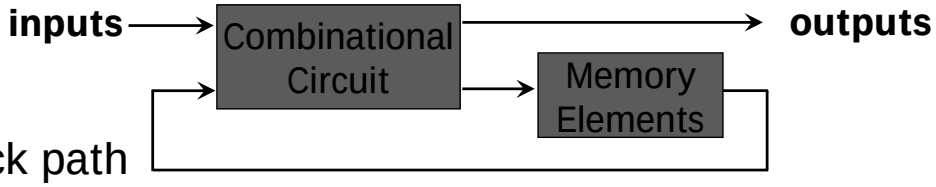
Chapter 5
Synchronous Sequential Logic

Outline

- Sequential Circuits
- Storage Elements: Latches
- Storage Elements: Flip-Flops
- Analysis of Clocked Sequential Circuits
- Synthesizable HDL Model of Sequential Circuits
- State Reduction and Assignment
- Design Procedure

Sequential Circuits

- Combinational circuits contain no memory elements and the outputs depends on the current inputs.
- Sequential circuits:

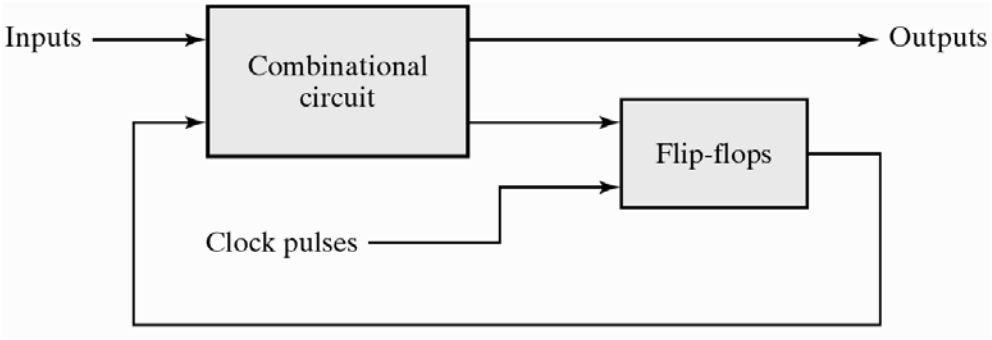


  - a feedback path
  - Memory to store the state of the sequential circuit
  - Combinational circuit computes (inputs, current state) $\Rightarrow$ (outputs, next state)
  - synchronous: the transition happens at discrete instants of time
  - asynchronous: depends on the input signals at any instant of time and the order in which the inputs change. (instable)

Synchronous Sequential Circuits

- A master-clock generator generates a periodic train of clock pulses and distributes it throughout the system.
- The clocked sequential circuit has no instability problems
- the memory elements: Flip-Flops
  - Binary cells capable of storing one bit of information
  - Two outputs: one for the normal value and one for the complement value
  - Maintain a stable binary state indefinitely until directed by an input signal to switch states

Synchronous Clocked Sequential Circuit

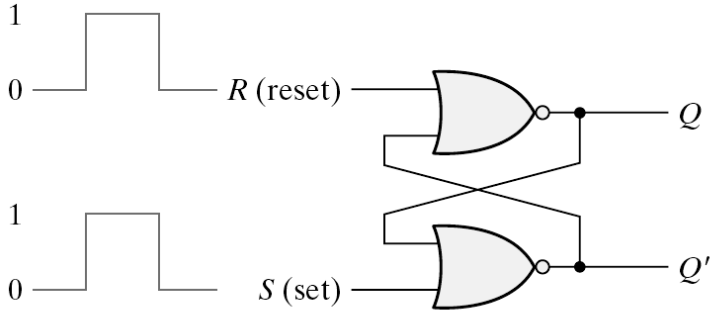


(b) Timing diagram of clock pulses

1. The stored value in a flip-flop is updated when a pulse of the clock signal occurs.
2. The output of the combinational circuit (CC) must reach a stable state before the next clock pulse. ⇒ speed limited by CC.
3. If clock pulse is not active ⇒ feedback loop is broken ⇒ value of flip-flop is not updating.

SR Latch with NOR Gates

- Latch can be formed by two cross-coupled NOR gates
  - (S, R) = (0, 0): no operation (Q and Q' state unchanged)
    (S, R) = (0, 1): reset state (Q = 0 and Q' = 1)
    (S, R) = (1, 0): set state (Q = 1 and Q' = 0)
    (S, R) = (1, 1): undefined state (Q = Q' = 0)
  - Latch is an asynchronous sequential circuit. (state changes whenever inputs change).
  - The (S, R) must go back to (0, 0) before any other change to avoid the occurrence of the undefined state.



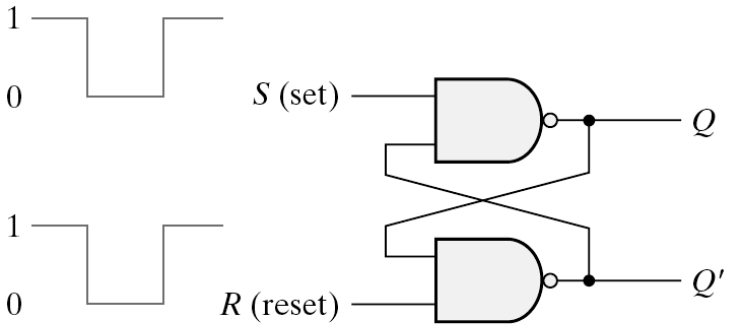
(a) Logic diagram

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

SR Latch with NAND Gates

- Latch can be formed by two cross-coupled NAND gates
  - (S, R) = (1, 1): no operation (Q and Q' state unchanged)
    (S, R) = (1, 0): reset state (Q = 0 and Q' = 1)
    (S, R) = (0, 1): set state (Q = 1 and Q' = 0)
    (S, R) = (0, 0): undefined state (Q = Q' = 1)
  - The (S, R) must go back to (1, 1) before any other change to avoid the occurrence of the undefined state.



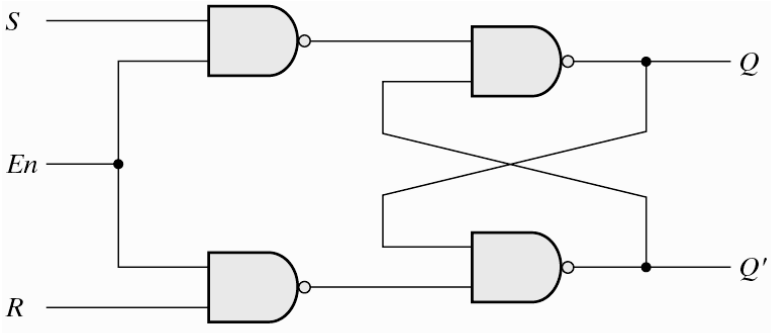
(a) Logic diagram

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

SR Latch with Control

- SR latch with control input
  - En = 0, no change
  - En = 1, operate as normal SR latch



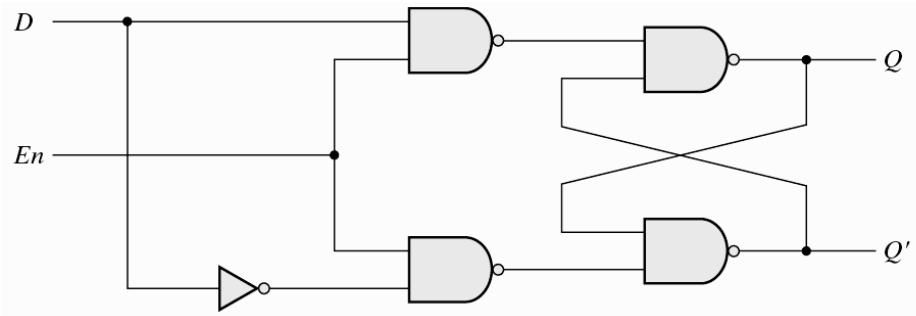
(a) Logic diagram

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

D Latch

- D Latch
  - D latch can eliminate the undesirable conditions of the indeterminate state in the SR latch
  - D: data
  - D ⇒ Q when En = 1; no change when En = 0
  - A *transparent* latch when En = 1

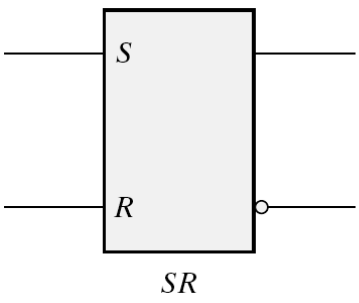


(a) Logic diagram

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

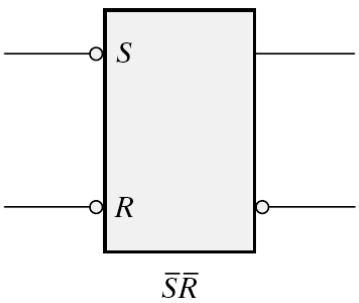
(b) Function table

Graphic Symbols for Latches



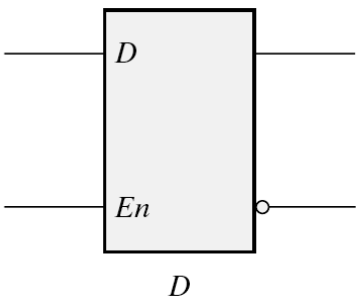
SR

NOR



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

NAND



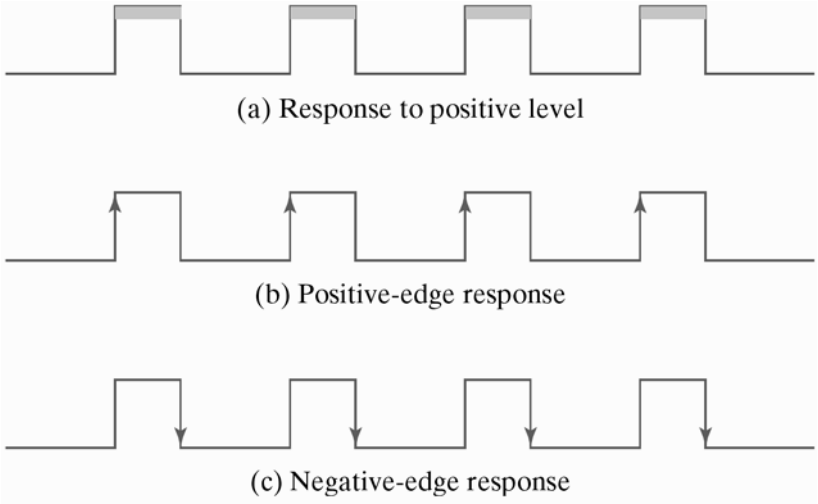
D

Trigger

- A trigger
  - The state of a latch or flip-flop is switched by a change of the control input. This momentary change is called trigger.
- Level triggered – latches
  - The state transition starts as soon as clock is during logic 1 or logic 0 level.
  - The change of input makes the combination logic keep changing with the input latch at logic 1 or logic 0.
- Edge triggered – Flip-Flops
  - The state transition starts only at positive or negative edge of the clock signal.
  - The edge trigger flip-flops will isolate the input changes (current state) and output driving logic (previous state).

