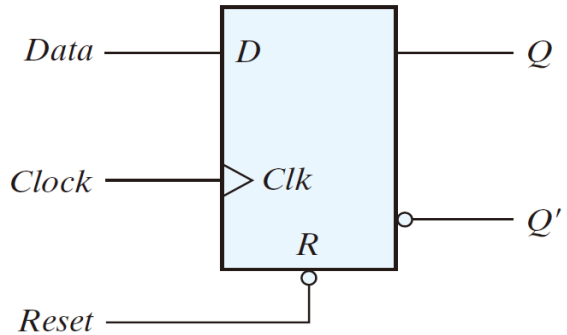
- Some flip-flop has a asynchronous set/reset line that forces the flip-flop to a particular state regardless of the clock




(a) Circuit diagram

# Asynchronous Reset/Set (2/2)

- The graphic symbol of a D flip-flop with asynchronous reset/set is as follows:



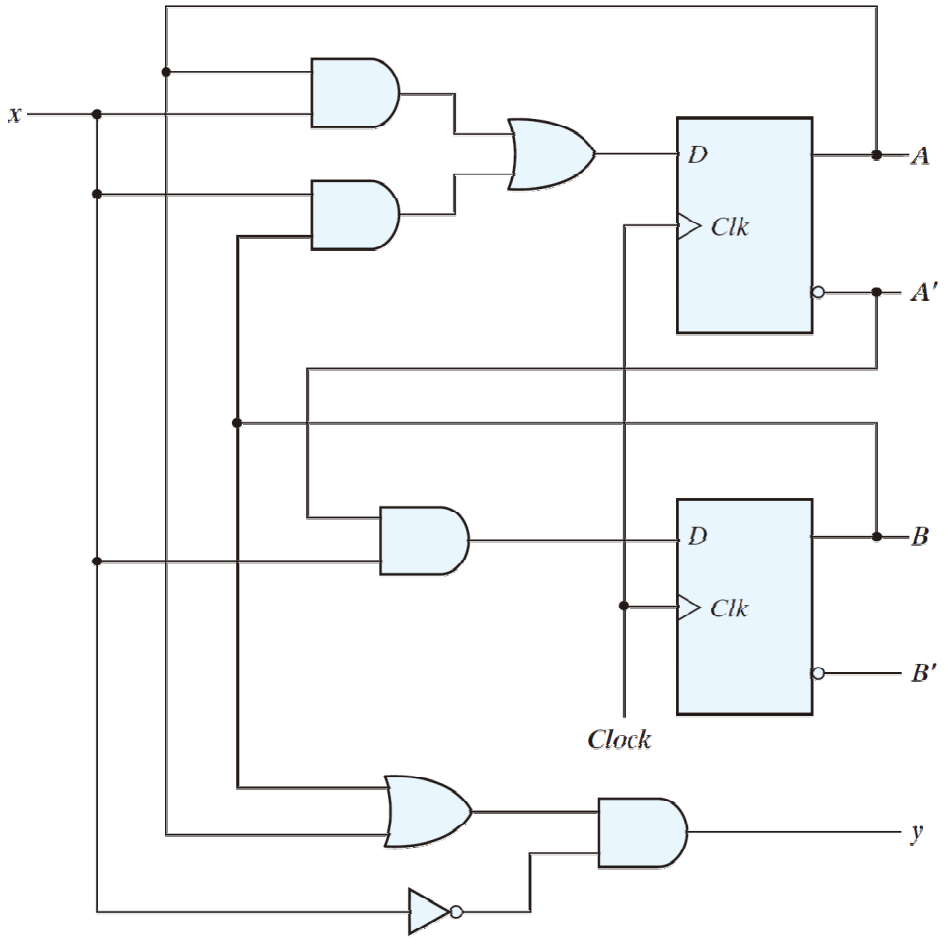

(b) Graphic symbol

| *R* | *Clk* | *D* | *Q* | *Q′* |
|---|---|---|---|---|
| 0 | X | X | 0 | 1 |
| 0 | ↑ | 0 | 0 | 1 |
| 0 | ↑ | 1 | 1 | 0 |

(b) Function table

# Analysis of Sequential Circuit

- A zero-detector in a sequence of binary numbers can be implemented as:



# State Equations

- The behavior of a clocked sequential circuit can be described by state equations
- Example of state equations:
  - $A(t+1) = A(t)x(t) + B(t)x(t)$ (or $A(t+1) = Ax + Bx$)
  - $B(t+1) = A(t)'x(t)$ (or $B(t+1) = A'x$)
- The output equations:
  - $y(t) = [A(t) + B(t)]\, x'(t)$ (or $y = (A + B)x'$)

# State Table (1/2)

- Example of the state table of previous circuit:

| 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 |

# State Table (2/2)

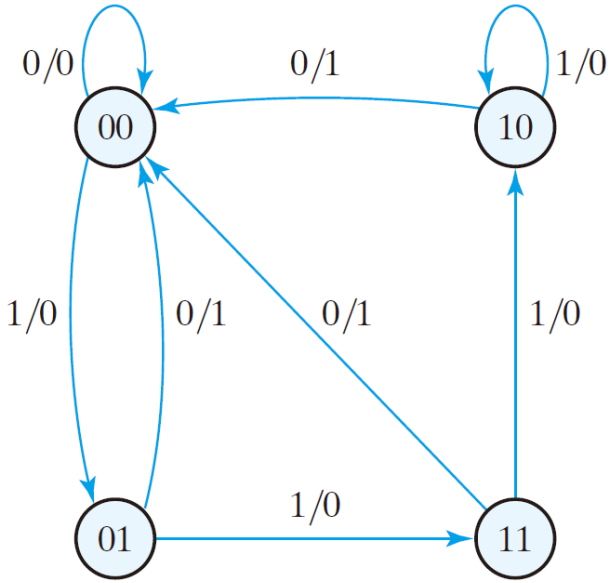
- A more condensed form of the state table is as follows:

*Second Form of the 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 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

# State Diagram

- A state transition diagram illustrates the transition of state and the corresponding output based on the current state and input:
  - Each circle represents a state
  - The numbers on each link represents input/output
- Example:



