Chapter 5
Synchronous Sequential Logic
OUTLINE OF CHAPTER 5

Sequential Circuits  Latches  Flip-flop  Analysis of Clocked Sequential Circuits  State Reduction and Assignment

Design Procedure
5.1 SEQUENTIAL CIRCUITS
• Every digital system is likely to have combinational circuits.
• Most systems encountered in practice also include storage elements, which require that the system be described in terms of sequential logic.
• The storage elements are devices capable of storing binary information.
• The binary information stored in these elements at any given time defines the *state* of the sequential circuit at that time.
• The sequential circuit receives binary information from external inputs.
• These inputs, together with the present state of the storage elements, determine the binary value of the outputs.
They also determine the condition for changing the state in the storage elements.

A sequential circuit is specified by a time sequence of inputs, output, and internal states.
There are two main types of sequential circuits.
Their classification depends on the timing of their signals.
• **Asynchronous** Sequential Circuit

---

- The behaviour of the circuit depends upon the input signals at any instant of time and the order in which the inputs change.
• **Asynchronous** Sequential Circuit

  – In gate – type asynchronous systems, the storage elements consist of logic gates whose propagation delay provides the required storage.

  – Thus, an asynchronous sequential circuit may be regarded as a combinational circuit with feedback.

  – Because of the feedback among logic gates, an asynchronous sequential circuit may become unstable at times.
• **Synchronous** Sequential Circuit

  - The behaviour can be defined from the knowledge of its signals at discrete instants of time.
SEQUENTIAL CIRCUITS

• **Synchronous** Sequential Circuit
  – Employs signals that affect the storage elements only at discrete instants of time.
  – Synchronisation is achieved by a timing device called a **clock generator**.
    • Provides a periodic train of **clock pulses**.
    • Clock pulses are distributed throughout the system in such a way that storage elements are affected only with the arrival of each pulse.
• **Synchronous** Sequential Circuit
  - In practice, the clock pulses are applied with other signals that specify the required change in the storage elements.
    - Circuits that use clock pulses in the inputs of storage elements are called *clocked sequential circuits*.
    - The storage elements used in clocked sequential circuits are called *flip–flops*.
    - A flip–flop is a binary storage device capable of storing one bit of information.
5.2 LATCHES
LATCHES

- **Latches** are the basic circuits from which all flip-flops are constructed.
- Although latches are useful for storing binary information and for the design of **asynchronous** sequential circuits.
- They are **not practical** for use in synchronous sequential circuits.
LATCHES

- **SR Latch**

![SR Latch Diagram]

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Set State
Hold State
Reset State
Hold State
Invalid State
LATCHES

- **SR Latch**

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q₀</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Q=Q'₀=0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q=Q'₁=1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Q₀</td>
</tr>
</tbody>
</table>
LATCHES

• SR Latch with Control Input

<table>
<thead>
<tr>
<th>C</th>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>HOLD</td>
<td>No change</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>HOLD</td>
<td>No change</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Q = 0</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Q = 1</td>
<td>Set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Q = Q’</td>
<td>Invalid</td>
</tr>
</tbody>
</table>
LATCHES

• **D Latch (D = Data)**
  
  – One way to eliminate the undesirable condition of the indeterminate state in the SR latch is to ensure that inputs S and R are never equal to 1 at the same time.
  
  – D latch has two inputs
    
    • D (data) - directly goes to the S input and its complement is applied to the R input.
    
    • C (control)
LATCHES

• **D Latch (D = Data)**

![D Latch Diagram]

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>HOLD</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Q = 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Q = 1</td>
</tr>
</tbody>
</table>

**Timing Diagram**

- Output may change
- Reset
- Set
- No change
LATCHES

- **D Latch (D = Data)**

```
+-------+-------+-------+
<table>
<thead>
<tr>
<th>D</th>
<th>S</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
+-------+-------+-------+
<table>
<thead>
<tr>
<th>C</th>
<th>R</th>
<th>Q̅</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
+-------+-------+-------+
```

Timing Diagram

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>HOLD</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Q = 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Q = 1</td>
</tr>
</tbody>
</table>

No change
Reset
Set

Output may change
LATCHES

• **D Latch (D = Data)**
  
  – The D latch has an ability to hold data in its internal storage.
  – It is suited for use as a temporary storage for binary information.
  – This circuit is often called **transparent** latch.
    
    • The output follow changes in the data input as long as the control input is **enabled**.
5.3 FLIP – FLOPS
Flip – flops are constructed in such a way to make D latches operate properly when they are part of a sequential circuit that employs a common clock.

The problem with the latch is that
- It responds to a change in the level of a clock pulse.
  - Positive level response in the control input allows changes, in the output when the D input changes while the control pulse stays at logic 1.

The key to the proper operation of a flip – flop is
- To trigger it only during a signal transition.
• Controlled latches are **level** – triggered

• Flip-Flops are **edge** – triggered

**CLK**

Positive Edge

Negative Edge
FLIP – FLOPS

• There are two ways that a latch can be modified to form a flip – flop.

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

2. Produce a flip – flop that triggers only during a signal transition.
   • From 0 to 1 or from 1 to 0 only.
   • Disabled during the rest of the clock pulse duration.
FLIP – FLOPS

- Master – Slave D flip – flops

Looks like it is negative edge-triggered
**FLIP – FLOPS**

- **Edge-Triggered D Flip – Flop**
- **Two latches respond to the external D (data) and CLK (clock inputs).**
- **Third latch provides the outputs for the flip – flop.**

![Diagram of Edge-Triggered D Flip – Flop]
I. When CLK = 0, S = 1 and R = 1. Output = present state.