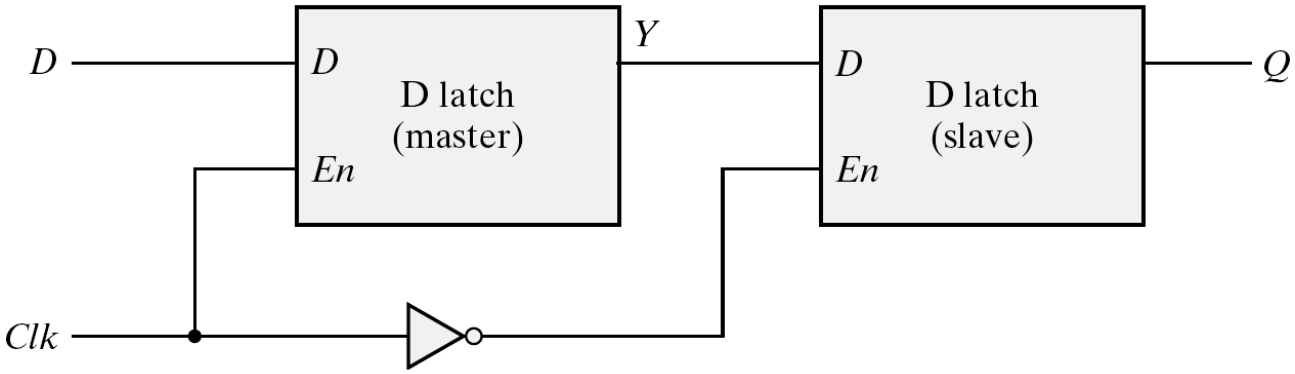
Edged-Triggered Flip-Flops

- If level-triggered latches are used,
  - The output result will affect the inputs through the feedback path and cause instability problem.
- If edge-triggered latches are used,
  - the output of flip-flop is isolated from being affected by the input signals, and
  - the state transition happens only at the clock edge.



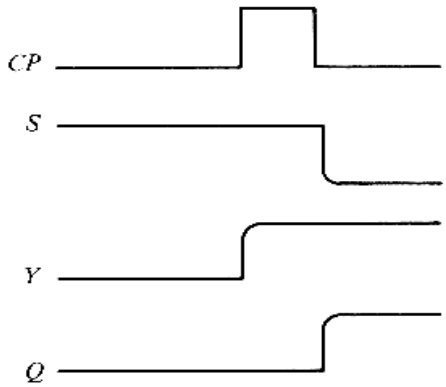
Edge-Triggered D Flip-Flop

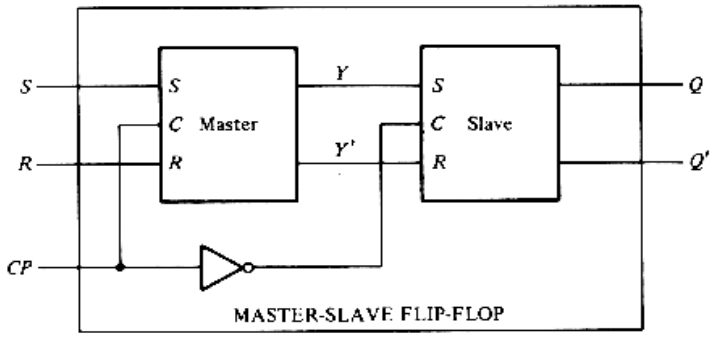
- Master-slave D flip-flop is formed by two separate latches and one inverter.
  - A master D latch (positive-level triggered)
  - A slave D latch (negative-level triggered)



Edge-Triggered D Flip-Flop

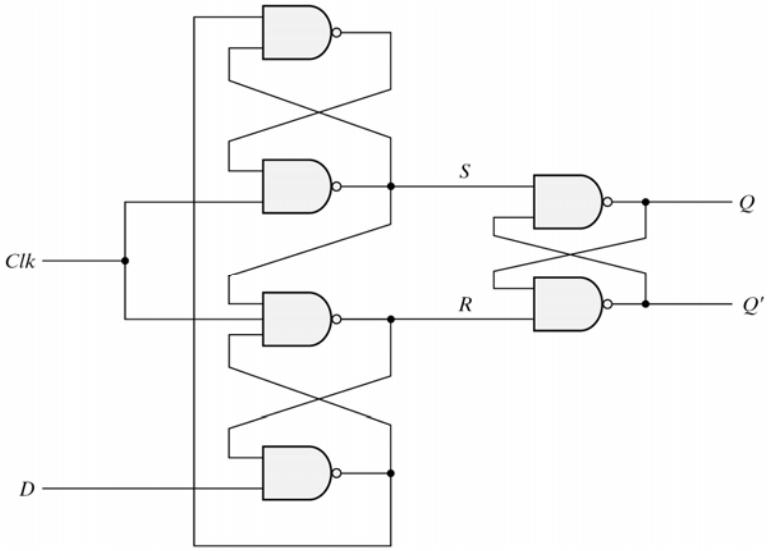
- Two stage operations
  - CP = 1:
    - Master D latch active: $(S, R) \Rightarrow (Y, Y')$;
    - Slave D latch inactive: (Q, Q') holds
  - CP = 0:
    - Master D latch inactive: (Y, Y') holds;
    - Slave D latch active: $(Y, Y') \Rightarrow (Q, Q')$
- (S, R) could not affect (Q, Q') directly.
- The state changes at the negative edge transition of CP.





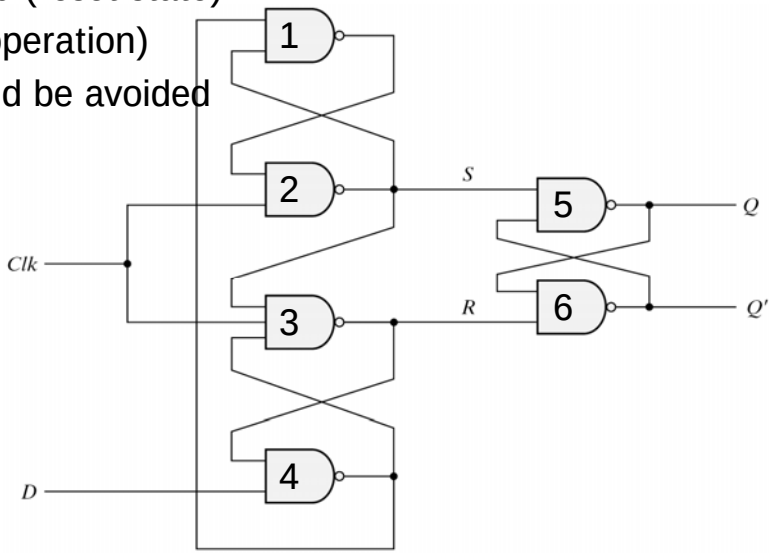
Edge-Triggered Flip-Flops

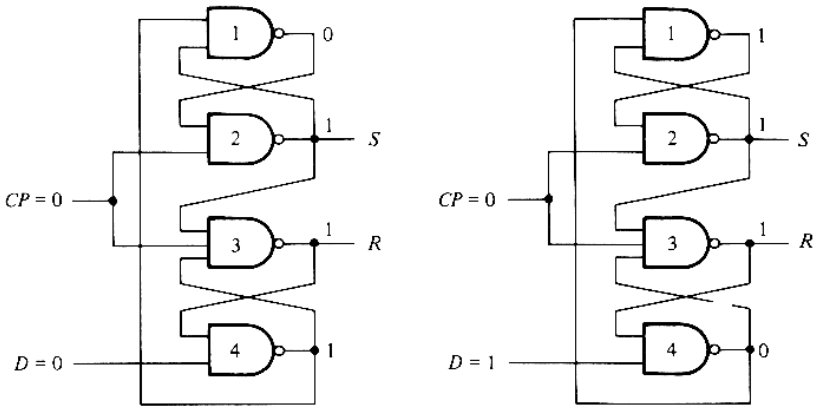
- Edge-triggered flip-flops
  - the state changes during a clock-pulse transition
- A D-type positive-edge-triggered flip-flop with three SR latches.



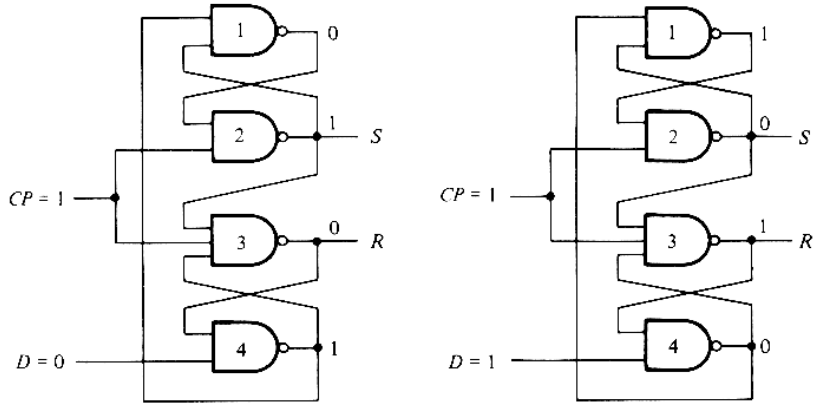
Edge-Triggered Flip-Flops with Three Latches

- Output Latches (5)(6)
  - Clk = 0: (S, R) = (1, 1), no operation
  - Clk = 1: operates as a normal latch
    - (S, R) = (0, 1): Q = 1 (set state)
    - (S, R) = (1, 0): Q = 0 (reset state)
    - (S, R) = (1, 1): (no operation)
    - (S, R) = (0, 0): should be avoided