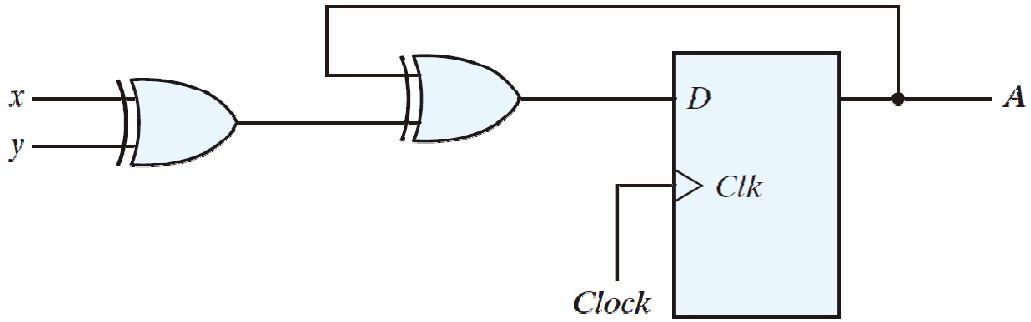

| Present State | | Next State | | | | Output | |
|---|---|---|---|---|---|---|---|
| | | $x = 0$ | | $x = 1$ | | $x = 0$ | $x = 1$ |
| $A$ | $B$ | $A$ | $B$ | $A$ | $B$ | $y$ | $y$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

# Flip-Flop Input Equations

- The part of circuit that generates the inputs to flip-flops is called the input equations
  - Also called excitation functions
- Example of the previous circuit
  - $D_Q = x + y$
  - $D_A = Ax + Bx$
  - $D_B = A'x$

# Analysis with D Flip-Flops

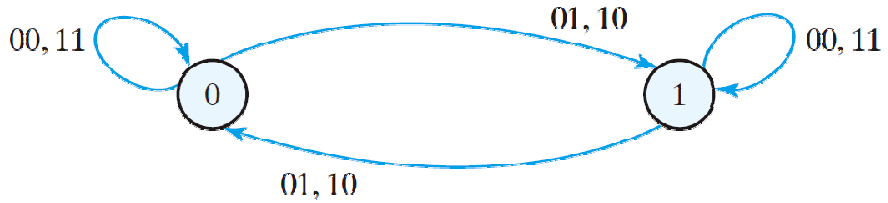
- Analyze the circuit with a D flip-flop:
  - Input equation: $D_A = A \oplus x \oplus y$
  - State equation: $A(t+1) = A \oplus x \oplus y$




(a) Circuit diagram

| Present state | Inputs | | Next state |
|---|---|---|---|
| $A$ | $x$ | $y$ | $A$ |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

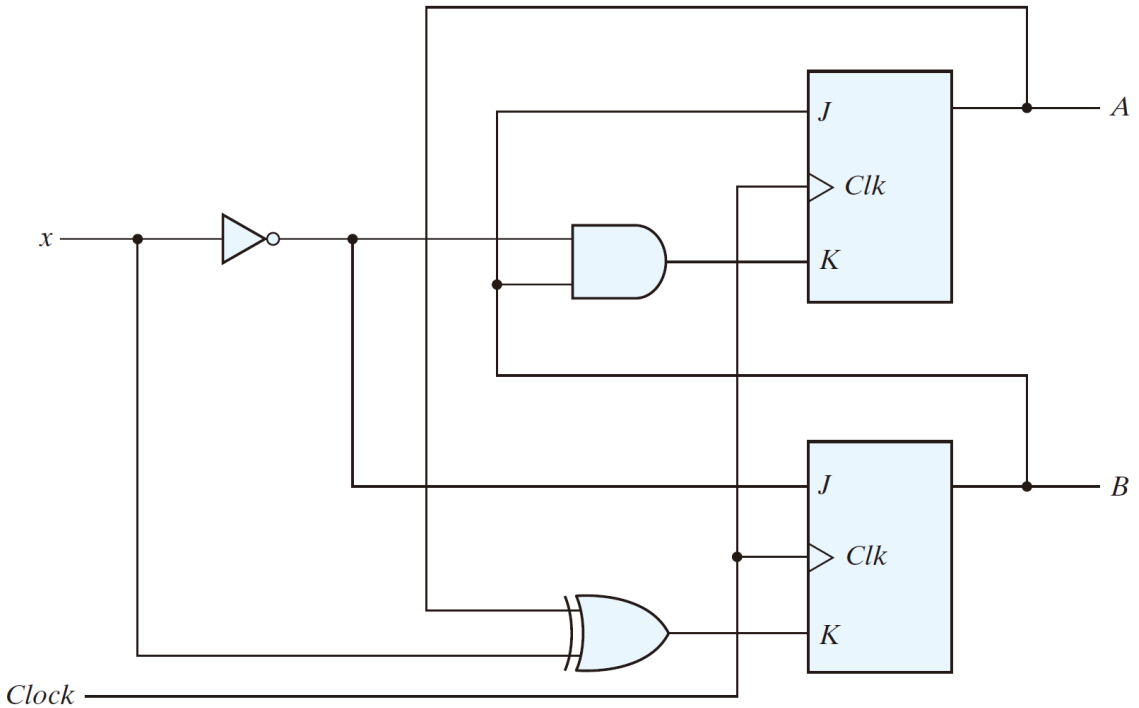
(b) State table




(c) State diagram

# Analysis with JK Flip-Flops (1/3)

- Steps of sequential circuit analysis
  - Determine the flip-flop input functions in terms of the present state and input variables
  - Use the flip-flop characteristic table to get the next state



# Analysis with JK Flip-Flops (2/3)

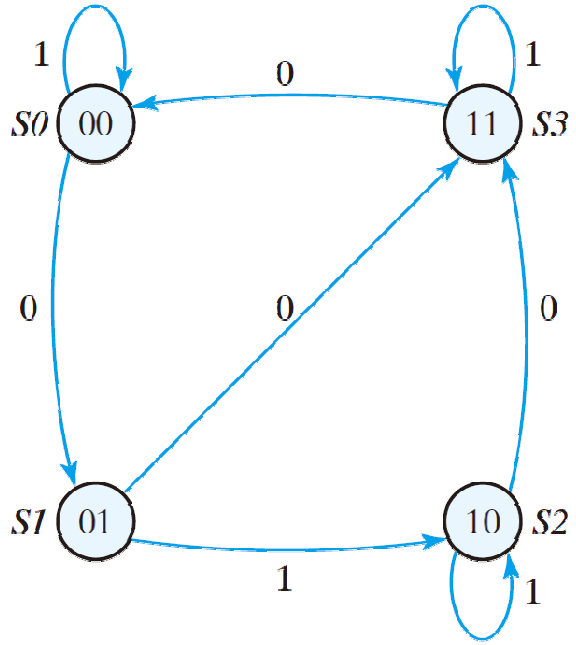
- Input functions:
  - $J_A = B,\ K_A = Bx'$
  - $J_B = x',\ K_B = A'x + Ax'$
- State table:

*State Table for Sequential Circuit with JK Flip-Flops*

| 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 |

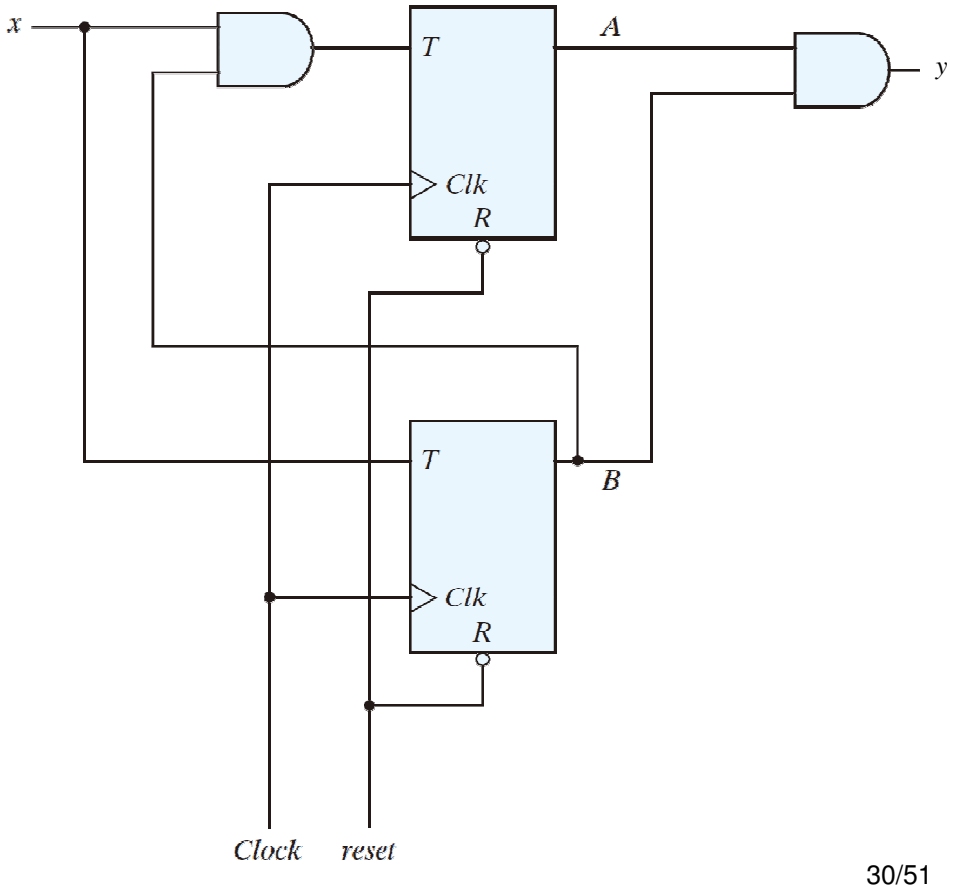
# Analysis with JK Flip-Flops (3/3)

- Derive the state equations using characteristic eq.
  - $A(t+1) = JA' + K'A = BA' + (Bx')'A = A'B + AB' + Ax$
  - $B(t+1) = x'B + (A \oplus x)'B = B'x + ABx + A'Bx'$.
- The state transition diagram:



# Analysis with T Flip-Flops (1/2)

- The characteristic equation of a binary counter:
  - $Q(t+1) = T \oplus Q' = TQ' + T'Q$
- Input functions:
  - $T_A = Bx$
  - $T_B = x$
- Output functions:
  - $y = AB$

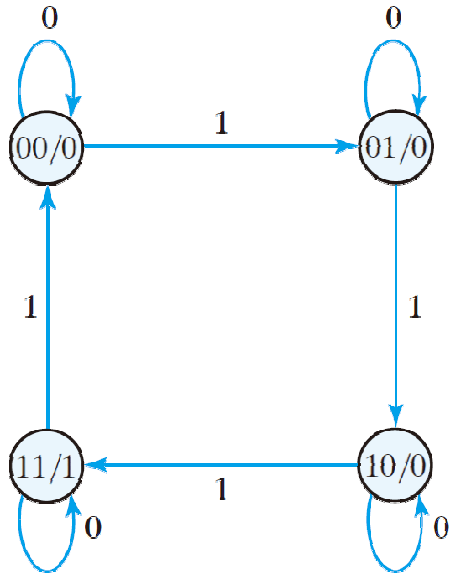


# Analysis with T Flip-Flops (2/2)

- State equations:
  - $A(t+1) = (Bx)'A + (Bx)A' = AB' + Ax' + A'Bx$
  - $B(t+1) = x \oplus B$
- State table and state diagram:

*State Table for Sequential Circuit with T Flip-Flops*

| 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 |

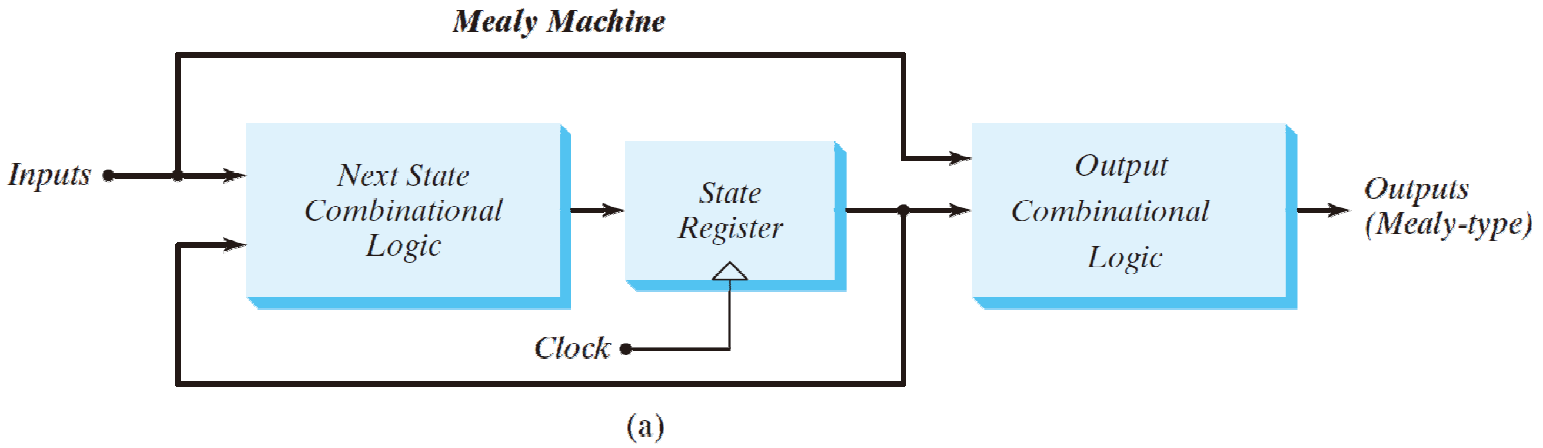


# Mealy and Moore Models (1/2)

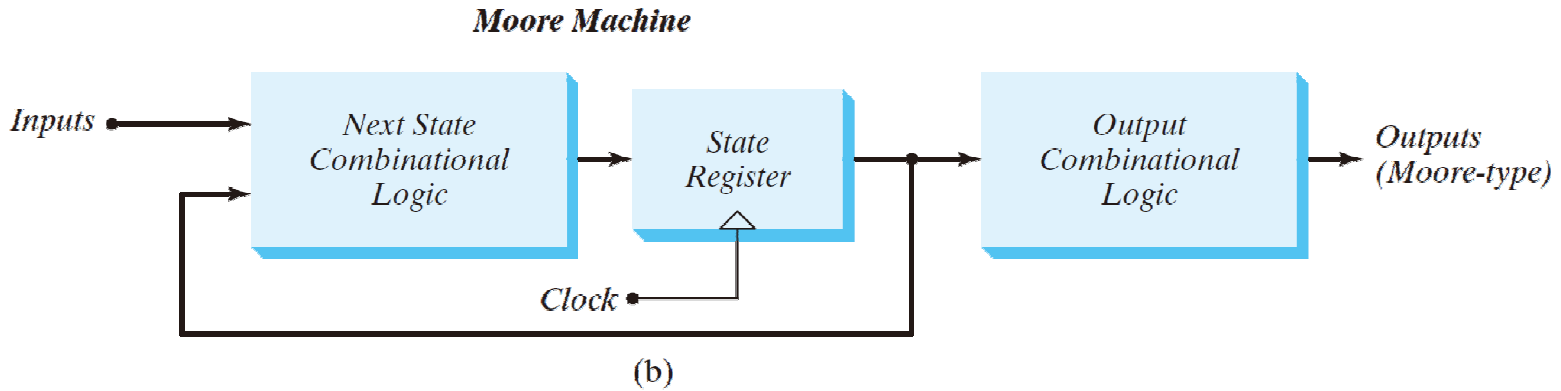
- The Mealy model: the outputs are functions of both the present state and inputs
  - The outputs are combinational circuits of the input; the outputs may have momentary false values unless the inputs are synchronized with the clocks
- The Moore model: the outputs are functions of the present state only
  - The outputs are synchronized with the clocks

# Mealy and Moore Models (2/2)

- Block diagrams of Mealy and Moore models:



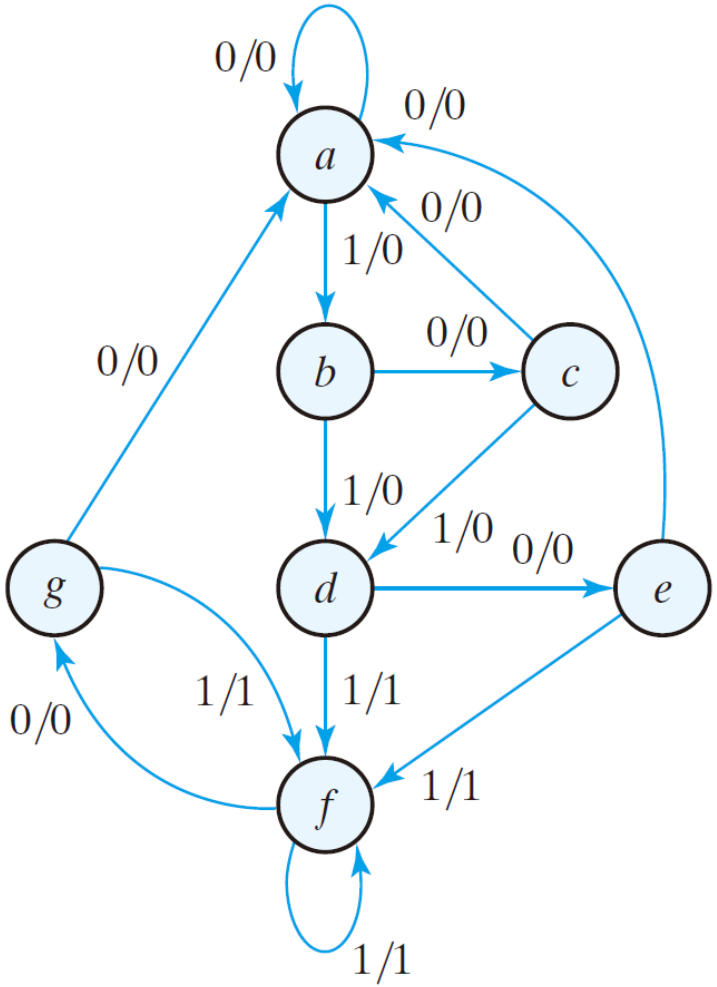

(a)




(b)