II. If D = 0, when CLK \( \rightarrow \) 1
1. R changes to 0
2. Flip – flop goes to the RESET state.
3. Q = 0.
FLIP – FLOPS

Edge-Triggered D Flip – Flop

III. If D changes when CLK = 1 then
   1. R remains at 0.
   2. Flip – flop is locked out
   3. Unresponsive to further changes in the input.

IV. When CLK → 0, 
   1. R → 1
   2. Placing the output latch in the quiescent condition.
   3. No change in the output.
• **Edge-Triggered D Flip – Flop**

V. **If $D = 1$ when $CLK = 0 \rightarrow 1$,**

1. $S$ changes to 0.
2. Circuit goes to SET state
3. $Q = 1$.
4. Any change in $D$ while $CLK = 1$ does not affect the output.
When CLK in the positive-edge-triggered flip – flop
- Makes positive transition
  - The value of D is transferred to Q.
- Makes negative transition
  - Does not affect the output.
- Steady CLK 1 or 0
  - Does not affect the output.
Edge-Triggered D Flip – Flop

- The timing of the response of a flip–flop to input data and clock must be taken into consideration when using edge–triggered flip-flops.
  - There is a minimum time, called setup time, for which the D input must be maintained at a constant value prior to the occurrence of the clock transition.
  - There is a minimum time, called hold time, for which the D input must not change after the application of the positive transition of the clock.
FLIP – FLOPS

• Edge-Triggered D Flip – Flop

---

Positive Edge

Dynamic input

Negative Edge
• The most economical and efficient flip – flop constructed is the edge – triggered D flip – flop.
  – It requires smallest number of gates.
• Other types of flip – flops can be constructed by using the D flip – flop and external logic.
  – JK flip – flops
  – T flip - flops
There are three operations that can be performed with a flip–flop:
- Set it to 1
- Reset it to 0
- Complement its output
**FLIP – FLOPS**

- **JK Flip – Flop**
  - Performs all three operations.
  - When J = 1, sets the flip – flop to 1.
  - When K = 1, resets the flip – flop to 0.

\[ D = JQ' + K'Q \]
• JK Flip – Flop

D = JQ' + K'Q

Operation 1
• When J = 1 and K = 0,
  – D = 1.Q' + 1.Q  (Post2b)
  – D = Q' + Q   (Post5a)
  – D = 1
  – Next clock edge sets the output to 1.
• JK Flip – Flop

\[ D = JQ' + K'Q \]

**Operation 2**

• When \( J = 0 \) and \( K = 1 \),
  
  – \( D = 0.Q' + 0.Q \)  
  
  (Theo2b)
  
  – \( D = 0 + 0 \)
  
  – \( D = 0 \)
  
  – Next clock edge sets the output to 0.
• JK Flip – Flop

\[ D = JQ' + K'Q \]

Operation 3

• When \( J = 1 \) and \( K = 1 \),
  
  - \( D = 1.Q' + 0.Q \) \hspace{1cm} \text{(Post2b)}
  
  - \( D = Q' + 0 .Q \) \hspace{1cm} \text{(Theo2b)}
  
  - \( D = Q' + 0 \) \hspace{1cm} \text{(Post2a)}
  
  - \( D = Q' \)
  
  - Next clock edge complements the output.
• JK Flip – Flop

\[ D = JQ' + K'Q \]

• When \( J = 0 \) and \( K = 0 \),
  - \( D = 0.Q' + 1.Q \) (Theo2b)
  - \( D = 0 + 1 .Q \) (Post2b)
  - \( D = 0 + Q \) (Post2a)
  - \( D = Q \)
  - Next clock edge the output is unchanged.
FLIP – FLOPS

• JK Flip – Flop

\[ D = JQ' + K'Q \]
• T (toggle) Flip – Flop
  – Complementing flip – flop.
  – Can be obtained from a JK flip – flop.
  – When inputs J and K are tied together.
  – Useful for designing binary counters.
FLIP – FLOPS

- **T** (toggle) Flip – Flop
  - When $T = 0$ ($J = K = 0$)
  - A clock edge does not change the output.
  - When $T = 1$ ($J = K = 1$)
  - A clock edge complements the output.

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

$$D = TQ' + T'Q = T \oplus Q$$
**FLIP – FLOPS**

- **T (toggle) Flip – Flop**
  - Can be constructed with a D flip – flop and an XOR gate.
  - When $T = 0$ then $D = Q$
    - No change in the output.
  - When $T = 1$ then $D = Q'$
    - Output complements

\[
D = TQ' + T'Q = T \oplus Q
\]
FLIP – FLOPS

- T (toggle) Flip – Flop

(a) From JK Flip – Flop

(b) From D Flip – Flop

(c) Graphic Symbol
• Flip – Flop Characteristics Table

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

Q(t+1) = D

D  Q (t+1)
FLIP – FLOPS

- Flip – Flop Characteristics Table

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Q (t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q(t)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Q'(t)</td>
</tr>
</tbody>
</table>

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

No change
Reset
Set
Toggle
FLIP – FLOPS

• Flip – Flop Characteristics Table

\[
\begin{array}{cc}
T & Q(t+1) \\
0 & Q(t) \quad \text{No change} \\
1 & Q'(t) \quad \text{Toggle} \\
\end{array}
\]

\[Q(t+1) = T \oplus Q\]
• Some flip – flops have asynchronous inputs that are used to force the flip – flop to a particular state independent of the clock.