(a) With $CP = 0$



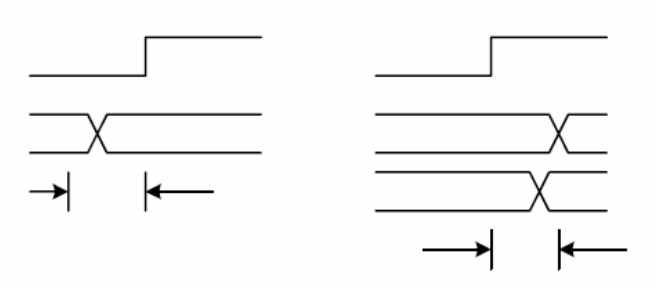
(b) With $CP = 1$

Edge-Triggered Flip-Flops with Three Latches

- Input latches (1)(2)(3)(4)
  - Clk = 0: (S, R) = (1, 1)
    - D = 0 output of (1) = 0, output of (4) = 1
    - D = 1 output of (1) = 1, output of (4) = 0
  - Clk = 1:
    - D = 0 output of (1) = 0, output of (4) = 1
      $\Rightarrow$ (S, R) = (1, 0)
      $\Rightarrow$ Q = 0
    - D = 1 output of (1) = 1, output of (4) = 0
      $\Rightarrow$ (S, R) = (0, 1)
      $\Rightarrow$ Q = 1
- Positive transition of clock (Clk) transfers the input data D to output Q.
- A negative transition of clock (Clk) does not affect the output

Setup Time / Hold Time

- The setup time
  - D input must be maintained at a constant value prior to the application of the positive *Clk* pulse.
  - Equals the propagation delay through gates 4 and 1
  - data to the internal latches
- The hold time
  - D input must not change after the application of the positive *Clk* pulse
  - Equals the propagation delay of gate 3
  - clock to the internal latch

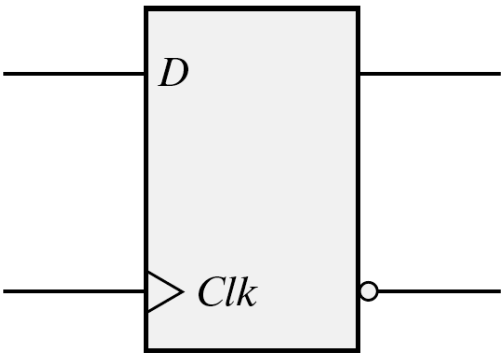


Edge-Triggered Flip-Flops with Three Latches

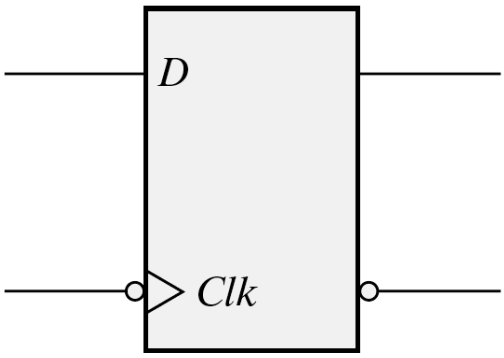
- Summary
  - *Clk* = 0: (*S*, *R*) = (1, 1), no state change
  - *Clk* = ↑ : state change once
  - *Clk* = 1: state holds
  - To eliminate the feedback problems in sequential circuits
- All flip-flops must make their transition at the same time.

Other Flip-Flops

- The edge-triggered D flip-flops is the most economical and efficient because it requires the smallest number of gates.
- Other two flip-flops widely used in the design of digital systems are – ***JK-flip-flops*** and ***T flip-flops*****.**



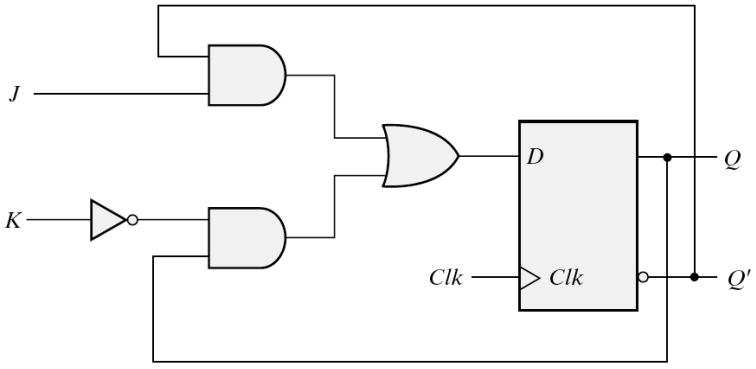
(a) Positive-edge



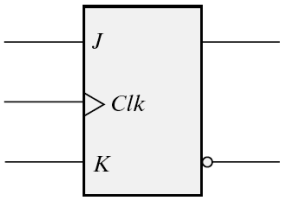
(a) Negative-edge

JK Flip-Flop

- JK Flip-Flop



(a) Circuit diagram

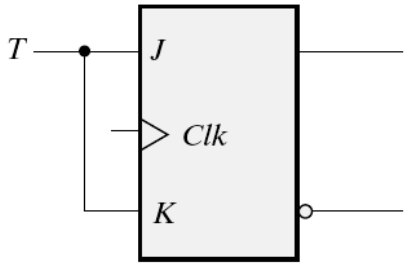


(b) Graphic symbol

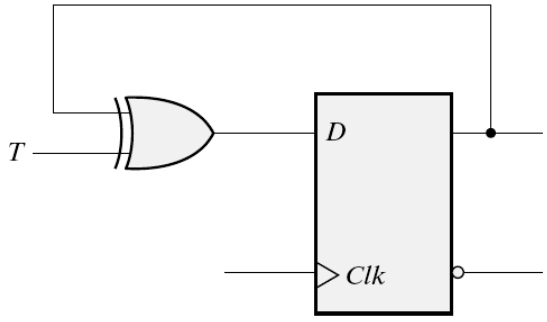
- $D = JQ' + K'Q$
  - $J = 0, K = 0$: $D = Q$, no change (no operation)
  - $J = 0, K = 1$: $D = 0 \Rightarrow Q = 0$ (reset state)
  - $J = 1, K = 0$: $D = 1 \Rightarrow Q = 1$ (set state)
  - $J = 1, K = 1$: $D = Q' \Rightarrow Q = Q'$ (complement state)

T Flip-Flop

- T flip-flop
  - T flip-flop is a complementing flip-flop controlled by input T.



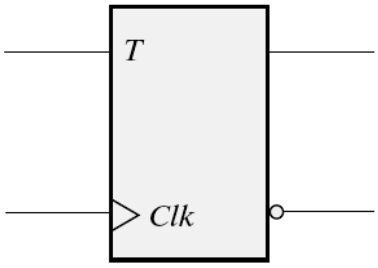
(a) From *JK* flip-flop



(b) From *D* flip-flop

$D = T \oplus Q = TQ' + T'Q$

- T = 0: $D = Q$, no change (no operation)
- T = 1: $D = Q' \Rightarrow Q = Q'$ (complement state)



(c) Graphic symbol

Characteristic Tables

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

- D Flip-Flop
  - $Q(t + 1) = D$
- JK Flip-Flop
  - $Q(t + 1) = JQ' + K'Q$
- T Flip-Flop
  - $Q(t + 1) = T \oplus Q$

D Flip-Flop

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

T Flip-Flop

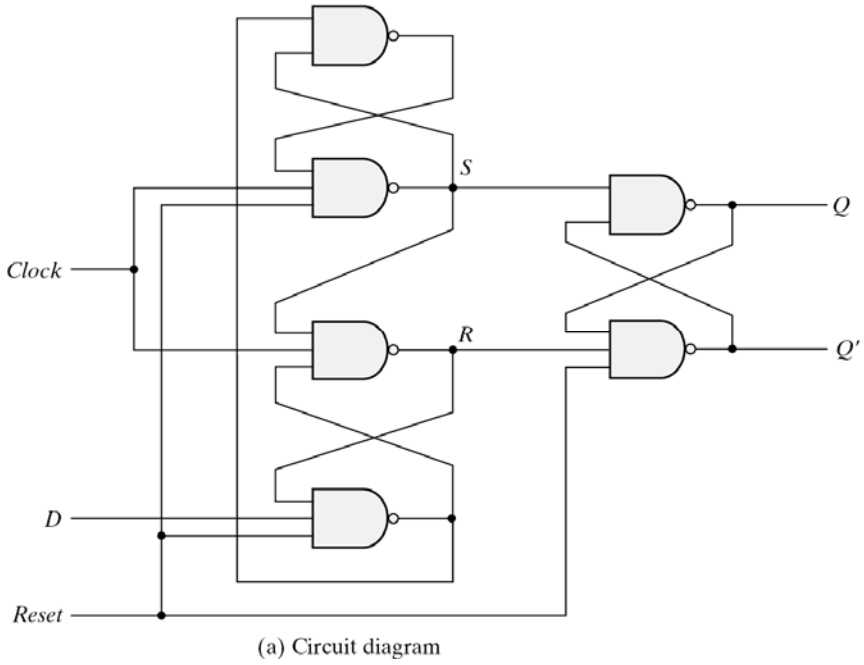
T	Q(t + 1)	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

Direct Inputs

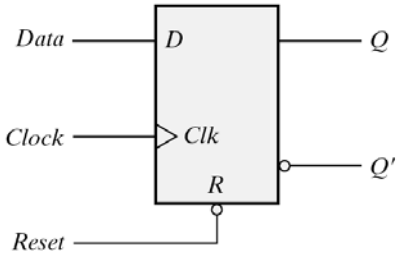
- When power is on, the state of flip-flop is unknown. The direct input will force the flip-flops in the system to a know starting state before the system starts.
- *Preset* or (*Direct Set*) sets the flip-flop to 1
- *Clear* or (*Direct Reset*) sets the flip-flop to 0

Asynchronous Set/Reset

- D Flip-Flop with asynchronous reset



(a) Circuit diagram



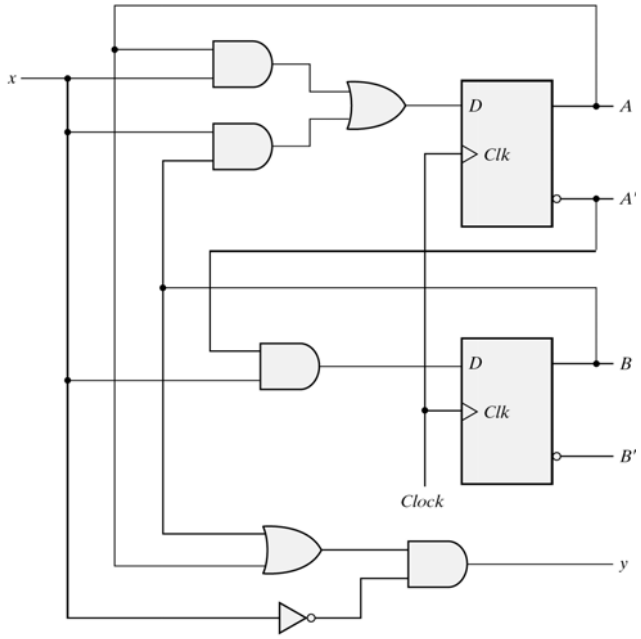
(b) Graphic symbol

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

(b) Function table

Analysis of Clocked Sequential Circuit

- A sequential circuit
  - (inputs, current state) ⇒ (output, next state)
  - a state transition table or state transition diagram
- An example :