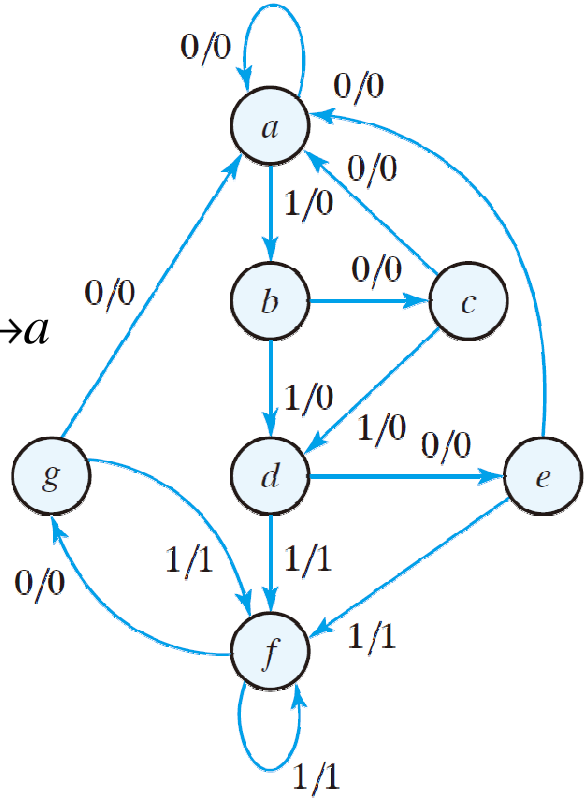
# State Reduction

- Often, we want to reduce the number of flip-flops and the number of gates
- A reduction in the number of states may result in a reduction in the number of flip-flops and gates



# State Reduction Principle

- Two circuits are equivalent if they
  - Have identical outputs for all input sequences
  - The number of states is not important
- Example: if the initial state is $a$
  - Input: $0\rightarrow1\rightarrow0\rightarrow1\rightarrow0\rightarrow1\rightarrow1\rightarrow0\rightarrow1\rightarrow0\rightarrow0$
  - Out: $0\rightarrow0\rightarrow0\rightarrow0\rightarrow0\rightarrow1\rightarrow1\rightarrow0\rightarrow1\rightarrow0\rightarrow0$
  - State: $a\rightarrow a\rightarrow b\rightarrow c\rightarrow d\rightarrow e\rightarrow f\rightarrow f\rightarrow g\rightarrow f\rightarrow g\rightarrow a$
- Two state are equivalent if
  - For each member of the set of inputs, they give exactly the same output and send the circuit to the same state or to an equivalent state.



# Equivalent State Example (1/3)

- Initial state table:
  - Since $e \equiv g$, we can replace all $g$ by $e$

| 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 |
| $d$ | $e$ | $f$ | 0 | 1 |
| $e$ | $a$ | $f$ | 0 | 1 |
| $f$ | $g$ | $f$ | 0 | 1 |
| $g$ | $a$ | $f$ | 0 | 1 |

- 2nd state table:
  - Since $d \equiv f$, we can replace all $f$ by $d$

| 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 |
| $d$ | $e$ | $f$ | 0 | 1 |
| $e$ | $a$ | $f$ | 0 | 1 |
| $f$ | $e$ | $f$ | 0 | 1 |

# Equivalent State Example (2/3)

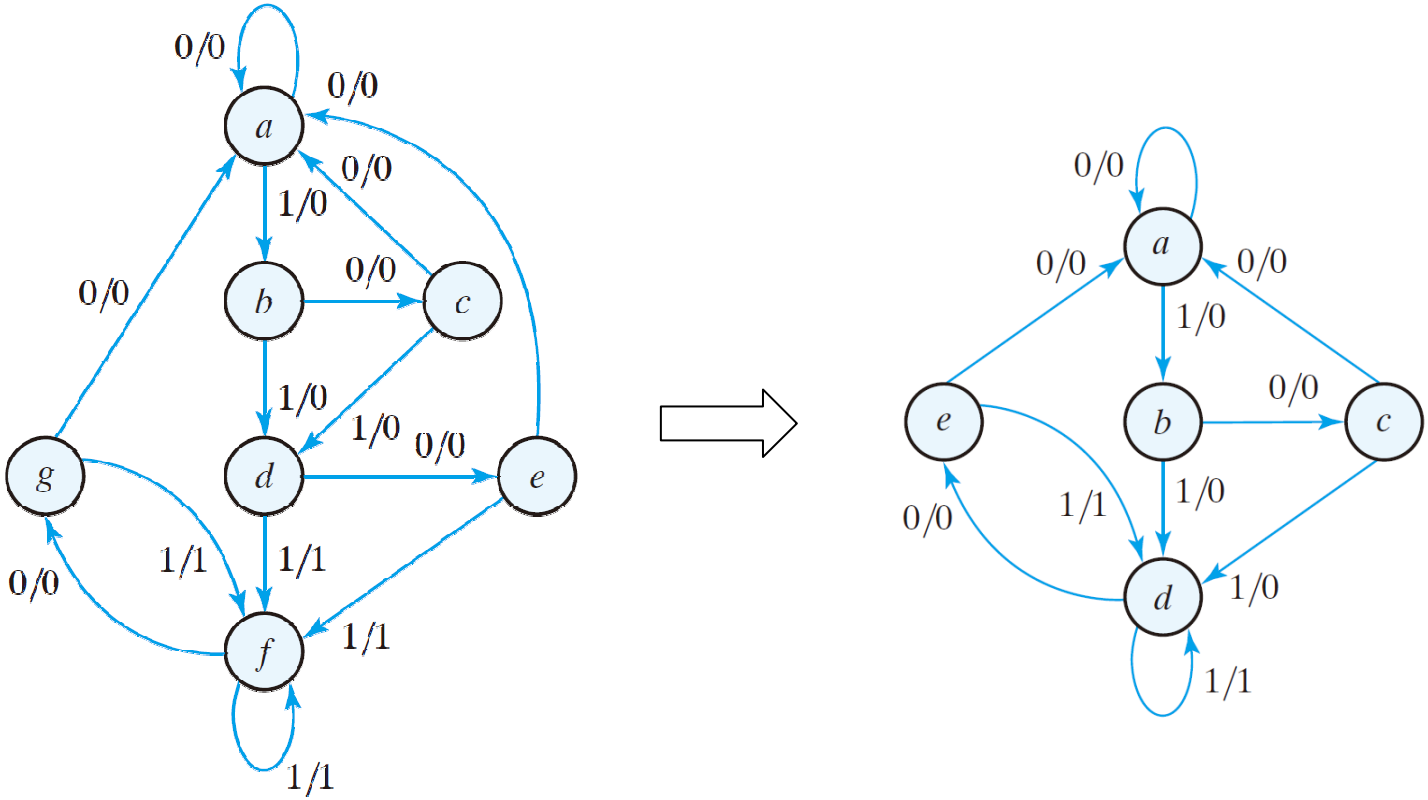
- The reduced finite state machine:

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

- Input-output behavior:
  - Input: $0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 0$
  - Out: $0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 1 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 0$
  - State: $a \rightarrow a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow d \rightarrow d \rightarrow e \rightarrow d \rightarrow e \rightarrow a$

# Equivalent State Example (3/3)

- Checking of each pair of states for equivalence can be done systematically via an implication table
- State reduction may not be necessary in practice!