• The input that sets the flip – flop to 1 is called preset.

• The input that clears the flip – flop to 0 is called clear or direct reset.

• When power is on in a digital system, the state of the flip flop is unknown.
FLIP – FLOPS

• When power is on in a digital system, the state of the flip flop 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.
• Asynchronous Reset

![Flip-Flop Diagram]

<table>
<thead>
<tr>
<th>R'</th>
<th>D</th>
<th>CLK</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>x</td>
<td>0</td>
</tr>
</tbody>
</table>
• Asynchronous Reset

```
<table>
<thead>
<tr>
<th>R'</th>
<th>D</th>
<th>CLK</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
```

Reset
FLIP – FLOPS

• Asynchronous Preset and Clear

\[
\begin{array}{c}
\text{Preset} \\
\overline{\text{PR}} \\
\text{D} \\
\overline{Q} \\
\overline{\text{CLR}} \\
\text{Reset}
\end{array}
\]

<table>
<thead>
<tr>
<th>PR'</th>
<th>CLR'</th>
<th>D</th>
<th>CLK</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>0</td>
</tr>
</tbody>
</table>
• Asynchronous Preset and Clear

\[\begin{array}{cccccc}
PR' & CLR' & D & CLK & Q(t+1) \\
1 & 0 & x & x & 0 \\
0 & 1 & x & x & 1 \\
\end{array}\]
• Asynchronous Preset and Clear

**FLIP – FLOPS**

Preset

\[
\begin{align*}
PR' & \quad CLR' & \quad D & \quad CLK & \quad Q(t+1) \\
1 & \quad 0 & \quad x & \quad x & \quad 0 \\
0 & \quad 1 & \quad x & \quad x & \quad 1 \\
1 & \quad 1 & \quad 0 & \quad \uparrow & \quad 0 \\
1 & \quad 1 & \quad 1 & \quad \uparrow & \quad 1 \\
\end{align*}
\]

Reset
5.4 ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS
ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS

• The behaviour of a clocked sequential circuit is determined from:
  – The inputs
  – The outputs
  – The state of its flip-flops

• The outputs and the next state are both a function of
  – The inputs
  – The present state
The analysis of sequential circuit consists of:

- Obtaining a table or a diagram for the time sequence of
  - Inputs
  - Outputs
  - Internal states
- It is also possible to write Boolean expression that describe the behaviour of the sequential circuit.
State Equations

• The behaviour of a clocked sequential circuit can be described algebraically by means of state equations (transition equations).

• A state equation specifies the next state as a function of
  – The present state
  – Inputs
Consider:

- Circuit consists of:
  - Two D flip-flops A and B.
  - An input \( x \).
  - An output \( y \).
  - It is possible to write a set of equations for the circuit.
Consider:

- $A(t+1) = A(t) \cdot x(t) + B(t) \cdot x(t)$
- $B(t+1) = A'(t) \cdot x(t)$
  - $(t+1) \rightarrow$ next state of the flip flop
  - Right side of the equation is a Boolean expression
    - Specifies the present state
    - Input conditions that make the next state $= 1$. 
Consider:

- $A(t+1) = A(t) \cdot x(t) + B(t) \cdot x(t)$
- $B(t+1) = A'(t) \cdot x(t)$
  - Since all the variables in the Boolean expression are a function of the present state
  - We can omit the designation $(t)$

- $A(t+1) = A \cdot x + B \cdot x$
- $B(t+1) = A' \cdot x$
Consider:

- Similarly,
- \( y(t) = [A(t) + B(t)] \ x'(t) \)
- \( y = (A + B) \ x' \)
Consider:

- \( A(t+1) = A \cdot x + B \cdot x \)
- \( B(t+1) = A' \cdot x \)
- \( y = (A + B) \cdot x' \)
State Table

• The time sequence of inputs, outputs and flip – flop can be enumerated in state table (transition table).

• In general, a sequential circuit with m flip – flops and n inputs needs $2^{m+n}$ rows in the state table.
### State Table

<table>
<thead>
<tr>
<th>Present State (t)</th>
<th>Input (t)</th>
<th>Next State (t+1)</th>
<th>Output</th>
<th>A</th>
<th>B</th>
<th>x</th>
<th>A</th>
<th>B</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- \( A(t+1) = A \cdot x + B \cdot x \)
- \( B(t+1) = A' \cdot x \)
- \( y = (A + B) \cdot x' \)
State Table 2

<table>
<thead>
<tr>
<th>Present State (t)</th>
<th>Next State (t+1)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=0</td>
<td>x=1</td>
<td>x=0</td>
</tr>
<tr>
<td>AB</td>
<td>AB</td>
<td>y</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>00</td>
<td>10</td>
</tr>
</tbody>
</table>

- \( A(t+1) = A \cdot x + B \cdot x \)
- \( B(t+1) = A' \cdot x \)
- \( y = (A + B) x' \)
State Diagram

• The information available in a state table can be represented graphically in the form of a state diagram.
• State is represented by a circle
• Transition between states are indicated by directed lines connecting the circles.
## Analysis of Clocked Sequential Circuits