State Equations

- A *state equation* (*transition equation*) specifies the next state as a function of the present state and input.
- State equation of an example in the previous slide.
  - $A(t+1) = A(t)x(t) + B(t)x(t)$
  - $B(t+1) = A'(t)x(t)$
- A compact form
  - $A(t+1) = Ax + Bx$
  - $B(t+1) = A'x$
- The output equation
  - $y(t) = (A(t) + B(t))x'(t)$
  - $y = (A + B)x'$

State Table

- A *state table* (*transition table*) enumerates the time sequence of inputs, outputs, and flip-flop state.
- State table consists of four sections labeled
  - *present state*
  - *input*
  - *next state*
  - *output*

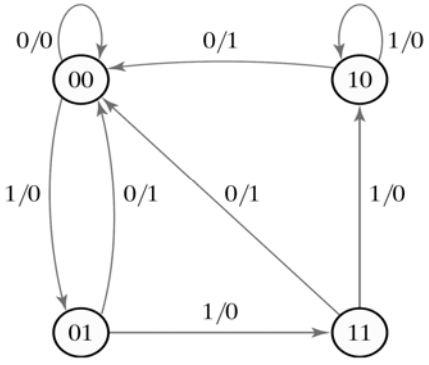
Table 5.2
State Table for the Circuit of Fig. 5.15

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 Diagram

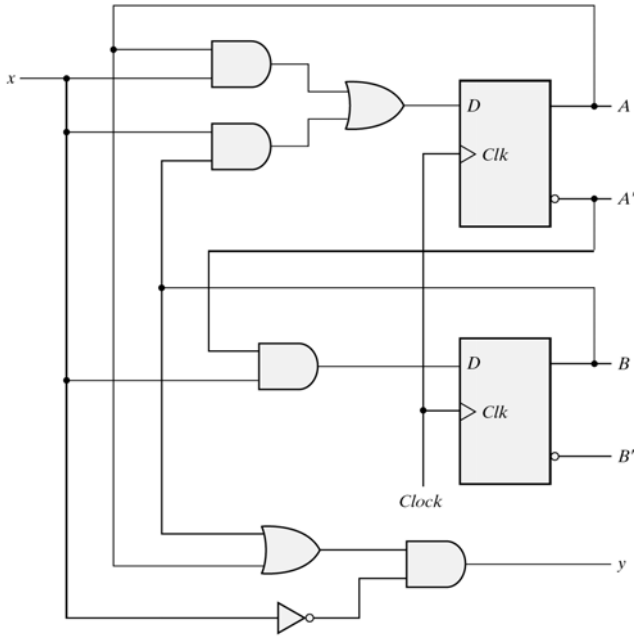
- State transition diagram
  - a circle: a state
  - a directed lines connecting the circles: the transition between the states
- Each directed line is labeled 'inputs/outputs'
  - a logic diagram ⇔ a state table ⇔ a state diagram
- Derive the state diagram from state table

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



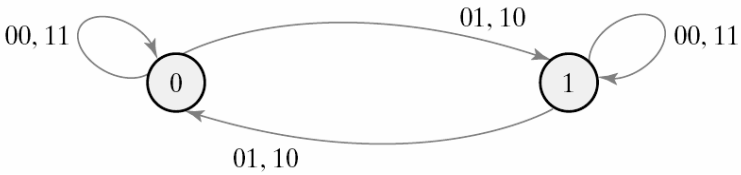
Flip-Flop Input Equations

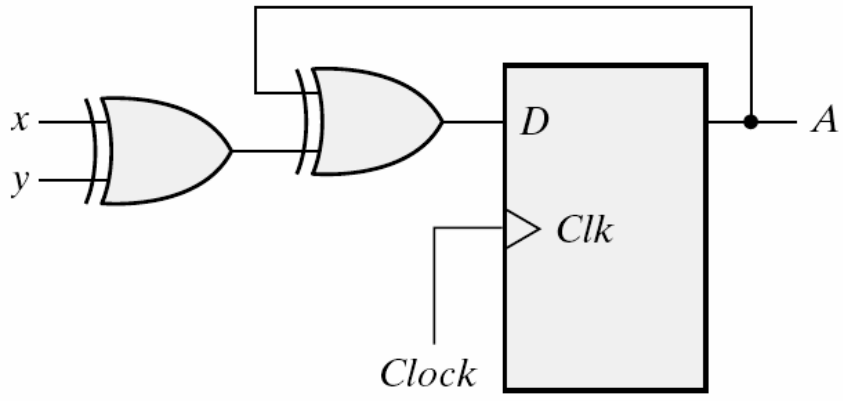
- Flip-flop *input equations* (*excitation equations*) describes the part of circuit that generates the inputs to flip-flops
  - $D_A = Ax + Bx$ (1)
  - $D_B = A'x$ (2)
- The *output equations*
  - $y = (A + B)x'$ (3)
- *Input equations* and *output equations* provide the necessary information for drawing the logic diagram of the sequential circuit.
- The *input equations* are identical to the corresponding *state equations* only for D Flip-flop.



Analysis with D flip-flops

- The input equation
  - $D_A = A \oplus x \oplus y$
- The state equation
  - $A(t + 1) = A \oplus x \oplus y$





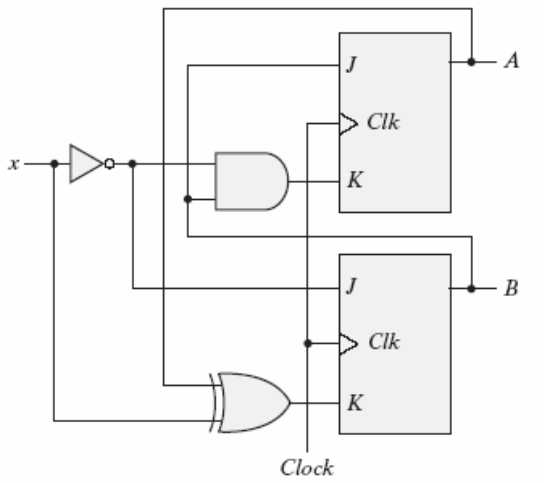
Present state	Inputs		Next state
A	x	y	A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Analysis with JK/T flip-flops

- For JK flip-flop and T flip-flop, the input equations and state equations are not the same.
  - 1. Determine the flip-flop input function in terms of the present states and input variables.
  - 2. List the binary values of each input equations.
  - 3. Use the corresponding flip-flop characteristic table to determine the next state.
  - 4. Compute output signals.
  - 5. Draw State Diagram.
- An example: JK Flip-Flop circuit

  Note:

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



Analysis with JK flip-flops

- Input equations of the circuit
  - $J_A = B$, $K_A = Bx'$ (1)
  - $J_B = x'$, $K_B = A'x + Ax'$ (2)
- State table
- Or, derive the state equations using characteristic equations of the flip-flop.

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

Table 5.4
State Table for Sequential Circuit with JK Flip-Flops

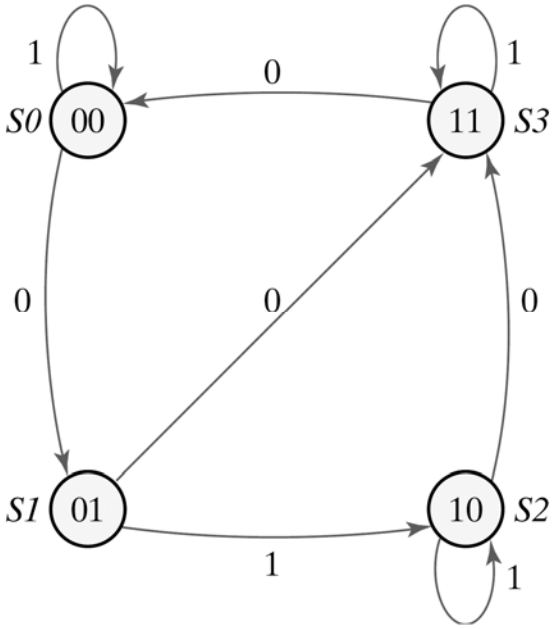
Present State		Input	Next State		Flip-Flop Inputs			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

Analysis with JK flip-flops

- Characteristic equation of JK flip-flops
  - $Q(t + 1) = JQ' + K'Q$ (3)
- Evaluation of the state equations from the characteristic equations
  - JK flip-flop A : $A(t + 1) = J_A A' + K_A' A$ (4)
  - JK flip-flop B : $B(t + 1) = J_B B' + K_B' B$ (5)
- Substituting the values of J_A, K_A, J_B, and K_B from the input equations (1)(2) to get the state equations of the circuit
  - $A(t + 1) = BA' + (Bx')'A = A'B + AB' + Ax$ (6)
  - $B(t + 1) = x'B' + (A \oplus x)'B = B'x' + ABx + A'Bx'$ (7)

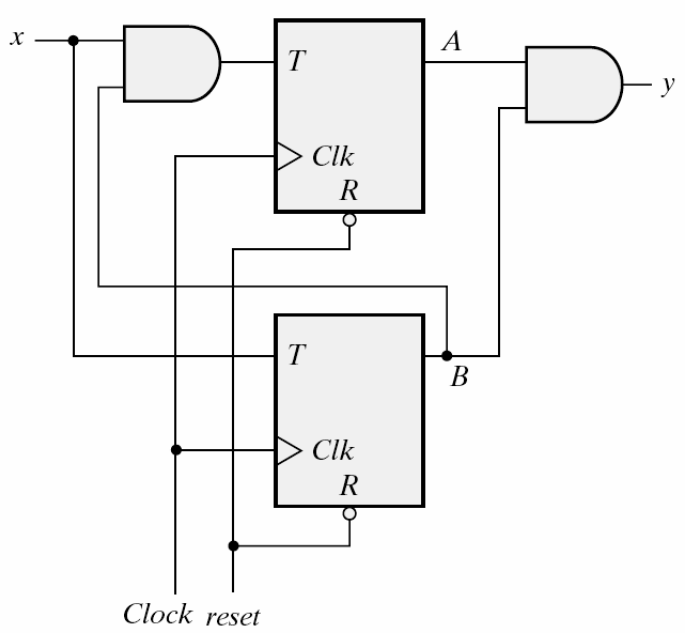
Analysis with JK flip-flops

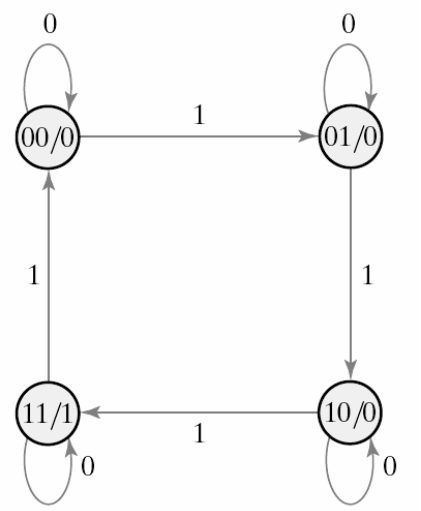
- Using the state equation or state table (preferred) of the circuit to draw the state transition diagram.



Analysis with T flip-flops

- The characteristic equation of the T flip-flops
  - $Q(t + 1) = T \oplus Q = TQ' + T'Q$





Analysis with T flip-flops

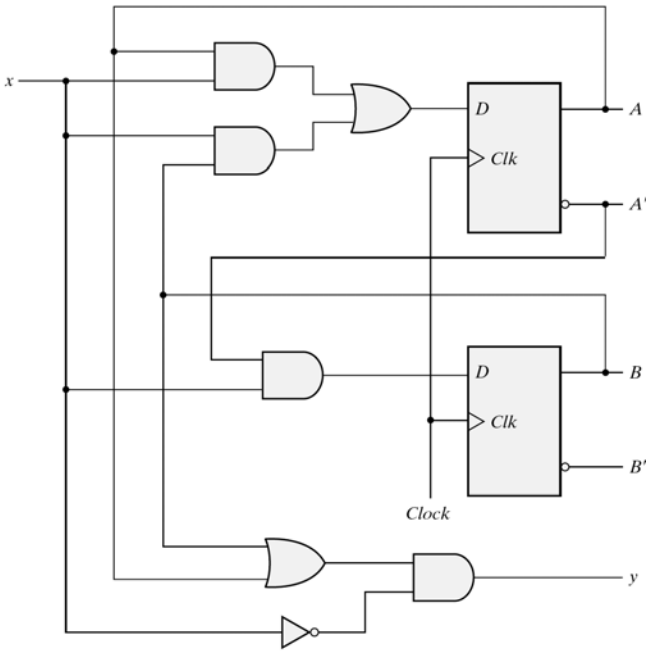
- The input and output functions
  - $T_A = Bx$
  - $T_B = x$
  - $y = AB$
- The state equations
  - $A(t + 1) = (Bx)'A + (Bx)A'$
    $= AB' + Ax' + A'Bx$
  - $B(t + 1) = x \oplus B$

State Table for Sequential Circuit with T Flip-Flops

Present State		Input			Next State		Output
A	B	x	T_A	T_B	A	B	y
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	0	0	1	0
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	0
1	0	1	0	1	1	1	0
1	1	0	0	0	1	1	1
1	1	1	1	1	0	0	1

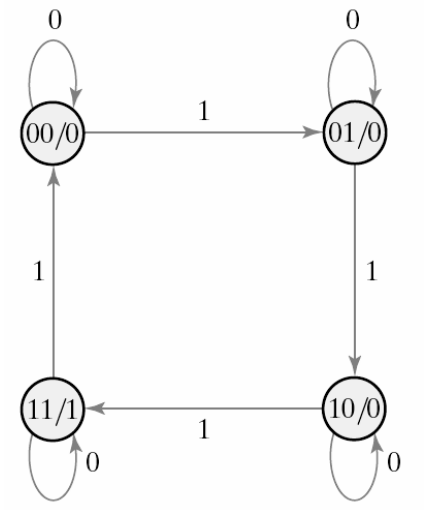
Mealy Model

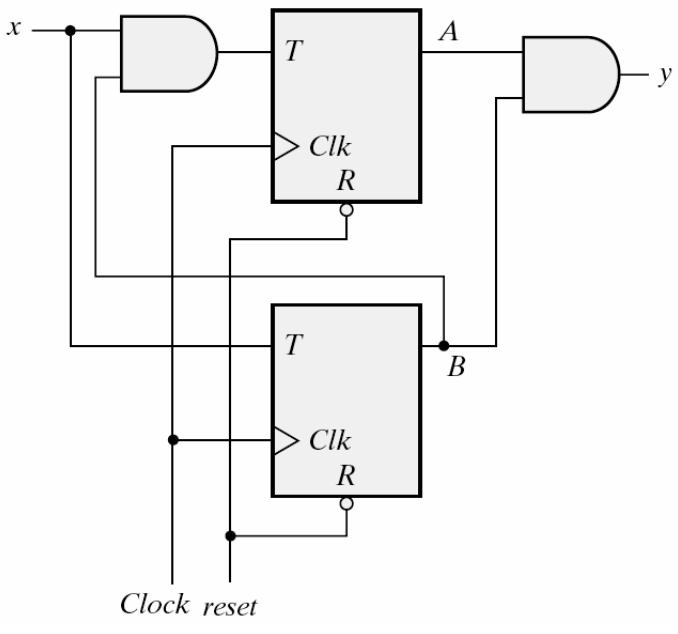
- *Mealy model*: the outputs are functions of both the present states and inputs
  - the outputs may change if the inputs change during the clock pulse period
    - the outputs may have momentary false values unless the inputs are synchronized with the clocks
- An example:



Moore Model

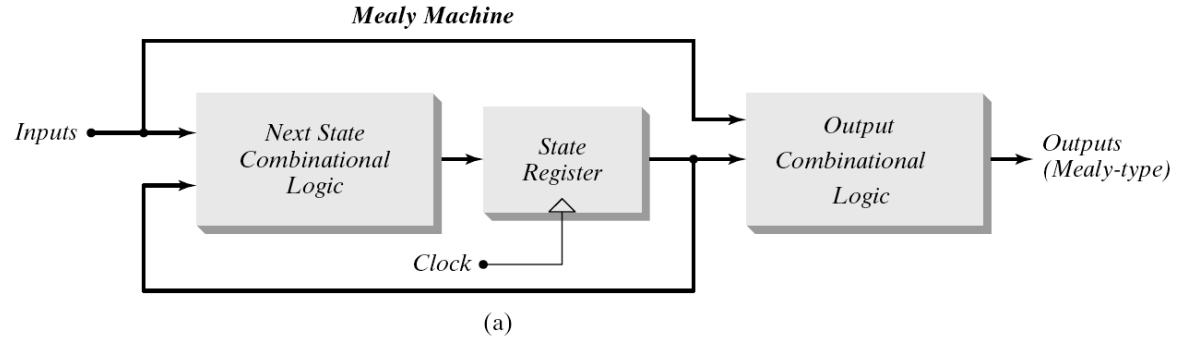
- *Moore model*: the outputs are functions of the present states only. The outputs are synchronous with the clock.
- An example:





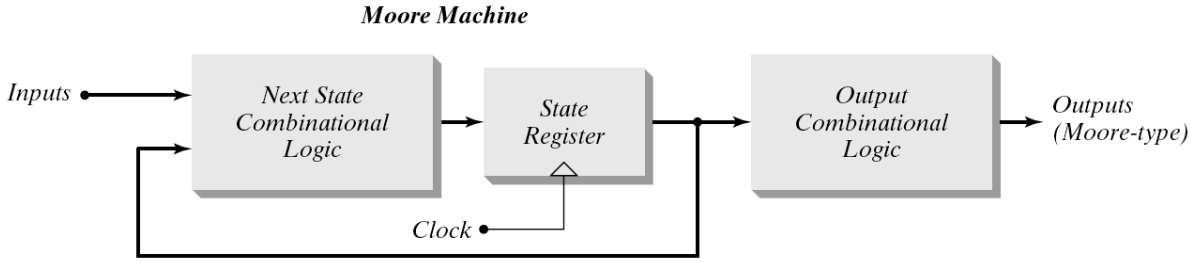
Moore Model v.s Mealy Model

- Mealy machine



(a)

- Moore machine



(b)

Synthesizable HDL Models of Sequential Circuits

- Behavioral Modeling
- Example: Two ways to provide free-running clock

```
initial
  begin
    clock = 1'b0;
    repeat (30)
      #10 clock = ~clock;
  end
```

```
initial
  begin
    clock = 1'b0;
  end

initial 300 $finish;
always #10 clock = ~clock;
```

- Example: Another way to describe free-running clock

```
initial begin clock = 0; forever #10 clock = ~clock; end
```

Behavioral Modeling

- **always** statement

```
always @ (event control expression) begin
    // Procedural assignment statements that execute when the condition is met
end
```

- Examples:

```
always @ (A or B or C)
```

```
always @(posedge clock or negedge reset)        // Verilog 1995
```

```
always @(posedge clock, negedge reset)          // Verilog 2001, 2005
```

Two procedural blocking assignments:

```
B = A
C = B + 1
```

Two nonblocking assignments:

```
B <= A
C <= B + 1
```


Procedural Assignment

- Procedural assignments update the values of registers under the control of two kinds of procedure assignment .
- Blocking procedure assignments are executed sequentially in the order they are listed.
- Non-Blocking procedure assignments evaluate the expressions on the right-hand side, but do not make the assignment to the left-hand side until all expressions are evaluated.

```
initial begin
  A=1;
  B=2;
  C=0;
end
always c = #5 ~c;
```

```
always @(posedge c)
begin
  B=A;     // B=1
  C=B+1;   // C=2
end
```

Blocking

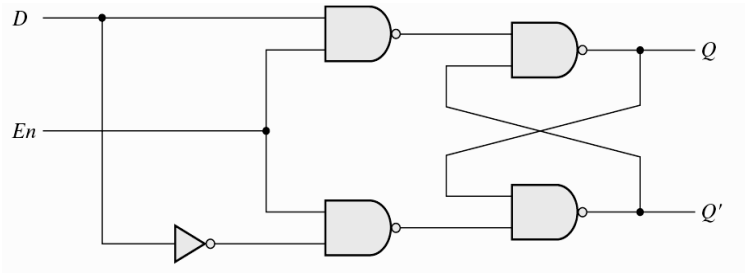
```
always @(posedge c)
begin
  B<=A;     // B=1
  C<=B+1;   // C=3
end
```

Non-Blocking

Flip-Flops and Latches

- HDL Example 5.1

```
// Description of D latch (transparent latch)
// See Fig. 5-6

module D_latch (Q, D, enable);
  output          Q;
  input           D, enable;
  reg             Q;

  always @ (enable or D)
    if (enable) Q <= D;      // Alternative: if (enable == 1) Q <= D;
endmodule

// Alternative syntax (Verilog 2001, 2005)
module D_latch (output reg Q, input enable, D);
  always @ (enable, D)
    if (enable) Q <= D;      // No action if enable is not asserted
endmodule
```



Flip-Flops and Latches

- HDL Example 5.2

```
// Description of D flip-flop without reset (See Fig. 5-11)

module D_FF (Q, D, Clk);
  output          Q;
  input           D, Clk;
  reg             Q;
  always @ (posedge Clk)
    Q <= D;
endmodule

// Description of D flip-flop with asynchronous reset, Verilog 2001, 2005

module DFF (output reg Q, input D, Clk, rst);
  always @ (posedge Clk, negedge rst)
    if (~rst) Q <= 1'b0;              // same as: if (rst == 0) Q <= 1'b0;
    else Q <= D;
endmodule
```

Characteristic Equation

$Q(t + 1) = Q \oplus T$ ⟹ For a *T* flip-flop

$Q(t + 1) = JQ' + K'Q$ ⟹ For a *JK* flip-flop

- HDL Example 5.3

```
// T flip-flop from D flip-flop and gates
module TFF (output Q, input T, Clk, rst);
  wire DT;
  assign DT = Q ^ T;
// Instantiate the D flip-flop
  DFF TF1 (Q, DT, Clk, rst);
endmodule

// JK flip-flop from D flip-flop and gates
module JKFF (output reg Q, input J, K, Clk, rst);
  wire JK;
  assign JK = (J & ~Q) | (~K & Q);
// Instantiate D flip-flop
  DFF JK1 (Q, JK, Clk, rst);
endmodule

// D flip-flop (V2001, V2005)
module DFF (output reg Q, input D, Clk, rst);
  always @ (posedge Clk, negedge rst)
    if (~rst) Q <= 1'b0;
    else Q <= D;
endmodule
```


HDL Example 5-4

```
// Function description of JK flip-flop (V2001, V2005)
module JK_FF (input J, K, Clk, output reg Q, output Q_b);
  assign Q_b = ~Q;
  always @ (posedge Clk)
    case ({J, K})
      2'b00: Q <= Q;          // No change
      2'b01: Q <= 1'b0;       // Set
      2'b10: Q <= 1'b1;       // Reset
      2'b11: Q <= ~Q;         // Complement
    endcasae
endmodule
```

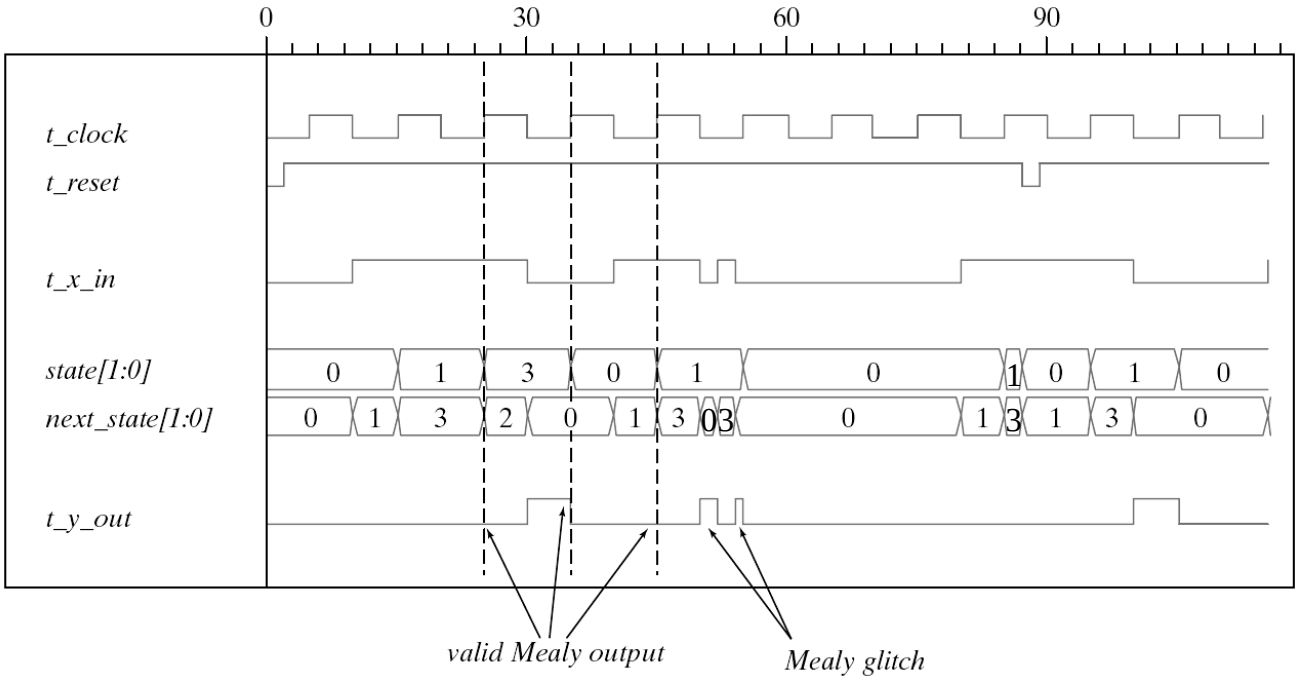
HDL Example 5.5: Mealy HDL model

```
// Mealy FSM zero detector
module Mealy_Zero_Detector (
  output reg      y_out,
  input           x_in, clock, reset
);
  reg [1: 0]      state, next_state;
  parameter       S0 = 2'b00, S1 =
2'b01, S2 = 2'b10, S3 = 2'b11;

  always @ (posedge clock, negedge
reset)     // state transition
    if (reset == 0) state <= S0;
    else state <= next_state;

  always @ (state, x_in) // Form the
next state
    case (state)
      S0:  if (x_in)   next_state = S1;
           else next_state = S0;
      S1:  if (x_in)  next_state = S3;
           else next_state = S0;
      S2:  if (~x_in) next_state = S0;
           else next_state = S2;
      S3: if (x_in)   next_state = S2;
           else next_state = S0;
    endcase

  always @ (state, x_in) // Form the
output
    case (state)
      S0:  y_out = 0;
      S1, S2, S3: y_out = ~x_in;
    endcase
endmodule
```


HDL Example 5.5 (Cont'd)

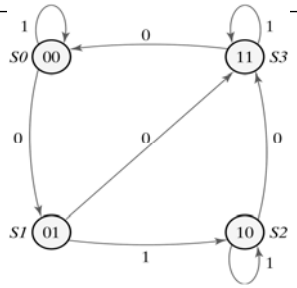
```
module t_Mealy_Zero_Detector;
  wire    t_y_out;
  reg     t_x_in, t_clock, t_reset;

Mealy_Zero_Detector M0 (t_y_out,
t_x_in, t_clock, t_reset);

initial #200 $finish;
initial begin t_clock = 0; forever #5
t_clock = ~t_clock; end

initial fork
      t_reset = 0;
  #2 t_reset = 1;
  #87 t_reset = 0;
  #89 t_reset = 1;
  #10 t_x_in = 1;
  #30 t_x_in = 0;
  #40 t_x_in = 1;
  #50 t_x_in = 0;
  #52 t_x_in = 1;
  #54 t_x_in = 0;
  #80 t_x_in = 1;
  #100 t_x_in = 0;
  #120 t_x_in = 1;
  #160 t_x_in = 0;
  #170 t_x_in = 1;
  join
endmodule
```

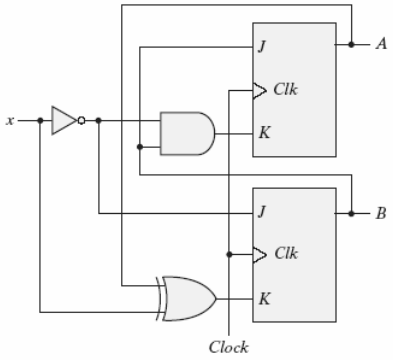
Mealy_Zero_Detector



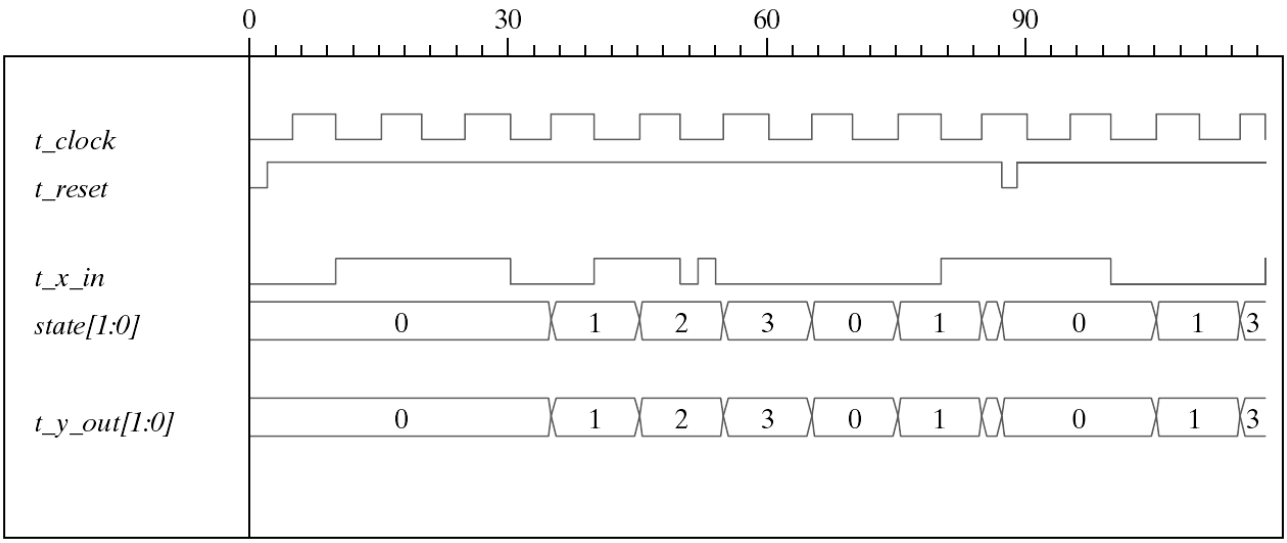
HDL Example 5-6: Moore Model FSM

```
// Moore FSM (See state diagram in Fig. 5-19)

module Moore_Model_Fig_5_19 (
  output [1: 0] y_out,
  input  x_in, clock, reset
);
  reg [1: 0]        state;
  parameter         S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;

  always @ (posedge clock, negedge reset)
    if (reset == 0) state <= S0;       // Initialize to state S0
    else case (state)
      S0:  if (~x_in)   state <= S1; else state <= S0;
      S1:  if (x_in)    state <= S2; else state <= S3;
      S2:  if (~x_in)   state <= S3; else state <= S2;
      S3:  if (~x_in)   state <= S0; else state <= S3;
    endcase

  assign y_out = state;     // Output of flip-flops
endmodule
```



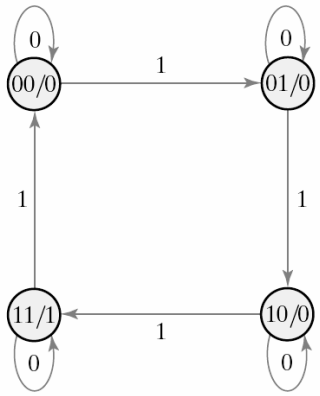


Simulation Output of HDL Example 5-6



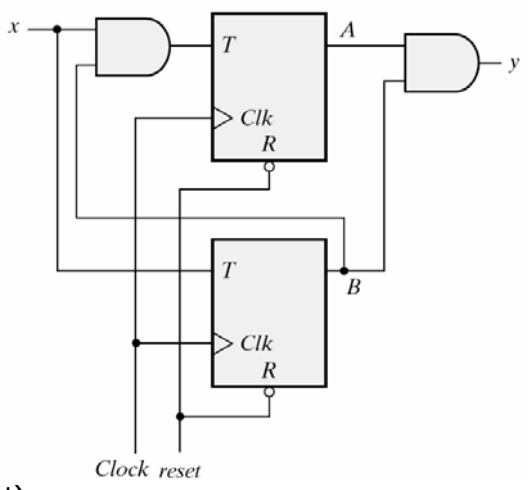
Structural Description of Clocked Sequential Circuits

```
module Moore_Model_Fig_5_20 (
  output          y_out,
  input           x_in, clock, reset
);
  reg [1: 0]      state;
  parameter       S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;

  always @ (posedge clock, negedge reset)
    if (reset == 0) state <= S0;       // Initialize to state S0
    else case (state)
      S0:  if (x_in) state <= S1; else state <= S0;
      S1:  if (x_in) state <= S2; else state <= S1;
      S2:  if (x_in) state <= S3; else state <= S2;
      S3:  if (x_in) state <= S0; else state <= S3;
    endcase

  assign y_out = (state == S3);     // Output of flip-flops
endmodule
```



HDL Example 5-7 (Cont'd)

```
module Moore_Model_STR_Fig_5_20 (
  output          y_out, A, B,
  input           x_in, clock, reset
);
  wire            TA, TB;

// Flip-flop input equations
  assign TA = x_in & B;
  assign TB = x_in;
//output equation
  assign y_out = A & B;
// Instantiate Toggle flip-flops
  Toggle_flip_flop_3 M_A (A, TA, clock, reset);
  Toggle_flip_flop_3 M_B (B, TB, clock, reset);

endmodule

module t_Moore_Fig_5_20;
  wire    t_y_out_2, t_y_out_1;
  reg     t_x_in, t_clock, t_reset;
```



HDL Example 5-7 (Cont'd)

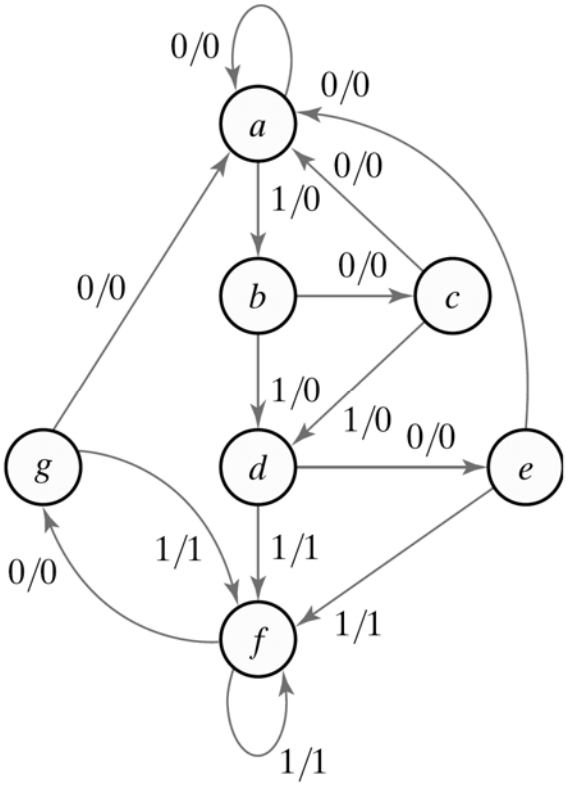
```
Moore_Model_Fig_5_20          M1(t_y_out_1, t_x_in, t_clock, t_reset);
Moore_Model_STR_Fig_5_20   M2 (t_y_out_2, A, B, t_x_in, t_clock, t_reset);
```

```
initial #200 $finish;
 initial begin
        t_reset = 0;
        t_clock = 0;
   #5 t_reset = 1;
   repeat (16)
     #5 t_clock = ~t_clock;
 end
```

```
 initial begin
          t_x_in = 0;
   #15 t_x_in = 1;
   repeat (8)
     #10 t_x_in = ~t_x_in;
 end
endmodule
```

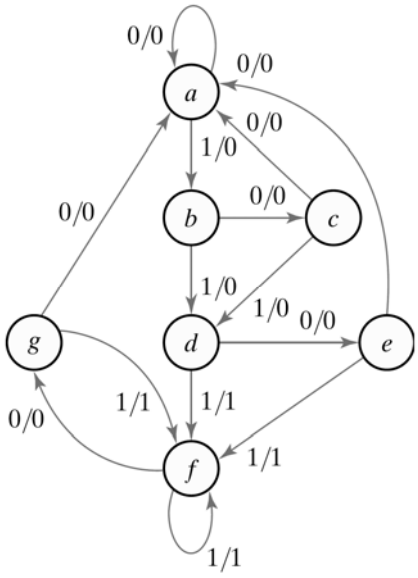
State Reduction and Assignment

- m flip-flops produce 2^m states.
- State Reduction
  - may result in a reduction in the number of flip-flops and the number of gates
- An example of state diagram:
- Note: different input sequence results in different output sequence.



State Reduction

- Input/Output sequences are the most important!
  - Each input of 0 or 1 produces an output of 0 or 1 and causes the circuit goes to the next state.
  - state a a b c d e f f g f g a
    input 0 1 0 1 0 1 1 0 1 0 0
    output 0 0 0 0 0 1 1 0 1 0 0
  - Only the input-output sequences are important.
  - So, two circuits are equivalent if they have identical outputs for all input sequences.
  - Note: the number of states is not important.



Equivalent States

- Two states are said to be equivalent:
  - 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.
  - When two states are equivalent, one of them can be removed.

	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	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

The Reduced Finite State Machine

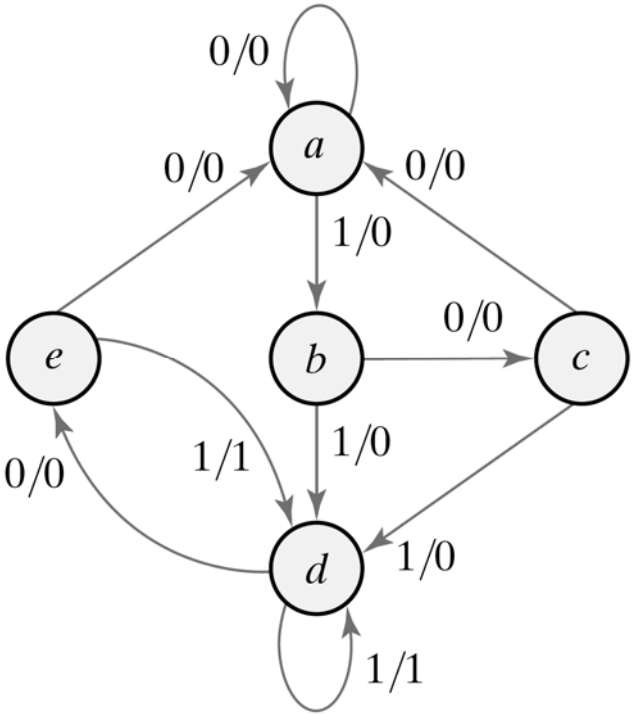
	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

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

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

The Reduced Finite State Machine

- The checking of each pair of states for possible equivalence can be done systematically (in Chap. 9-5)
- The unused states are treated as don't-care conditions .
- The more unused states
  $\Rightarrow$ the more don't care conditions
  $\Rightarrow$ the fewer combinational gates are required.



State Assignment

- For a circuit with m states, we need at least n-bit, where $2^n \geq m$, to encode the states.
- Three possible binary state assignments for the previous example:

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

- Any binary number assignment is satisfactory as long as each state is assigned a unique number
- The state table uses binary assignment 1:

	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

Design Procedure

- The design of synchronous sequential circuit consists of
  - Choosing flip-flops
    - Depending on the number of states
  - Finding the combinational logic
    - Evaluation of input equations and output equations
- Recommended steps of the design procedures for synchronous sequential circuits
  - 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
  - derive the simplified flip-flop input equations and output equations
  - draw the logic diagram

Synthesis Using D Flip-Flops

- An example: state diagram and state table

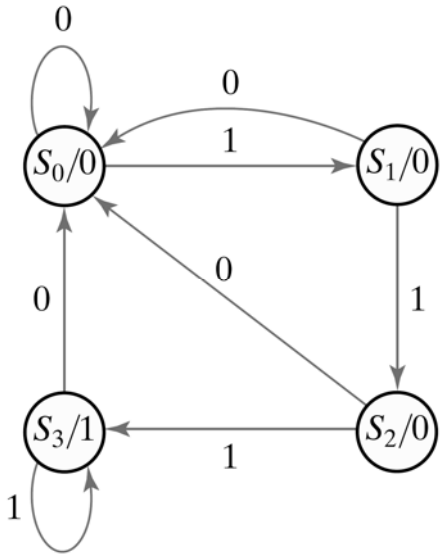


Table 5.11
State Table for Sequence Detector

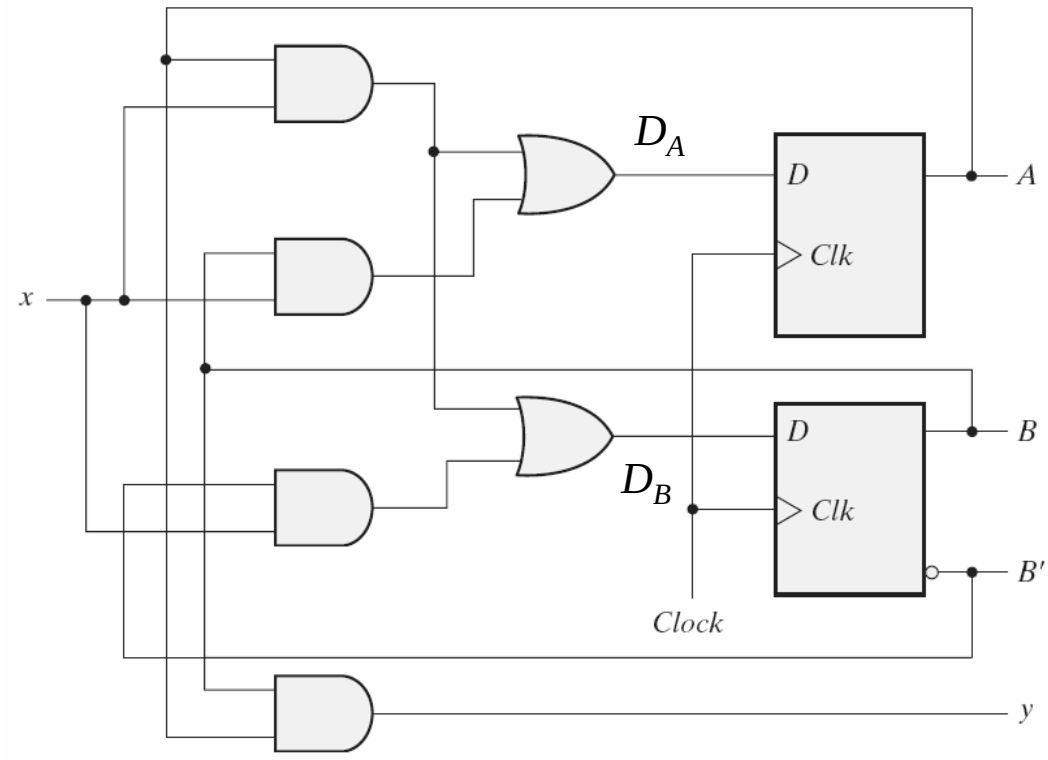
Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

Synthesis Using D Flip-Flops

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

Sequence Detector

- The logic diagram:



Excitation Tables

- A state diagram ⇒ flip-flop input functions
  - straightforward for D flip-flops
  - we need excitation tables for JK and T flip-flops

Table 5.12
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*

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

(b) *T*

Synthesis Using JK Flip-Flops

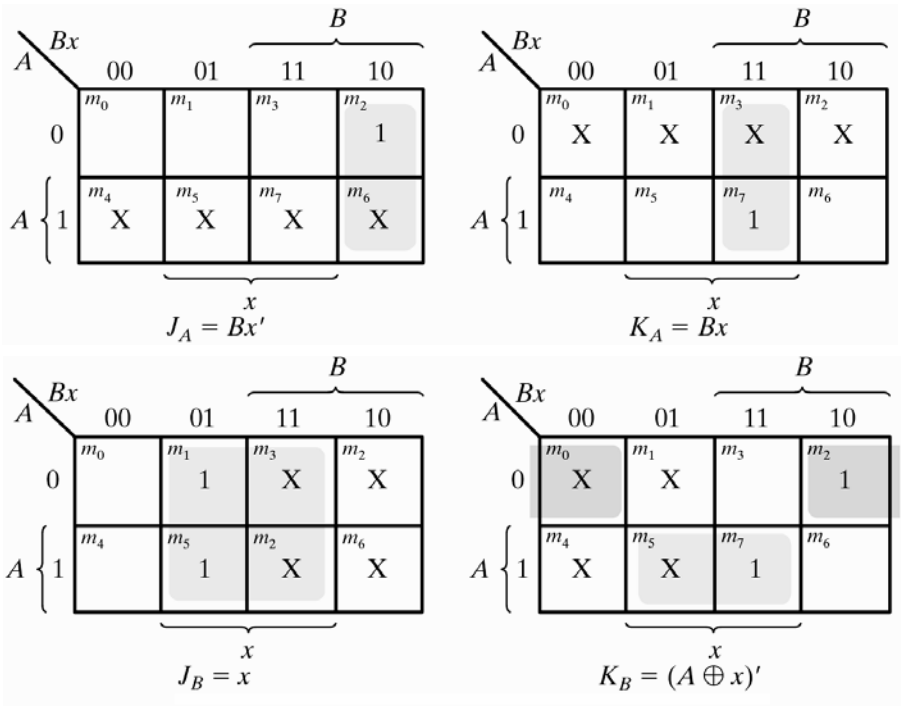
- The synthesis procedure is the same as DFF.
- The state table and JK flip-flop inputs

Table 5.13
State Table and JK Flip-Flop Inputs

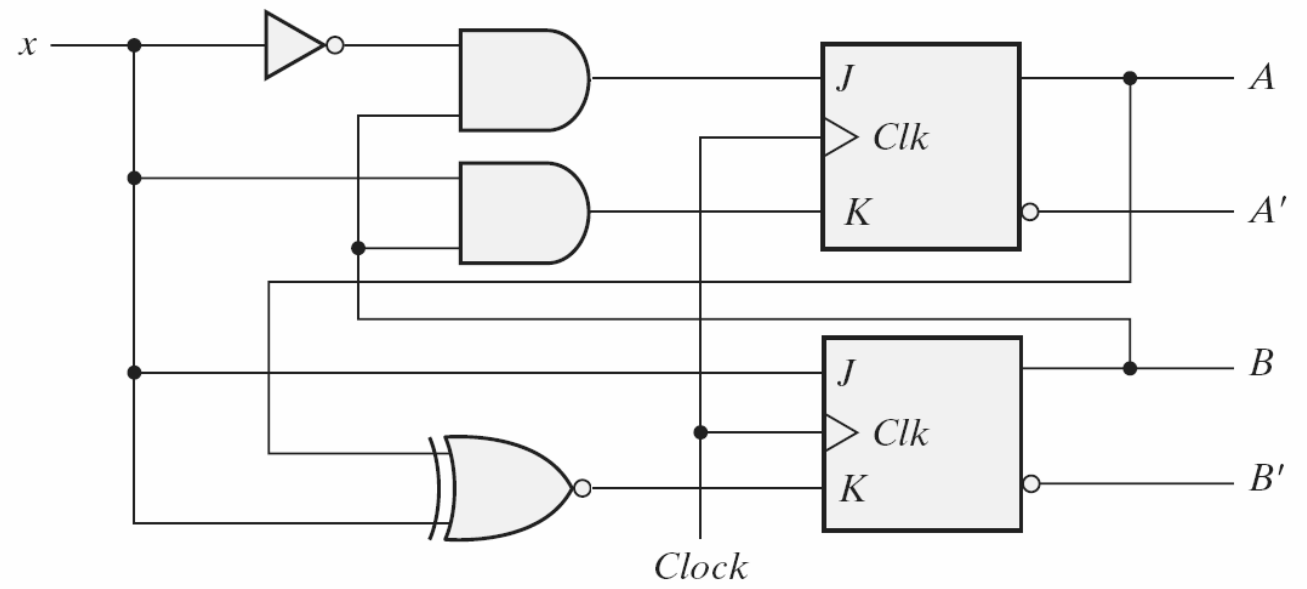
Present State		Input	Next State		Flip-Flop Inputs			
A	**B**	**x**	**A**	**B**	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Maps for JK Input Equations

– $J_A = Bx'$; $K_A = Bx$ – $J_B = x$; $K_B = (A \oplus x)'$



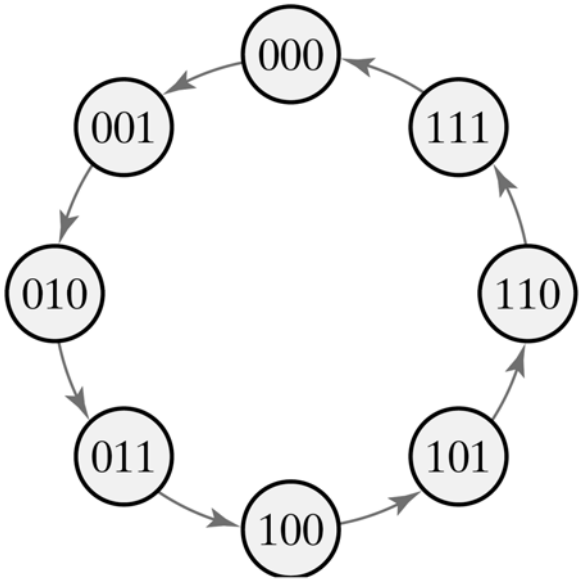
Logic Diagram for Sequential Circuit with JK Flip-Flops



Synthesis Using T flip-flops

- An n-bit binary counter
  - the state diagram
  - no inputs (except for the clock input)

State diagram of three-bit binary counter

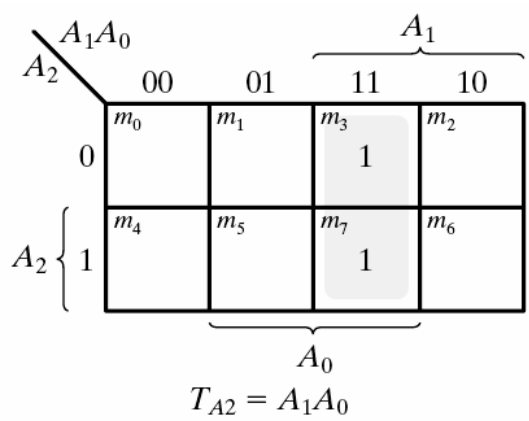


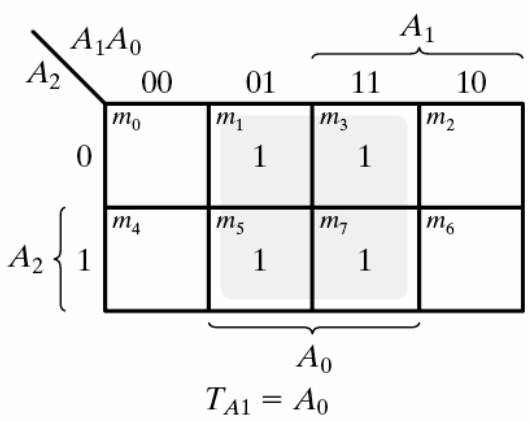
The State Table and The Flip-Flop Inputs

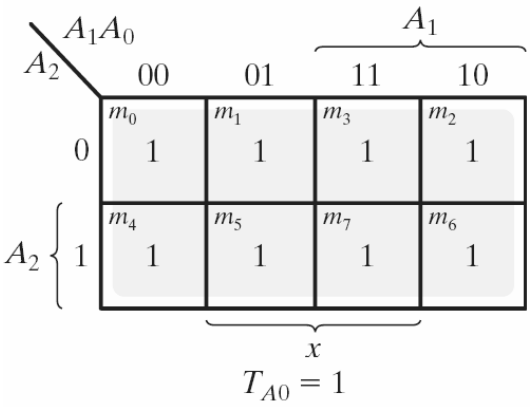
Table 5.14
State Table for Three-Bit Counter

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

Maps of Three-Bit Binary Counter







Logic Simplification Using The K Map

- $T_{A2} = A_1A_2$
- $T_{A1} = A_0$
- $T_{A0} = 1$

- The logic diagram