# State Assignment (1/2)

- For circuit implementation, each state must be assigned a binary value
- There are three possible ways to encode state values:

*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 |

# State Assignment (2/2)

- Any binary number assignment is satisfactory as long as each state is assigned a unique number
  - One-hot may be simpler because we do not need an extra decoder to decode the state value
- Example: binary assignment

*Reduced State Table with Binary Assignment 1*

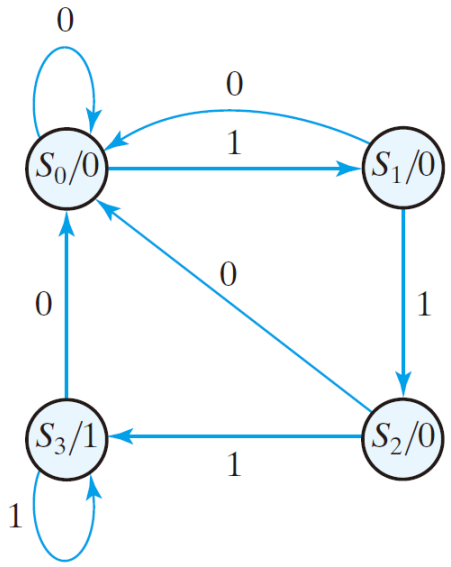
| | Next State | | Output | |
|---|---|---|---|---|
| **Present State** | $x = 0$ | $x = 1$ | $x = 0$ | $x = 1$ |
| 000 | 000 | 001 | 0 | 0 |
| 001 | 010 | 011 | 0 | 0 |
| 010 | 000 | 011 | 0 | 0 |
| 011 | 100 | 011 | 0 | 1 |
| 100 | 000 | 011 | 0 | 1 |

# Design Procedure for Large Circuits

- To design VLSI, we do not concern with the type of flip-flops or gates used in the implementation; instead, we focus on describing the “behaviors” to EDA tools
- However, it is still good to know the manual methods to design a sequential behavior
  - The word description of the circuit behavior (a state diagram)
  - State reduction if necessary
  - Assign binary values to the states
  - Obtain the binary-coded state table
  - Choose the type of flip-flops to be used
  - Derive the flip-flop input equation and output equations
  - Draw the logic diagram

# Synthesis Using D Flip-Flops (1/3)

- A sequence detector that detects a sequence of three or more consecutive 1's in a binary stream can be described by the following state diagram and table:

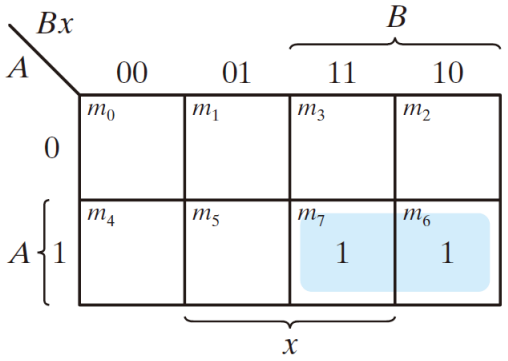



*State Table for Sequence Detector*

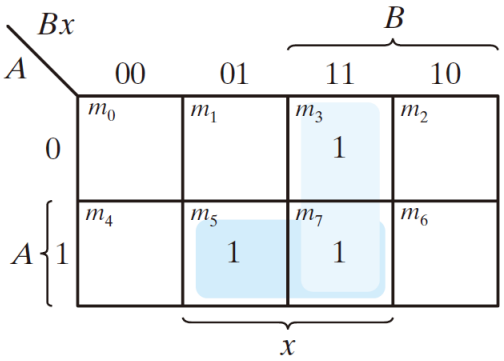
| Present State | | Input | Next State | | Output |
|---|---|---|---|---|---|
| *A* | *B* | *x* | *A* | *B* | *y* |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

# Synthesis Using D Flip-Flops (2/3)

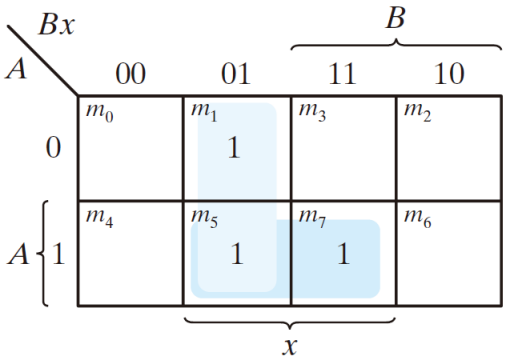
- The flip-flop input equations and the output equation
  - $A(t+1) = D_A(A, B, x) = \Sigma(3, 5, 7)$
  - $B(t+1) = D_B(A, B, x) = \Sigma(1, 5, 7)$
  - $y(A, B, x) = \Sigma(6, 7)$
- Logic minimization
  - $D_A = Ax + Bx$
  - $D_B = Ax + B'x$
  - $y = AB$