### State Diagram

<table>
<thead>
<tr>
<th>Present State (t)</th>
<th>Next State (t+1) x=0</th>
<th>x=1</th>
<th>Output y=0</th>
<th>y=1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AB</strong></td>
<td><strong>AB</strong></td>
<td><strong>AB</strong></td>
<td><strong>y</strong></td>
<td><strong>y</strong></td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
<td>11</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
<td>10</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>00</td>
<td>10</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **AB** represents input/output.
Analysis with D Flip – Flops

- \( A(t+1) = D_A = A \oplus x \oplus y \)

<table>
<thead>
<tr>
<th>Present state</th>
<th>Inputs</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Analysis with JK Flip – Flops

- $J_A = B$
- $K_A = B \cdot x'$
- $J_B = x'$
- $K_B = A \oplus x$
- $A(t+1) = J_A Q'_A + K'_A Q_A$
  \[ = A'B + AB' + A x \]
- $B(t+1) = J_B Q'_B + K'_B Q_B$
  \[ = B'x' + ABx + A'Bx' \]
Analysis with JK Flip – Flops

- \( J_A = B \)  \( K_A = B \times' \)
- \( J_B = x' K_B = A \oplus x \)
- \( A(t+1) = J_A Q'_A + K'_A Q_A \)
  \( = A'B + AB' + Ax \)
- \( B(t+1) = J_B Q'_B + K'_B Q_B \)
  \( = B'x' + ABx + A'Bx' \)

**Table:**

<table>
<thead>
<tr>
<th>Present State</th>
<th>I/P</th>
<th>Next State</th>
<th>Flip – Flop Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>x</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Analysis with JK Flip – Flops

<table>
<thead>
<tr>
<th>Present State</th>
<th>I/P</th>
<th>Next State</th>
<th>Flip – Flop Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>x</td>
</tr>
<tr>
<td>0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Analysis with T Flip – Flops

- **T_A** = B.x  \quad \text{\textbf{\textit{T_B}} = x}
- **y** = A \cdot B
- **Q(t+1) = T \oplus Q = T'Q + TQ'**
- **A(t+1) = T_A \oplus A = T_A' A + T_A A'**
  \quad = (Bx)' A + BxA'
  \quad = (B' + x')A + A'Bx
  \quad = AB' + Ax' + A'Bx
- **B(t+1) = T_B \oplus B = T_B' B + T_B B'**
  \quad = x'B +xB'
  \quad = x \oplus B
ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS

Analysis with T Flip – Flops

- \( T_A = B.x \) \( T_B = x \)
- \( y = A \cdot B \)
- \( Q(t+1) = T \bigoplus Q = T’Q + TQ’ \)
- \( A(t+1) = T_A \bigoplus A = T_A' A + T_A A' \)
  = \((Bx)’ A + BxA’\)
  = \((B’ + x’)A + A’Bx\)
  = \(AB’ + Ax’ + A’Bx\)
- \( B(t+1) = T_B \bigoplus B = T_B’ B + T_B B’ \)
  = \(x’B + xB’\)
  = \(x \bigoplus B\)

<table>
<thead>
<tr>
<th>Present State</th>
<th>I/P</th>
<th>Next State</th>
<th>FF Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>x</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Analysis with T Flip – Flops

<table>
<thead>
<tr>
<th>x</th>
<th>x = 0</th>
<th>x = 1</th>
<th>x = 0</th>
<th>x = 1</th>
<th>x = 0</th>
<th>x = 1</th>
<th>x = 0</th>
<th>x = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>x</td>
<td>A</td>
<td>B</td>
<td>T_A</td>
<td>T_B</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Mealy and Moore Models

- The most general model of a sequential circuit has:
  - Inputs
  - Outputs
  - Internal states.

- Sequential circuits are divided into two (they differ in the way output is generated):
  - Mealy model
  - Moore model
Mealy and Moore Models

• Mealy model:
  – The output is a function of both the present state and input.
  – 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.
  – Example of Sequential Circuit
Mealy and Moore Models

- Moore model:
  - The output is function of the present state only.
  - The outputs are synchronous with the clocks.
  - Example of Sequential Circuit
Mealy and Moore Models

**Mealy Machine**
- Inputs → Next State Combinational Logic → State Register → Outputs (Mealy - type)
- Clock

**Moore Machine**
- Inputs → Next State Combinational Logic → State Register → Outputs (Moore - type)
- Clock
### Mealy

<table>
<thead>
<tr>
<th>Present State</th>
<th>I/P</th>
<th>Next State</th>
<th>O/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B x A B y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1 0 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0 0 0 1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0 1 1 1 1 0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1 0 0 0 0 1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1 0 1 1 0 0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1 1 0 0 0 1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1 1 1 1 0 0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

For the same **state**, the output **changes** with the input.

### Moore

<table>
<thead>
<tr>
<th>Present State</th>
<th>I/P</th>
<th>Next State</th>
<th>O/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A B x A B y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1 0 1 0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0 1 0 0 1 0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0 1 1 1 0 0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1 0 0 1 0 0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1 0 1 1 1 0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1 1 0 1 1 1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1 1 1 0 0 1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

For the same **state**, the output **does not change** with the input.
5.5 STATE REDUCTION AND ASSIGNMENT
STATE REDUCTION AND ASSIGNMENT

• The analysis of sequential circuits
  – starts from a circuit diagram and
  – culminates in a state table or diagram.

• The design of a sequential circuits
  – starts from a set of specifications and
  – culminates in a logic diagram.
• State – reduction algorithms are concerned with procedures for reducing the number of states in a state table, while keeping the external input – output requirements unchanged.

• Since $m$ flip – flops produce $2^m$ states,
  – a reduction in the number of states may (or may not) result in a reduction in the number of flip – flops.

• An unpredictable effect in reducing the number of flip – flops is that sometimes the equivalent circuit (with fewer flip – flops) may require more combinational gates.
Consider a sequential circuit whose specification is given in the state diagram.

There are infinite number of input sequence that may be applied to the circuit;
- Each results in a unique output sequence.
Consider input sequence

- 01010110100
- Starting from the initial state a.
- Each input of 0/1 produces an output of 0/1 and causes circuit to go to the next state.
Consider input sequence
- 01010110100
STATE REDUCTION AND ASSIGNMENT

• Two circuits are **equivalent**
  – Have identical outputs for all input sequences;
  – The number of states is not important.

• The problem of state reduction is
  – To find ways of reducing the number of states in a sequential circuit without altering the input – output relationships.
**STATE REDUCTION AND ASSIGNMENT**

- **Equivalent States**
  - Two states are said to be 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.
  - When two states are equivalent, one of them can be removed without altering the input – output relationships.
• Reduce the number of states
  – Draw a state table

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( x = 0 )</td>
<td>( x = 1 )</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>e</td>
<td>a</td>
<td>f</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>f</td>
</tr>
<tr>
<td>g</td>
<td>a</td>
<td>f</td>
</tr>
</tbody>
</table>
STATE REDUCTION AND ASSIGNMENT

- Reduce the number of states
  - $e = g$ (remove $g$)
  - The row $g$ is removed.
  - State $g$ is replaced by state $e$ each time it occurs in the next-state columns.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x = 0$</td>
<td>$x = 1$</td>
</tr>
<tr>
<td>$a$</td>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>$b$</td>
<td>$c$</td>
<td>$d$</td>
</tr>
<tr>
<td>$c$</td>
<td>$a$</td>
<td>$d$</td>
</tr>
<tr>
<td>$d$</td>
<td>$e$</td>
<td>$f$</td>
</tr>
<tr>
<td>$e$</td>
<td>$a$</td>
<td>$f$</td>
</tr>
<tr>
<td>$f$</td>
<td>$g$</td>
<td>$f$</td>
</tr>
<tr>
<td>$g$</td>
<td>$a$</td>
<td>$f$</td>
</tr>
</tbody>
</table>
Reduce the number of states

- Present state \( f \) has now next states \( e \) and \( f \) and outputs 0 and 1 for \( x = 0 \) and \( x = 1 \).
- Then, \( d = f \) (remove \( f \)).
- The row \( f \) is removed.
- The state \( f \) is replaced by state \( d \).

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state ( x = 0 )</th>
<th>Next state ( x = 1 )</th>
<th>Output ( x = 0 )</th>
<th>Output ( x = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>d</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>d</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>f</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>a</td>
<td>f</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td>f</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Reduce the number of states

- Final table
- This table satisfies the original input–output specifications and will produce the required output sequence for any given input sequence.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state $x = 0$</th>
<th>Next state $x = 1$</th>
<th>Output $x = 0$</th>
<th>Output $x = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>d</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>d</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>d</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>a</td>
<td>d</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
• Reduce the number of states

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x = 0</td>
<td>x = 1</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>d</td>
</tr>
<tr>
<td>e</td>
<td>a</td>
<td>d</td>
</tr>
</tbody>
</table>
STATE REDUCTION AND ASSIGNMENT

• Reduce the number of states
  – The checking of each pair of states for possible equivalence can be done systematically using Implication Table.
  – The unused states are treated as don't-care condition \(\Rightarrow\) fewer combinational gates.
• Implication Table (**extra reading**)  
  - The state-reduction procedure for completely specified state tables is based on the algorithm that two states in a state table can be combined into one if they can be shown to be equivalent. There are occasions when a pair of states do not have the same next states, but, nonetheless, go to equivalent next states. Consider the following state table:
• Implication Table (extra reading)
  - Consider the following state table:
  - (a, b) imply (c, d) and (c, d) imply (a, b). Both pairs of states are equivalent; i.e., a and b are equivalent as well as c and d.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x = 0</td>
<td>x = 1</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>d</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>b</td>
<td>d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>x = 0</th>
<th>x = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x = 1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
STATE REDUCTION AND ASSIGNMENT

- Implication Table (extra reading)
  - The checking of each pair of states for possible equivalence in a table with a large number of states can be done systematically by means of an implication table. This a chart that consists of squares, one for every possible pair of states, that provide spaces for listing any possible implied states. Consider the following state table:
• Implication Table (**extra reading**)  
  – Consider the following state table:

<table>
<thead>
<tr>
<th>Present state</th>
<th>x = 0</th>
<th>x = 1</th>
<th>x = 0</th>
<th>x = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>d</td>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>e</td>
<td>a</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>g</td>
<td>f</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>a</td>
<td>d</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>a</td>
<td>d</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>c</td>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>a</td>
<td>e</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
STATE REDUCTION AND ASSIGNMENT

• Implication Table (extra reading)
  – On the left side along the vertical are listed all the states defined in the state table except the last, and across the bottom horizontally are listed all the states except the last.
  – The states that are not equivalent are marked with a ‘x’ in the corresponding square, whereas their equivalence is recorded with a ‘√’.
• Implication Table (extra reading)
  – Some of the squares have entries of implied states that must be further investigated to determine whether they are equivalent or not.
  – The step-by-step procedure of filling in the squares is as follows:
    1. Place a cross in any square corresponding to a pair of states whose outputs are not equal for every input.
    2. Enter in the remaining squares the pairs of states that are implied by the pair of states representing the squares. We do that by starting from the top square in the left column and going down and then proceeding with the next column to the right.
STATE REDUCTION AND ASSIGNMENT

• Implication Table (extra reading)

3. Make successive passes through the table to determine whether any additional squares should be marked with a ‘x’. A square in the table is crossed out if it contains at least one implied pair that is not equivalent.

4. Finally, all the squares that have no crosses are recorded with check marks. The equivalent states are: \((a, b), (d, e), (d, g), (e, g)\).
STATE REDUCTION AND ASSIGNMENT

- Implication Table (extra reading)
  - We now combine pairs of states into larger groups of equivalent states. The last three pairs can be combined into a set of three equivalent states \((d, e, g)\) because each one of the states in the group is equivalent to the other two. The final partition of these states consists of the equivalent states found from the implication table, together with all the remaining states in the state table that are not equivalent to any other state:
  - \((a, b)\) \((c)\) \((d, e, g)\) \((f)\)
STATE REDUCTION AND ASSIGNMENT

• State Assignment
  – In order to design a sequential circuit with physical components, it is necessary to assign coded binary values to the states.
  – To minimize the cost of the combinational circuits.
  – For a circuit with m states, the codes must contain n bits where $2^n \geq m$.
  – Ex: with 3 bits it is possible to assign codes to 8 states denoted by binary numbers 000 through 111.
STATE REDUCTION AND ASSIGNMENT

• State Assignment
  – If the state table1 is used, we must assign binary values to 7 states.
    • Remaining state is unused.
  – If the state table2 is used, only five states need binary assignment.
    • Remaining 3 state is unused.
    • Unused states treated as don’t care conditions.
    • Since don’t care conditions usually help in obtaining a simpler circuit, it is more likely that the circuit with five states will require fewer combinational gates than the one with seven states.
## State Assignment

<table>
<thead>
<tr>
<th>Present State</th>
<th>Assignment 1</th>
<th>Assignment 2</th>
<th>Assignment 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Binary</td>
<td>Gray Code</td>
<td>One-hot</td>
</tr>
<tr>
<td>a</td>
<td>000</td>
<td>000</td>
<td>00001</td>
</tr>
<tr>
<td>b</td>
<td>001</td>
<td>001</td>
<td>00010</td>
</tr>
<tr>
<td>c</td>
<td>010</td>
<td>011</td>
<td>00100</td>
</tr>
<tr>
<td>d</td>
<td>011</td>
<td>010</td>
<td>01000</td>
</tr>
<tr>
<td>e</td>
<td>100</td>
<td>110</td>
<td>10000</td>
</tr>
</tbody>
</table>

*Binary and Gray Code are used for assignment, while One-hot is used for present state.*
STATE REDUCTION AND ASSIGNMENT

• State Assignment
  – Any binary number assignment is satisfactory as long as each state is assigned a unique number.
  – Use binary assignment 1.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x = 0</td>
<td>x = 1</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>010</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>000</td>
<td>1</td>
</tr>
</tbody>
</table>
5.6 Design Procedure
DESIGN PROCEDURE

• The design of a clocked sequential circuit starts from
  – a set of specifications and
  – culminates in a logic diagram or
  – a list of Boolean functions from which the logic diagram can be obtained.
DESIGN PROCEDURE

1. Derive a state diagram for the circuit from the word description.
2. Reduce the number of states if necessary.
3. Assign binary values to the states.
4. Obtain the binary-coded state table.
5. Choose the type of flip-flops.
6. Derive the simplified flip-flop input equations and output equations.
7. Draw the logic diagram.
• Example: We wish to design a circuit that detects three or more consecutive 1’s in a string of bits coming through an input line.

• State diagram:
This is a Moore model sequential circuit since the output is 1 when the circuit is in State3 and 0 otherwise.

<table>
<thead>
<tr>
<th>State</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$S_1$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$S_2$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$S_3$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Design Procedure

<table>
<thead>
<tr>
<th>Present State</th>
<th>I/P</th>
<th>Next State</th>
<th>O/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₀ / 0</td>
<td>0</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>S₁ / 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S₃ / 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>S₂ / 0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Diagram:**

- **S₀ / 0** to **S₁ / 0** transition on input 0, output 0.
- **S₁ / 0** to **S₂ / 0** transition on input 0, output 0.
- **S₂ / 0** to **S₃ / 1** transition on input 1, output 0.
- **S₃ / 1** to **S₀ / 0** transition on input 0, output 1.

**Notes:**

- The table shows the state transitions and outputs for each input pair.
- The transitions are represented by arrows in the diagram.
DESIGN PROCEDURE

• To implement the circuit,
  – Two D flip-flops are chosen to represent the four states and label their outputs A and B.
  – There is one input x.
  – There is one output y.
  – The characteristic equation of the D flip-flop is
    • $Q(t+1) = DQ$. 

23 December, 2016
To implement the circuit,
- The flip-flop input equations can be obtained directly from the next-state columns of A and B and expressed in sum of minterms.
  - \( A(t+1) = D_A(A,B,x) = \sum (3, 5, 7) \)
  - \( B(t+1) = D_B(A,B,x) = \sum (1, 5, 7) \)
  - \( y(A,B,x) = \sum (6, 7) \)
• Synthesis using D Flip - flops
  – \( A(t+1) = D_A(A,B,x) = \sum (3, 5, 7) \)
  – \( B(t+1) = D_B(A,B,x) = \sum (1, 5, 7) \)
  – \( y(A,B,x) = \sum (6, 7) \)

• \( D_A \)'s K - Map

\[
\begin{array}{ccc}
A & Bx & \text{B} \\
0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & & & \text{m}_1 & & \\
1 & & \text{m}_3 & & & \text{m}_2 & \text{M}_2 \\
1 & \text{m}_5 & & \text{m}_7 & \text{m}_1 & & \text{m}_6 & \text{M}_6 \\
\end{array}
\]

\[ D_A = Ax + Bx \]
• Synthesis using D Flip – flops
  - A(t+1) = D_A(A,B,x) = \sum 3, 5, 7
  - B(t+1) = D_B(A,B,x) = \sum 1, 5, 7
  - y(A,B,x) = \sum 6, 7

• D_B’s K - Map

D_A = A_x + B'x
• Synthesis using D Flip – flops
  - \( A(t+1) = D_A(A,B,x) = \sum (3, 5, 7) \)
  - \( B(t+1) = D_B(A,B,x) = \sum (1, 5, 7) \)
  - \( y(A,B,x) = \sum (6, 7) \)

• \( y \)'s K-Map

\[
\begin{array}{cccc}
A & 00 & 01 & 11 & 10 \\
Bx & 0 & m_0 & m_1 & m_3 & M_2 \\
A & 1 & m_4 & m_5 & m_7 & M_6 \\
\end{array}
\]

\( y = AB \)
• Synthesis using D Flip – flops
  – \( D_A = Ax + Bx \)
  – \( D_B = Ax + B'x \)
  – \( y = AB \)

• Logic Diagram of Sequence Detector

\[
\begin{align*}
\text{A} & \quad \text{D} \quad \text{Q} \quad \text{Q} \\
\text{CLK} & \quad \text{D} \quad \text{Q} \quad \text{Q}
\end{align*}
\]
DESIGN PROCEDURE

• When – D type flip-flops are employed, the input equations are obtained directly from the next state.

• This is not the case for the JK and T types of flip-flops. In order to determine the input equations for these flip flops, it is necessary to derive a functional relationship between the state table and the input equations.
• During the design process we usually know the transition from present state to the next state and wish to find the flip – flop input conditions that will cause the required transition.

• For this reason, we need a table that lists the required inputs for a given change of state. Such table is called an **excitation table**.
D Flip – Flop Excitation table

D Flip – Flop Characteristic Table

<table>
<thead>
<tr>
<th>D</th>
<th>Q(t+1)</th>
<th>Q(t+1) = D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Present State | Next State | F.F. Input
Q(t) | Q(t+1) | D
---|---|---
0 | 0 | 0
0 | 1 | 1
1 | 0 | 0
1 | 1 | 1
**DESIGN PROCEDURE**

- JK Flip – Flop Excitation table

### JK Flip – Flop Characteristic Table

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Q (t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q(t)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Q'(t)</td>
</tr>
</tbody>
</table>

\[Q(t+1) = JQ' + K'Q\]

### Present State | Next State | F.F. Input

<table>
<thead>
<tr>
<th>Q(t)</th>
<th>Q(t+1)</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

- 0 0 (No change)
- 0 1 (Reset)
- 1 0 (Set)
- 1 1 (Toggle)
T Flip – Flop Excitation table

T Flip – Flop Characteristic Table

<table>
<thead>
<tr>
<th>T</th>
<th>Q (t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Q(t)</td>
</tr>
<tr>
<td>1</td>
<td>Q'(t)</td>
</tr>
</tbody>
</table>

\[ Q(t+1) = T \oplus Q \]
• Synthesis Using JK Flip – Flops: Detect 3 or more consecutive 1’s
### DESIGN PROCEDURE

- **Synthesis Using JK Flip – Flops: Detect 3 or more consecutive 1’s**

<table>
<thead>
<tr>
<th>Present State</th>
<th>Input</th>
<th>Next State</th>
<th>Flip-Flop Inputs</th>
<th>J_A(A, B, x) = \sum (3, 4, 5, 6, 7)</th>
<th>K_A(A, B, x) = \sum (0, 1, 2, 3, 4, 6)</th>
<th>J_B(A, B, x) = \sum (1, 2, 3, 5, 6, 7)</th>
<th>K_B(A, B, x) = \sum (0, 1, 2, 3, 4, 5, 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>x</td>
<td>A</td>
<td>B</td>
<td>J_A</td>
<td>K_A</td>
<td>J_B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>
Synthesis Using JK Flip – Flops:
Detect 3 or more consecutive 1’s
- \( J_A(A, B, x) = \sum \{3, 4, 5, 6, 7\} \)
- \( K_A(A, B, x) = \sum \{0, 1, 2, 3, 4, 6\} \)
- \( J_B(A, B, x) = \sum \{1, 2, 3, 5, 6, 7\} \)
- \( K_B(A, B, x) = \sum \{0, 1, 2, 3, 4, 5, 6\} \)

\( J_A \)'s K-Map

\[ J_A = Bx \]
### Synthesis Using JK Flip – Flops:

Detect 3 or more consecutive 1’s

- \( J_A(A, B, x) = \sum (3, 4, 5, 6, 7) \)
- \( K_A(A, B, x) = \sum (0, 1, 2, 3, 4, 6) \)
- \( J_B(A, B, x) = \sum (1, 2, 3, 5, 6, 7) \)
- \( K_B(A, B, x) = \sum (0, 1, 2, 3, 4, 5, 6) \)

### \( K_A \)’s K-Map

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( K_A = x' \)
Synthesis Using JK Flip – Flops:

Detect 3 or more consecutive 1’s

- \( J_A(A, B, x) = \Sigma \{3, 4, 5, 6, 7\} \)
- \( K_A(A, B, x) = \Sigma \{0, 1, 2, 3, 4, 6\} \)
- \( J_B(A, B, x) = \Sigma \{1, 2, 3, 5, 6, 7\} \)
- \( K_B(A, B, x) = \Sigma \{0, 1, 2, 3, 4, 5, 6\} \)

\( J_B \)'s K-Map:

\[
\begin{array}{cccc}
\text{A} & & \text{Bx} & \\
0 & & 00 & 01 & 11 & 10 \\
0 & m_0 & m_1 & m_3 & m_2 & \text{X} & \text{X} \\
1 & m_4 & m_5 & m_7 & M_6 & \text{X} \\
\end{array}
\]

\( J_B = \text{x} \)
Synthesis Using JK Flip – Flops:

- \( J_A (A, B, x) = \sum (3, 4, 5, 6, 7) \)
- \( K_A (A, B, x) = \sum (0, 1, 2, 3, 4, 6) \)
- \( J_B (A, B, x) = \sum (1, 2, 3, 5, 6, 7) \)
- \( K_B (A, B, x) = \sum (0, 1, 2, 3, 4, 5, 6) \)

**K_B**’s K-Map

\[
\begin{array}{cccccc}
A & 00 & 01 & 11 & 10 \\
Bx & \times & \times & 1 & 1 \\
\hline
0 & m_0 & m_1 & m_3 & m_2 \\
1 & m_4 & m_5 & m_7 & m_6 \\
\end{array}
\]

\( K_B = A' + x' \)
• Synthesis Using JK Flip – Flops:
  Detect 3 or more consecutive 1’s
  - \( J_A = Bx \)
  - \( K_A = x' \)
  - \( J_B = x \)
  - \( K_B = A' + x' \)

• Logic Diagram of Sequence Detector

![Logic Diagram of Sequence Detector](image)
• Synthesis Using T Flip – Flops: 3-bit Counter. An n-bit binary counter consists of n flip – flops that can count in binary from 0 to $2^n - 1$. 
• Synthesis Using T Flip – Flops: 3-bit Counter.

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Flip-Flop Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2 A1 A0</td>
<td>A2 A1 A0</td>
<td>T_{A2} T_{A1} T_{A0}</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 1 0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1 0 0</td>
<td>1 1 1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 1 0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 1 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0 0 0</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

\[
T_{A2}(A_2, A_1, A_0) = \sum (3, 7)
\]
\[
T_{A1}(A_2, A_1, A_0) = \sum (1, 3, 5, 7)
\]
\[
T_{A0}(A_2, A_1, A_0) = \sum (0, 1, 2, 3, 4, 5, 6, 7)
\]
Synthesis Using T Flip – Flops: 3-bit Counter.

- \( T_{A2}(A_2, A_1, A_0) = \sum (3, 7) \)
- \( T_{A1}(A_2, A_1, A_0) = \sum (1, 3, 5, 7) \)
- \( T_{A0}(A_2, A_1, A_0) = \sum (0, 1, 2, 3, 4, 5, 6, 7) \)

\[ T_{A2}'s \ K-Map \]

\[ T_{A2} = A_1A_0 \]
Synthesis Using T Flip – Flops: 3-bit Counter.

- \( T_{A2}(A_2, A_1, A_0) = \sum (3, 7) \)
- \( T_{A1}(A_2, A_1, A_0) = \sum (1, 3, 5, 7) \)
- \( T_{A0}(A_2, A_1, A_0) = \sum (0, 1, 2, 3, 4, 5, 6, 7) \)

\[ \begin{array}{c|ccc|c}
& 00 & 01 & 11 & 10 \\
A_2 & m_0 & m_1 & m_3 & m_2 \\
A_1 & m_4 & m_5 & m_7 & m_6 \\
A_0 & & & & \\
\end{array} \]

\[ T_{A1} = A_0 \]
Design Procedure

- Synthesis Using T Flip – Flops: 3-bit Counter.
  - $T_{A_2}(A_2, A_1, A_0) = \sum (3, 7)$
  - $T_{A_1}(A_2, A_1, A_0) = \sum (1, 3, 5, 7)$
  - $T_{A_0}(A_2, A_1, A_0) = \sum (0, 1, 2, 3, 4, 5, 6, 7)$

- $T_{A_0}$'s K-Map

\[
\begin{array}{cccc}
A_1 & 00 & 01 & 11 & 10 \\
A_2 & 0 & m_0 & m_1 & m_2 \\
& 1 & 1 & 1 & 1 \\
& 1 & m_3 & m_4 & m_5 \\
& 1 & 1 & m_6 & 1 \\
A_0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

$T_{A_0} = 1$
Synthesis Using T Flip – Flops: 3-bit Counter.
- \( T_{A2} = A_1A_0 \)
- \( T_{A1} = A_0 \)
- \( T_{A0} = 1 \)

Logic Diagram of 3-bit Binary Counter
THANK YOU!
GOOD LUCK!