$y = AB$



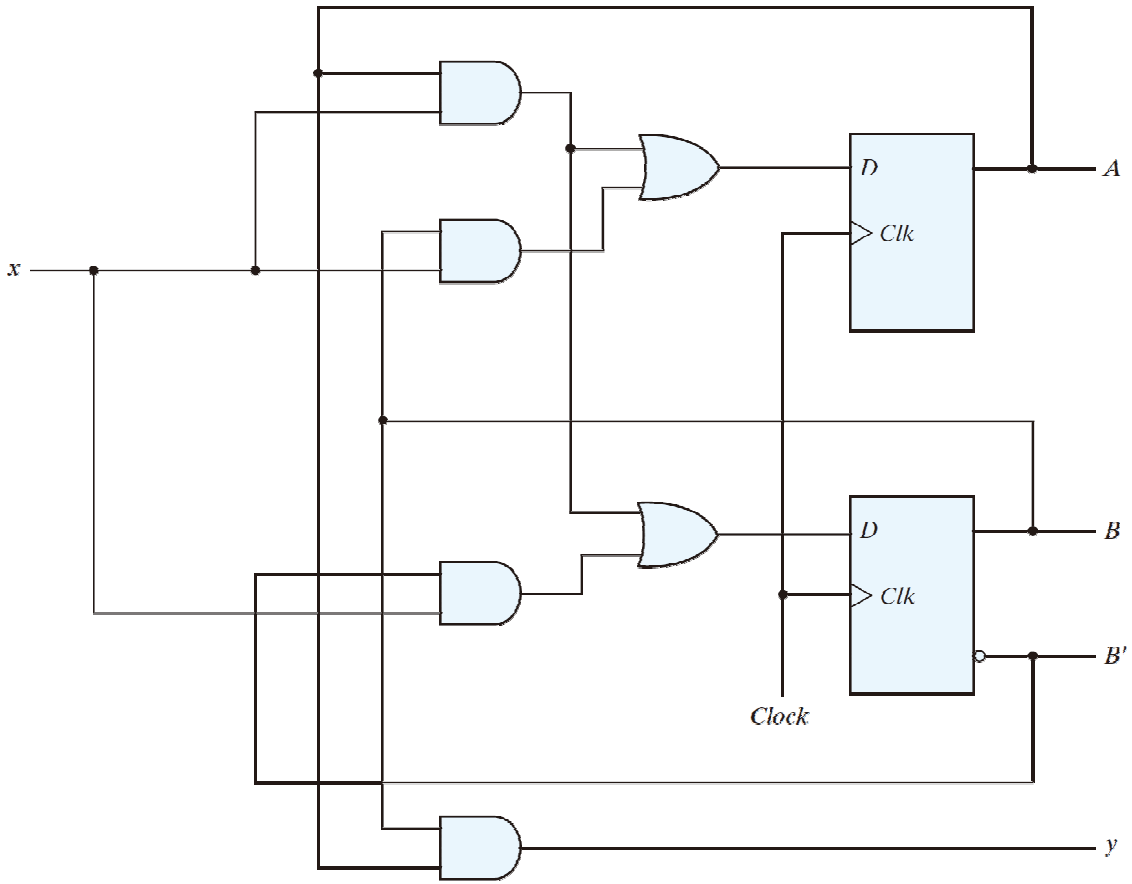

$D_A = Ax + Bx$




$D_B = Ax + B'x$

# Synthesis Using D Flip-Flops (3/3)

- Logic diagram (Moore-type machine):



# Synthesis with Other Flip-Flops

- The mapping between the input functions of the state machine and the input functions of flip-flops is trivial only when D-type flip-flops are used
- For other flip-flops, we need to use the excitation tables to design the input functions to flip-flops:

*Flip-Flop Excitation Tables*

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

(a) *JK* Flip-Flop

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

(b) *T* Flip-Flop

# Synthesis Using JK Flip-Flops (1/3)

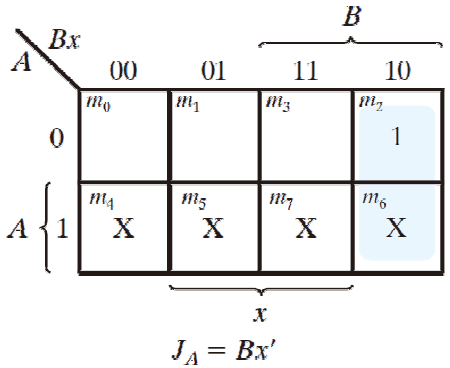
- The behavior of the sequence detector using JK flip-flops can be described by the following state table:

*State Table and JK Flip-Flop Inputs*

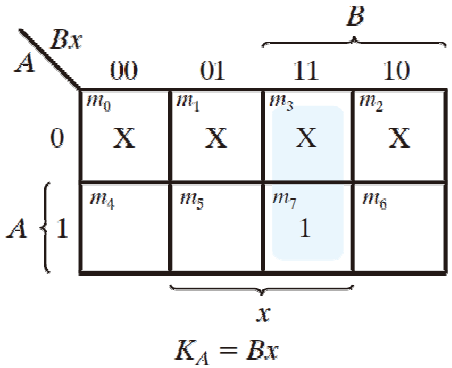
| Present State | | Input | Next State | | Flip-Flop Inputs | | | |
|---|---|---|---|---|---|---|---|---|
| **A** | **B** | **x** | **A** | **B** | $J_A$ | $K_A$ | $J_B$ | $K_B$ |
| 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X |
| 0 | 0 | 1 | 0 | 1 | 0 | X | 1 | X |
| 0 | 1 | 0 | 1 | 0 | 1 | X | X | 1 |
| 0 | 1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

# Synthesis Using JK Flip-Flops (2/3)

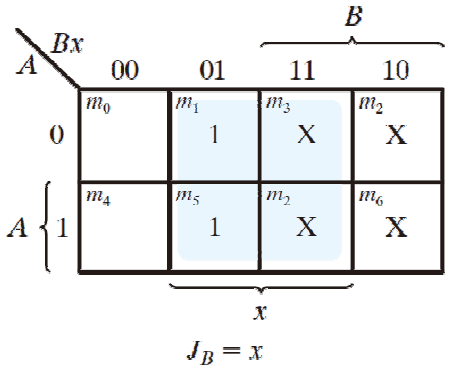
- Input functions:
  - $J_A = Bx'$, $K_A = Bx$
  - $J_B = x$, $K_B = (A \oplus x)'$
  - $y = AB$



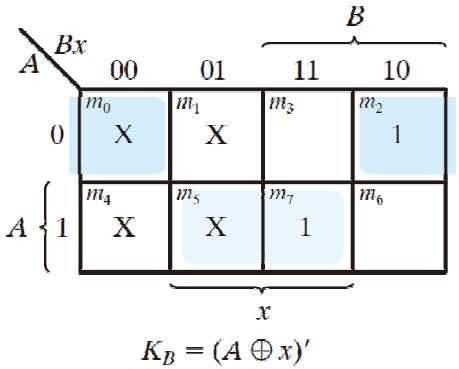

$J_A = Bx'$




$K_A = Bx$



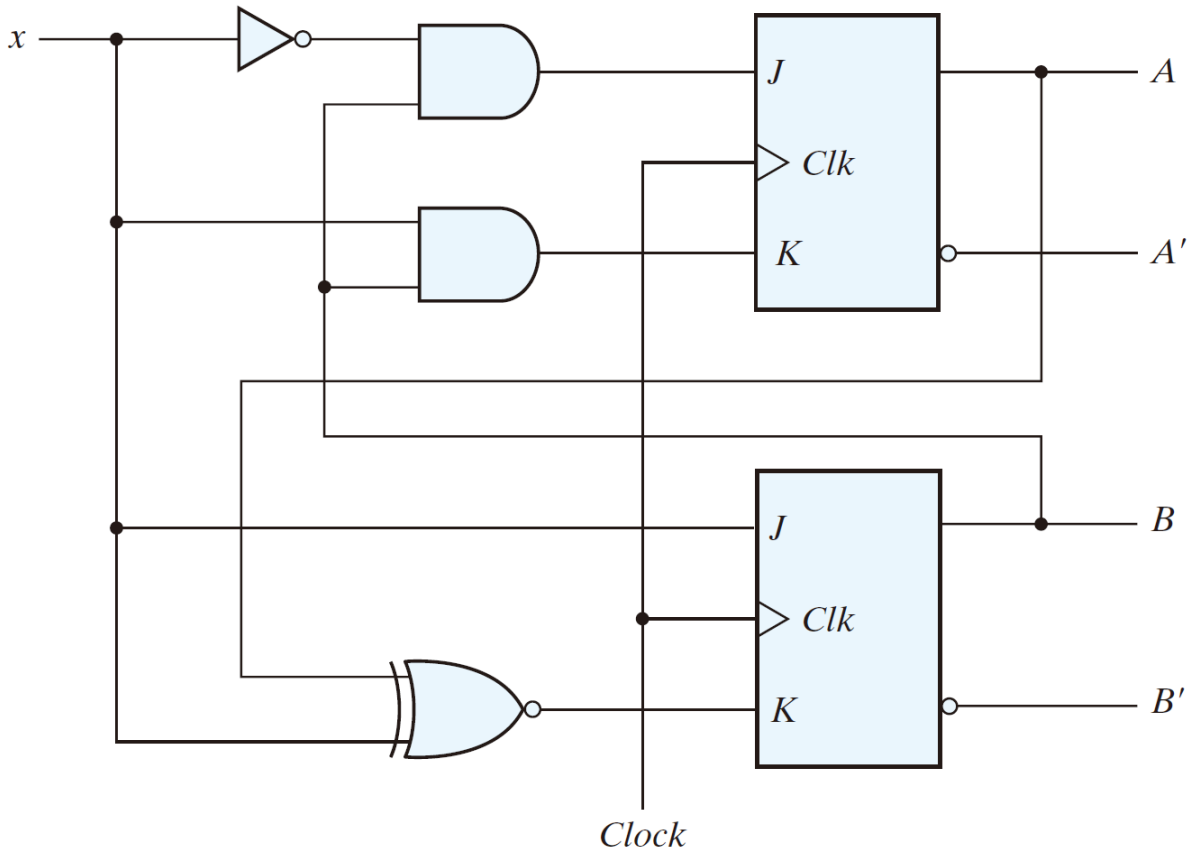

$J_B = x$




$K_B = (A \oplus x)'$

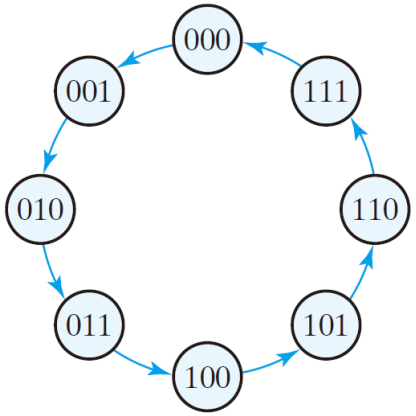
# Synthesis Using JK Flip-Flops (3/3)

- The logic diagram:



# Synthesis Using T Flip-Flops (1/3)

- The state diagram and state table of a 3-bit binary counter are as follows; there is no input, only output

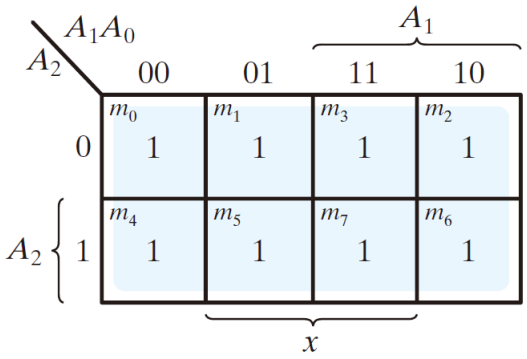



*State Table for Three-Bit Counter*

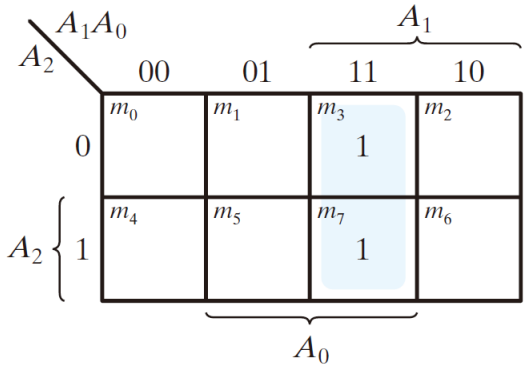
| Present State | | | Next State | | | Flip-Flop Inputs | | |
|---|---|---|---|---|---|---|---|---|
| $A_2$ | $A_1$ | $A_0$ | $A_2$ | $A_1$ | $A_0$ | $T_{A2}$ | $T_{A1}$ | $T_{A0}$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0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

# Synthesis Using T Flip-Flops (2/3)

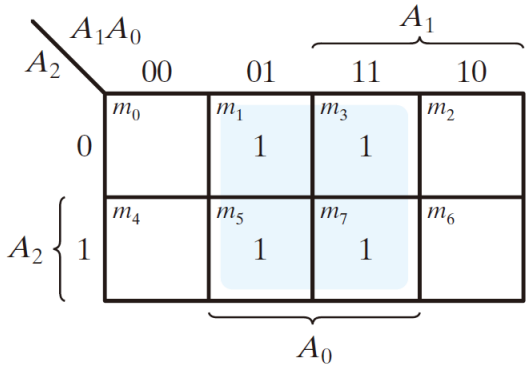
- Simplified functions:
  - $T_{A_0} = 1$
  - $T_{A_1} = A_0$
  - $T_{A_2} = A_1 A_2$




$T_{A0} = 1$




$T_{A2} = A_1 A_0$




$T_{A1} = A_0$

# Synthesis Using T Flip-Flops (3/3)

- The logic diagram: