# Sequential Circuits

**Instructor: Saraju P. Mohanty**

## Part 2

- Sequential circuit analysis
- Sequential circuit design
- Designing with D Flip-Flop
- Designing with JK Flip-Flop

NOTE: Mostly adapted from Dr. Valavanis lectures.

# Sequential Circuit Analysis: basics

- Behavior is determined from input, output, and present state of the circuit.
- The output is dependent on the present input and the present states in case of a Mealy machine.
- In a Moore machine, the output is only a function of the present states.
- The next state functions are dependent on the present input and the present states.
- The input equations, state tables and state diagrams are three important tools to describe a sequential machine.

# Sequential Circuit Analysis: Input Equations

The part of the combinatorial circuit that generates the signals for the inputs of flip-flops can be described by a set of Boolean expressions called flip-flop input equations.

For example: $J_A = (XB+Y'C)$, $K_A = (YB'+C)$

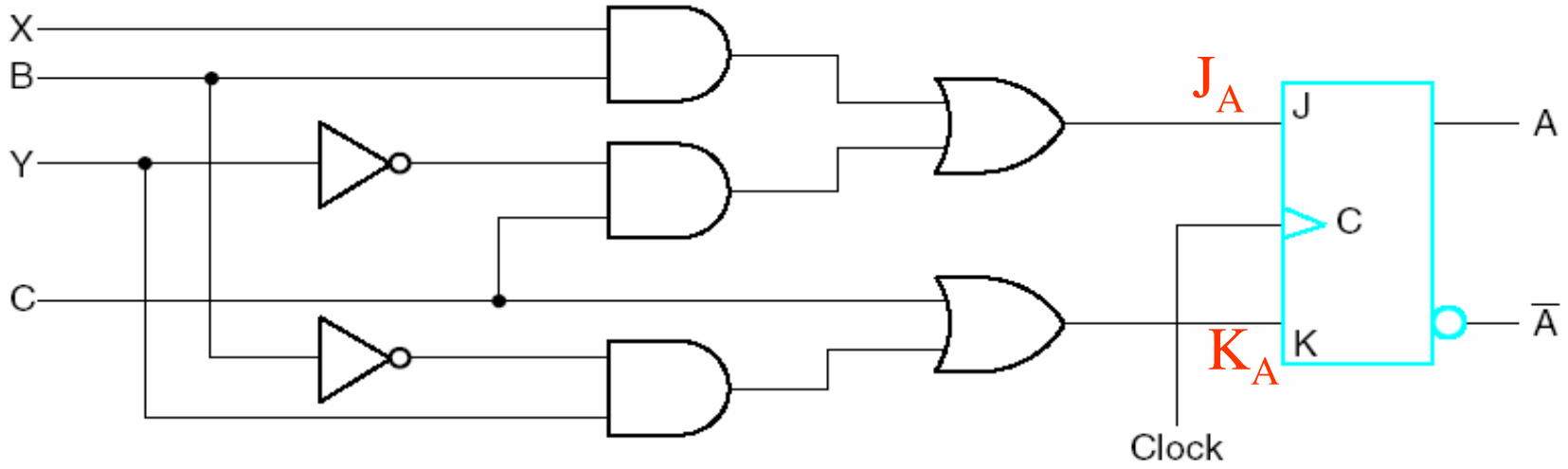
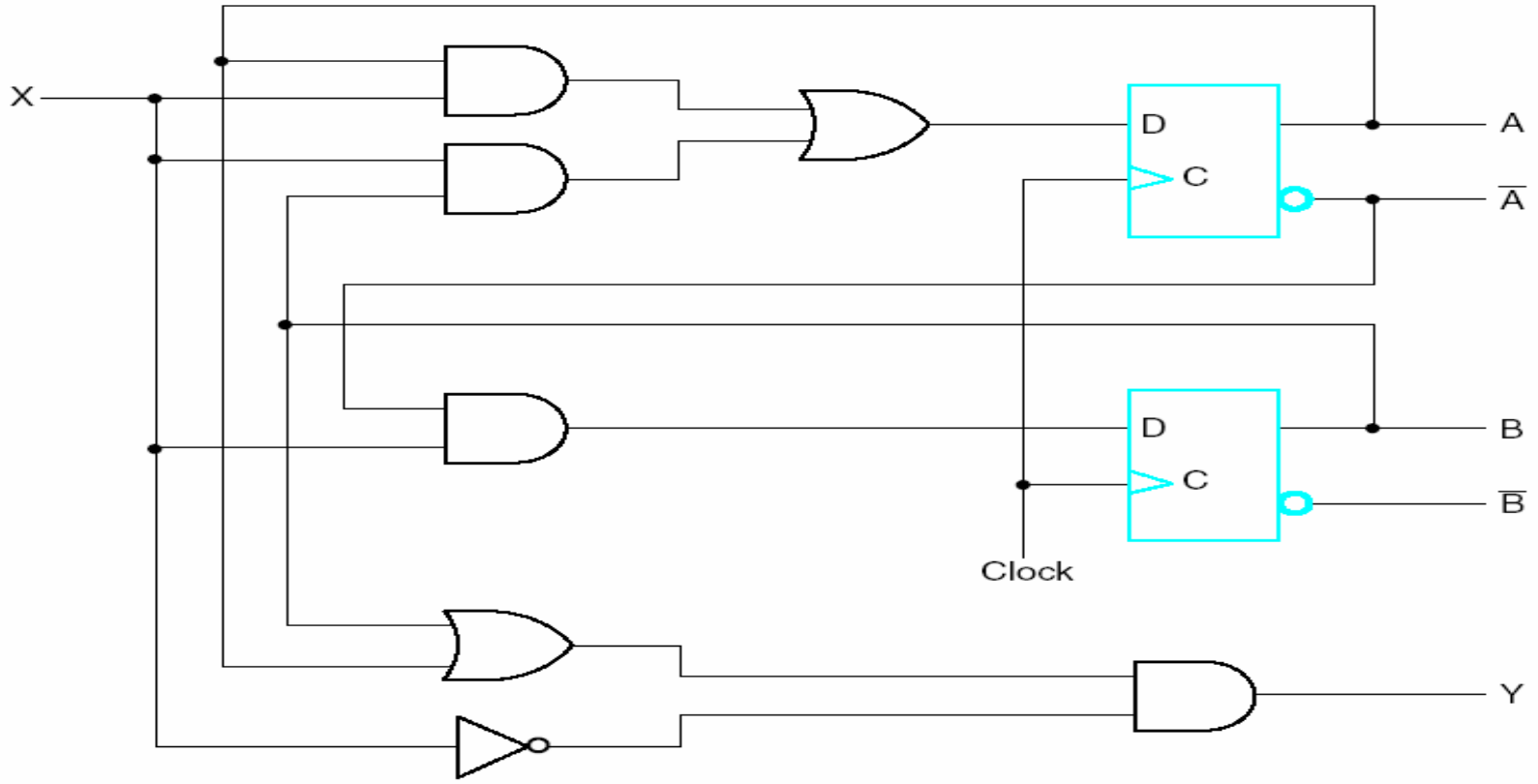



Fig. 4-17 Implementing Input Equations

Convention: J and K are the input symbols of a FF and A is the output of the FF which is used as subscript.

# Sequential Circuit Analysis: Input Equations




The circuit has two D-type of FFs, an input X, and an output Y. The following equations specify the circuit.

$$D_A = (AX+BX),\ D_B = A'X,\ \text{and}\ Y = (A+B)X'$$

# Sequential Circuit Analysis: State Table

•State table specify the functional relationship between the inputs, outputs and FF states of a sequential circuits.

•The table consists of four sections, labeled present state, input, next state, and output.

| 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 |

Table 4-2 State Table for Circuit of Figure 4-18

# Sequential Circuit Analysis: State Table

- The derivation of a state table has following steps:
  - List all possible binary combinations of present state and inputs.
  - Determine next states from logic diagram or from flip-flop equations.
- In general, a sequential circuit with m FFs and n inputs needs $2^{m+n}$ rows in state table. The next state section has m columns, one for each FF.

# Sequential Circuit Analysis: State Table

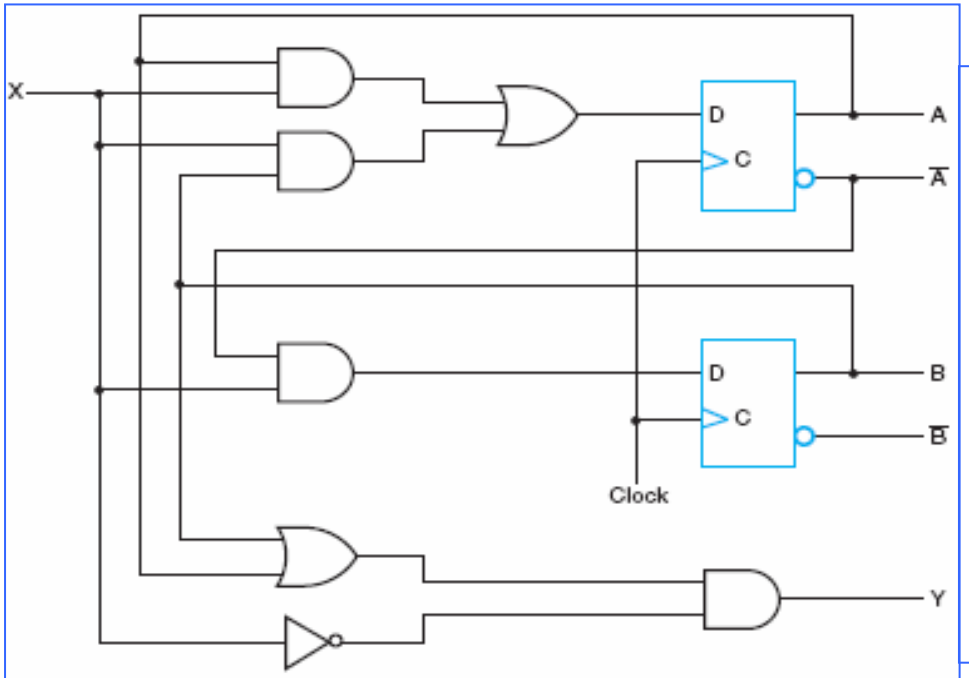



Fig. 4-18 Example of a Sequential Circuit

For Mealy circuits, two dimensional state tables cane be used. Two dimensional, meaning the present state and input combination are separated.

Example given below

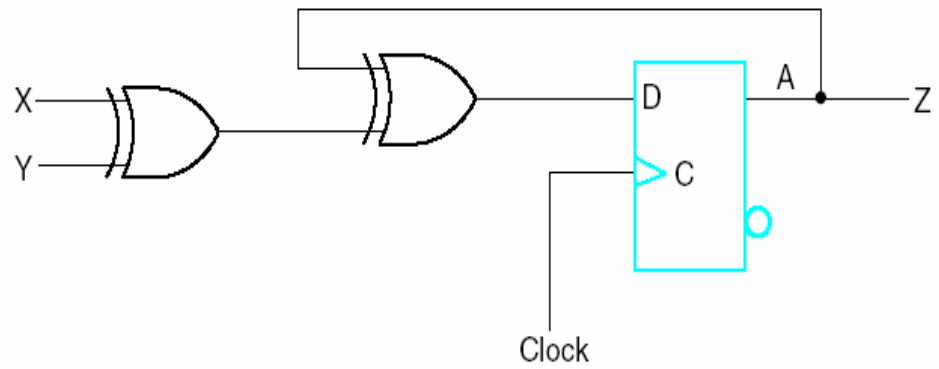
| Present State | | Input | Next State | | Output |
|---|---|---|---|---|---|
| A | B | X | A | B | Y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

Table 4-2 State Table for Circuit of Figure 4-18

| 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 |

Table 4-3 Two-Dimensional State Table for the Circuit in Figure 4-18

# Sequential Circuit Analysis: State Table




(a)

| Present state | Inputs | | Next state | Output |
|---|---|---|---|---|
| A | X | Y | A | Z |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

(b) State table

Fig. 4-19 Logic Diagram and State Table for $D_A = A \oplus X \oplus Y$

## Example Moore Circuit

- The output is function of states only.
- The state table is given.
- The flip-flop input equations:

$$D_A = A \text{ XOR } X \text{ XOR } Y$$

- The output equation:

$$Z = A$$

# Sequential Circuit Analysis: State Table (Analysis with JK flip-flops)

- The analysis of circuits with other FFs can be performed similarly.
- For circuits with other types of FFs, such as JK, the next-state values are obtained by following a two-step procedure:
  - Obtain the binary values of each FF input equation in terms of the present-state and input variables
  - Use the corresponding FF characteristic to determine the next-state.

(a) *JK* Flip-Flop

| J | K | $Q(t+1)$ | Operation |
|---|---|---|---|
| 0 | 0 | $Q(t)$ | No change |
| 0 | 1 | 0 | Reset |
| 1 | 0 | 1 | Set |
| 1 | 1 | $\overline{Q}(t)$ | Complement |

# Sequential Circuit Analysis: State diagrams

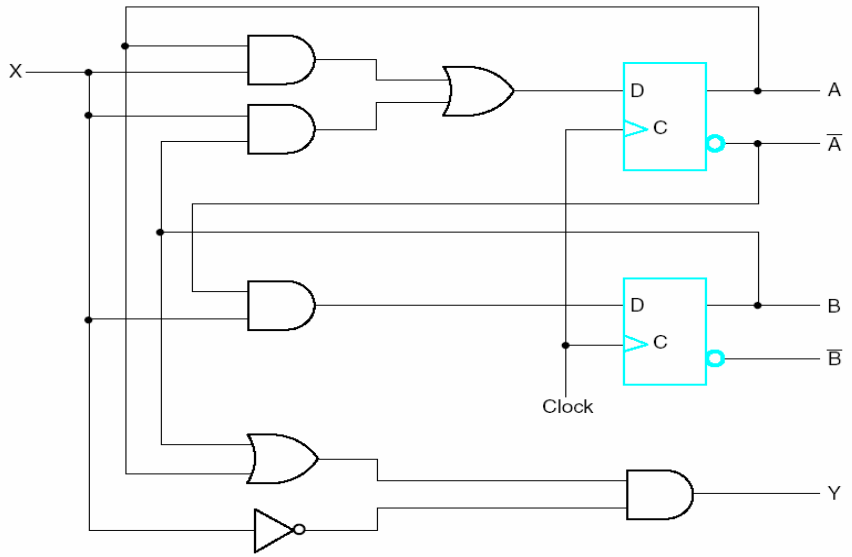


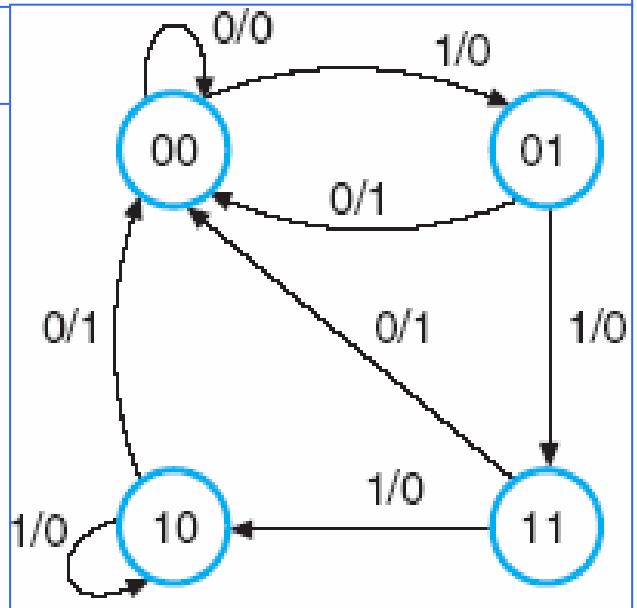

Fig. 4-18 Example of a Sequential Circuit

**TABLE 4-2**
**State Table for Circuit of Figure 4-18**

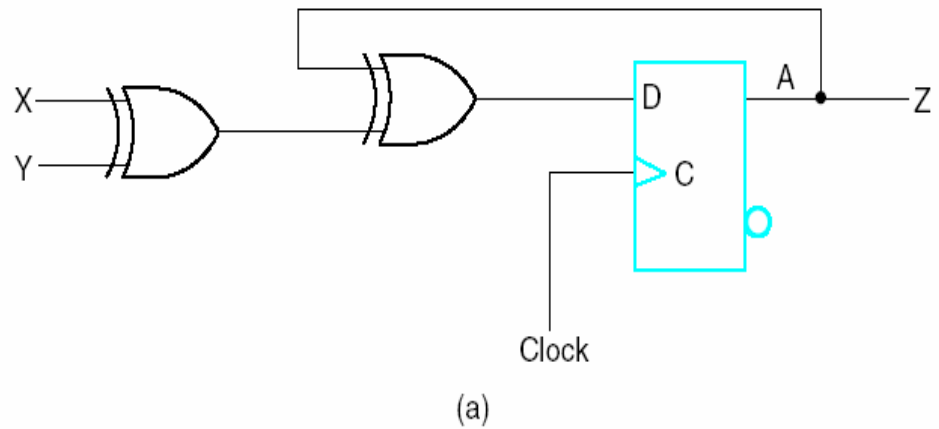
| Present State | | Input | Next State | | Output |
|---|---|---|---|---|---|
| A | B | X | A | B | Y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

Table 4-2 State Table for Circuit of Figure 4-18

Binary number inside each circle identifies FF states. Directed lines are labeled with two binary numbers separated by /. The input value during the present state precedes slash, value following slash gives output value during present state with given input applied. Example: directed line from 00 to 01 is 1/0, meaning when circuit in present state 00 and input is 1, output is 0. After next clock transition, circuit goes to 01.



# Sequential Circuit Analysis: State diagrams



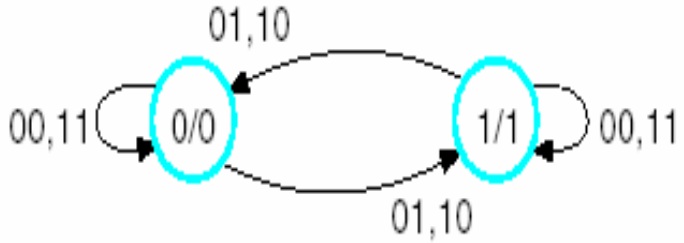

(a)

| Present state | Inputs | | Next state | Output |
|---|---|---|---|---|
| A | X | Y | A | Z |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

(b) State table

Fig. 4-19 Logic Diagram and State Table for $D_A = A \oplus X \oplus Y$

Only one FF with two states is needed. Two binary inputs and the output depends only on the state of the FF. Slash is not included, since outputs depend only on the state and not input values. Two input conditions for each state transition, separated by a comma.




(b)

# Sequential Circuit Analysis: Few Points

- There is no difference between a state table and a state diagram.
- The state diagram is a graphical representation of the state table.
- The state table is easier to derive from a given logic diagram and input equations.
- The state diagram directly follows from the state table.
- The state diagram that gives pictorial view of state transitions is more suitable for human interpretation.
- In general, a Moore machine requires more states than a Mealy machine.

# Sequential Circuit Design

1. Obtain either the state diagram or the state table from problem statement.
2. If only a state diagram is available from step 1, obtain state table.
3. Assign binary codes to the states.
4. Derive the FF input equations from the next-state entries in the encoded state table.
5. Derive output equations from the output entries in the state table.
6. Simplify the FF input and output equations.
7. Draw the logic diagram with D flip-flops and combinatorial gates as specified by the FF input and output equations.

# Sequential Circuit Design: Example

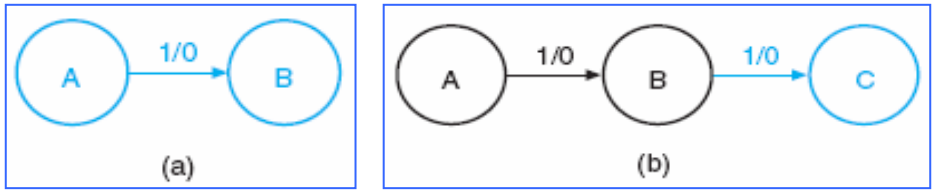
Problem statement: The circuit has one input X and one output Z. Recognize the occurrence of a sequence of 1101 on X by making Z equal to 1, when the previous three inputs to the circuits were 110 and current input is a 1. Otherwise Z equals 0.

## Key Point

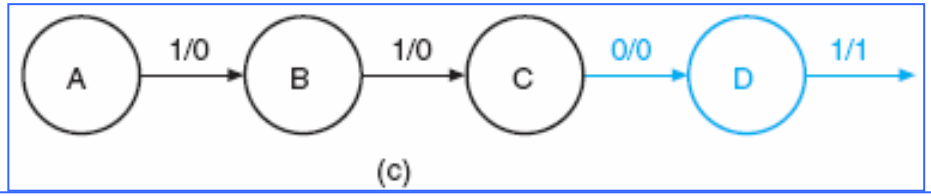
States are used to "remember" something about history of past inputs. For example, for sequence 1101, in order to be able to produce the output value 1 coincident with the final 1 in the sequence, the circuit must be in a state that remembers previous three inputs.

# Sequential Circuit Design: Example …

- "A" is a state in which none of the first portion of the sequence to be recognized has occurred.
- If input is "1", since 1 is the first bit in the sequence, the event must be remembered and a new state is established. State B remembers "1".

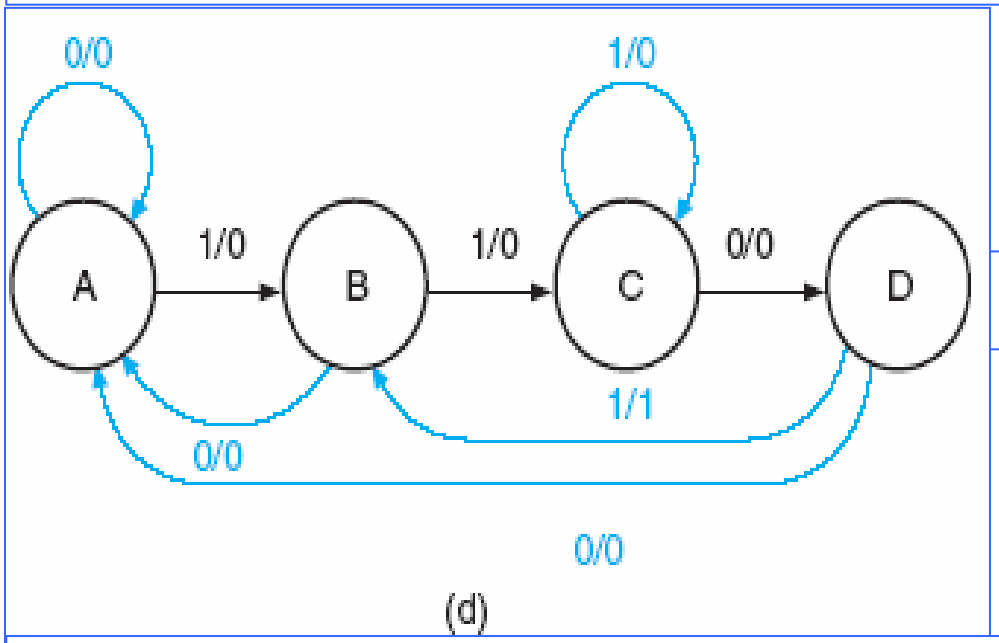



- The next bit of the sequence is 1 again. When this 1 occurs in state B, a state is needed to represent the occurrence of two 1's in a row on the input. Thus giving rise to state C.
- The next bit of the sequence is a 0. Thus a state is needed to represent two 1's followed by a 0. So, additional state D.



# Sequential Circuit Design: Example …

Since state D represents the occurrence of 110 as the previous three input bit values on X, the occurrence of a 1 in state D completes the sequence to be recognized. So the transition for the input value 1 from state D has output value of 1.




Complete state diagram.

NOTE: Similarly state diagram of a BCD to excess-3 decoder can be done (refer, Example 4-2, page-211).

# Example: BCD-to-Excess-3 code converter

Excess-3 code for a decimal digit is the binary combination corresponding to the decimal digit plus 3. Excess-3 code for decimal 5 is 8.

Only 10 out of 16 combinations are listed below. Rest treated as "don't care conditions". BCD and excess-3 use 4 bits → 4 input and 4 output variables.

**TABLE 3-2**
**Truth Table for Code Converter Example**

| Decimal Digit | Input BCD | | | | Output Excess-3 | | | |
|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | W | X | Y | Z |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

Table 3-2 Truth Table for Code Converter Example

# Example: BCD-to-Excess-3 code converter …

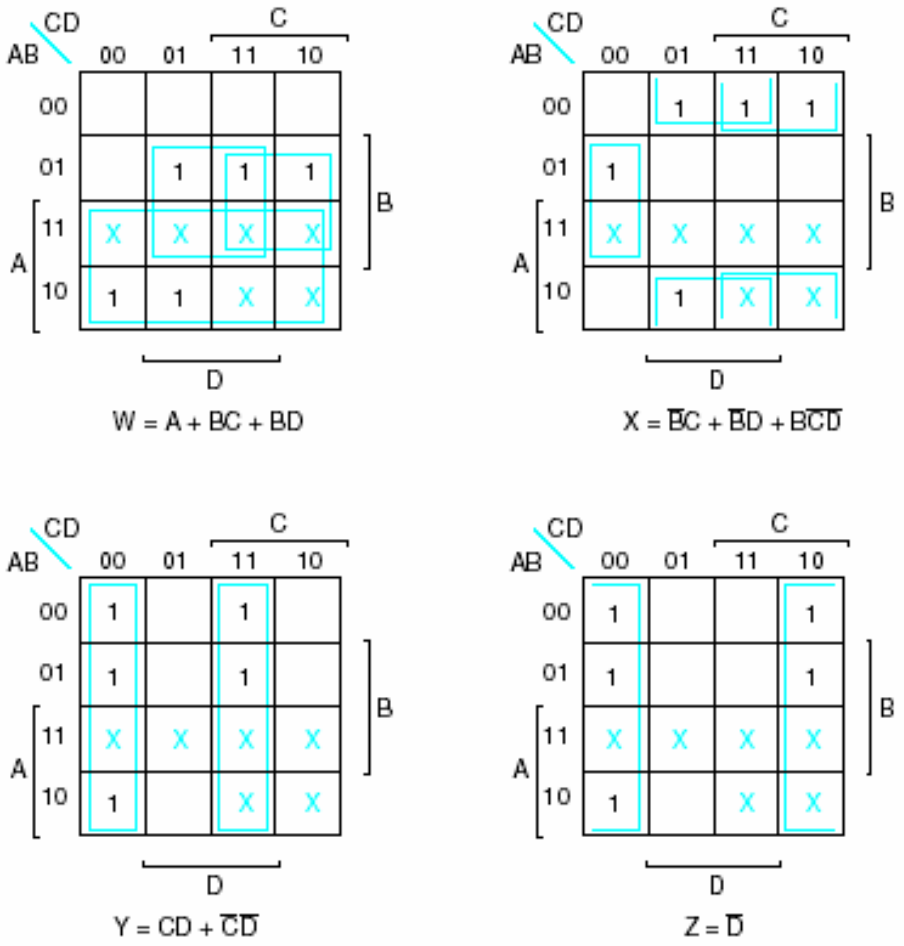



Fig. 3-10 Maps for BCD-to-Excess-3 Code Converter

Maps needed to get simplified expressions for the output variables.

# Example: BCD-to-Excess-3 code converter …

W = A+BC+BD = A+B(C+D), X = B'C+B'D+BC'D' = B'(C+D)+BC'D', Y=CD+C'D' = (C XOR D)', Z=D'

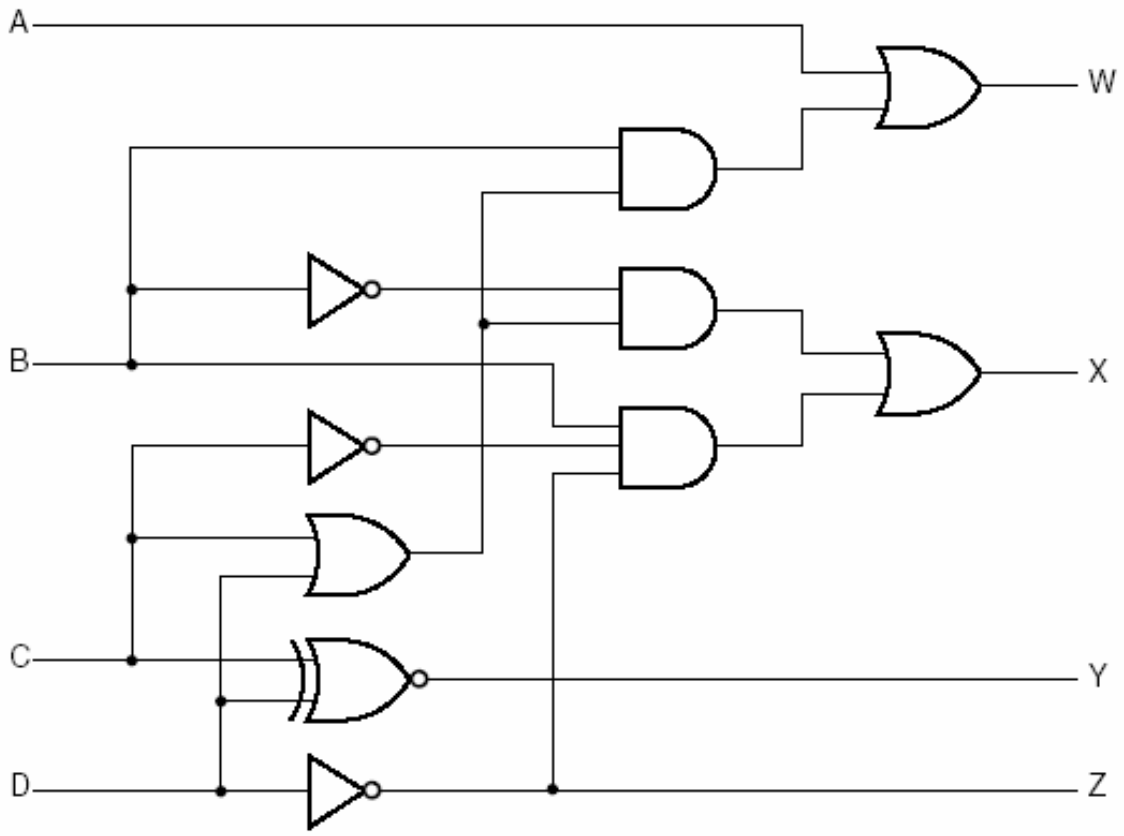



Fig. 3-11 Logic Diagram of BCD-to-Excess-3 Code Converter

# Example: BCD-to-Excess-3 code converter …

Inputs, rather than being presented simultaneously are presented serially, LSB first during successive clock cycles. E.g., during four successive clock cycles, if 1010 is applied to input, output will be 0001. To produce each output bit in the same clock cycle as the corresponding input bit, the output depends on the presently-applied input value as well as the state. The specifications state that circuit must be ready to receive a new 4-bit sequence as soon as the prior sequence is complete.

LSB first, 1st part of Table

Sort rows of first part according to first bit value, second bit value and the third bit value…

**TABLE 4-6**
**Sequence Tables for Code Converter Example**

| Sequences in Order of Digits Represented | | | | | | | | Sequences in Order of Common Prefixes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BCD Input | | | | Excess-3 Output | | | | BCD Input | | | | Excess-3 Output | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

Table 4-6 Sequence Tables for Code Converter Example

# Example: BCD-to-Excess-3 code converter …

Observe that 0 produces a 1, a 1 produces a zero

**TABLE 4-6**
**Sequence Tables for Code Converter Example**

| Sequences in Order of Digits Represented | | | | | | | | Sequences in Order of Common Prefixes | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BCD Input | | | | Excess-3 Output | | | | BCD Input | | | | Excess-3 Output | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

Table 4-6 Sequence Tables for Code Converter Example

# Example: BCD-to-Excess-3 code converter …

Do we need to remember the value of the first bit? When the first bit is 0, a 0 in the second bit results in an output of 1 and a 1 in the second bit gives an output of 0. In contrast, if the first bit is a 1, a 0 in the second bit output 0 and a 1 in the second bit gives output 1. Thus, first input equal 0 and first input equal 1 must give different states as:

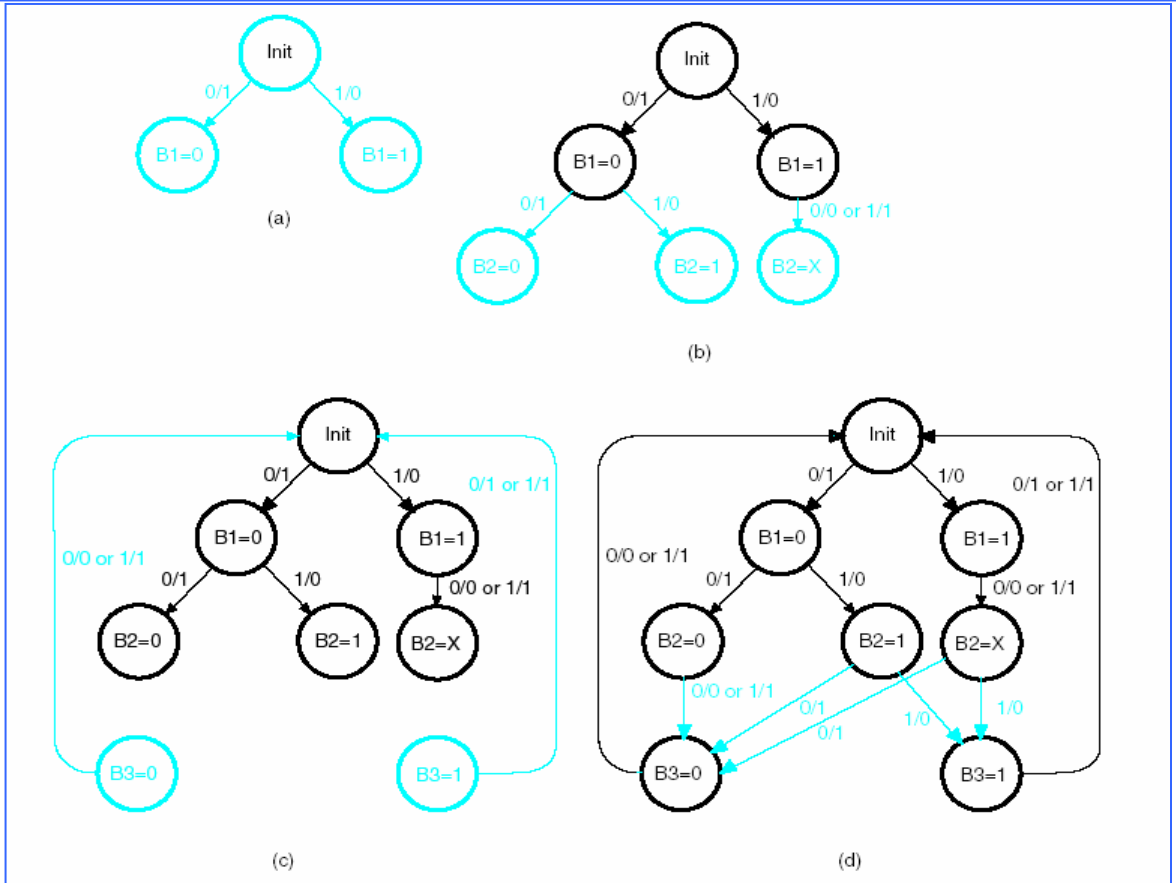



Fig. 4-22 Construction of a State Diagram

# Design with D flip-flops

Remainder steps of sequential circuit design.

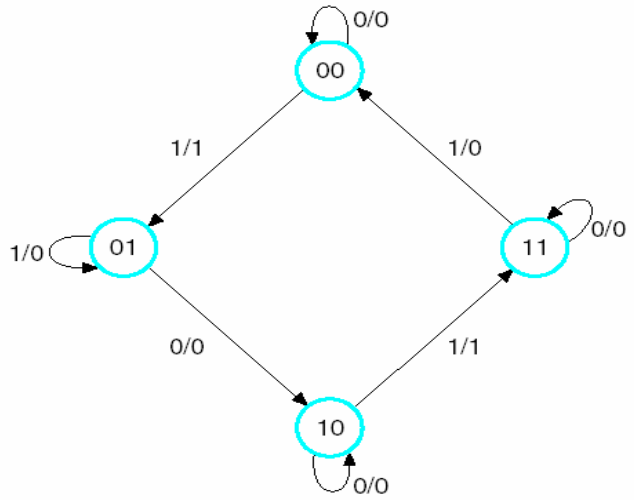



Fig. 4-23 State Diagram for Design Example

**TABLE 4-8**
**State Table for Design Example**

| Present State | | Input | Next State | | Output |
|---|---|---|---|---|---|
| A | B | X | A | B | Y |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

Table 4-8 State Table for Design Example

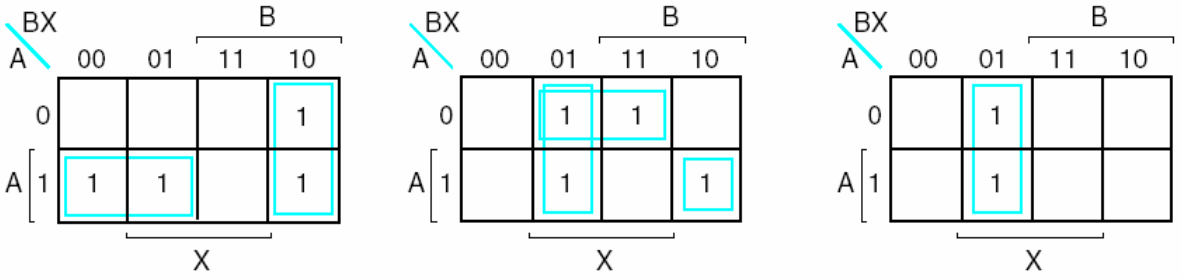



Fig. 4-24 Maps for Input Equations and Output *Y*

4 states, so 2 D FFs

# Design with D flip-flops …

$D_A = AB' + BX'$, $D_B = A'X + B'X + ABX'$, $Y = B'X$

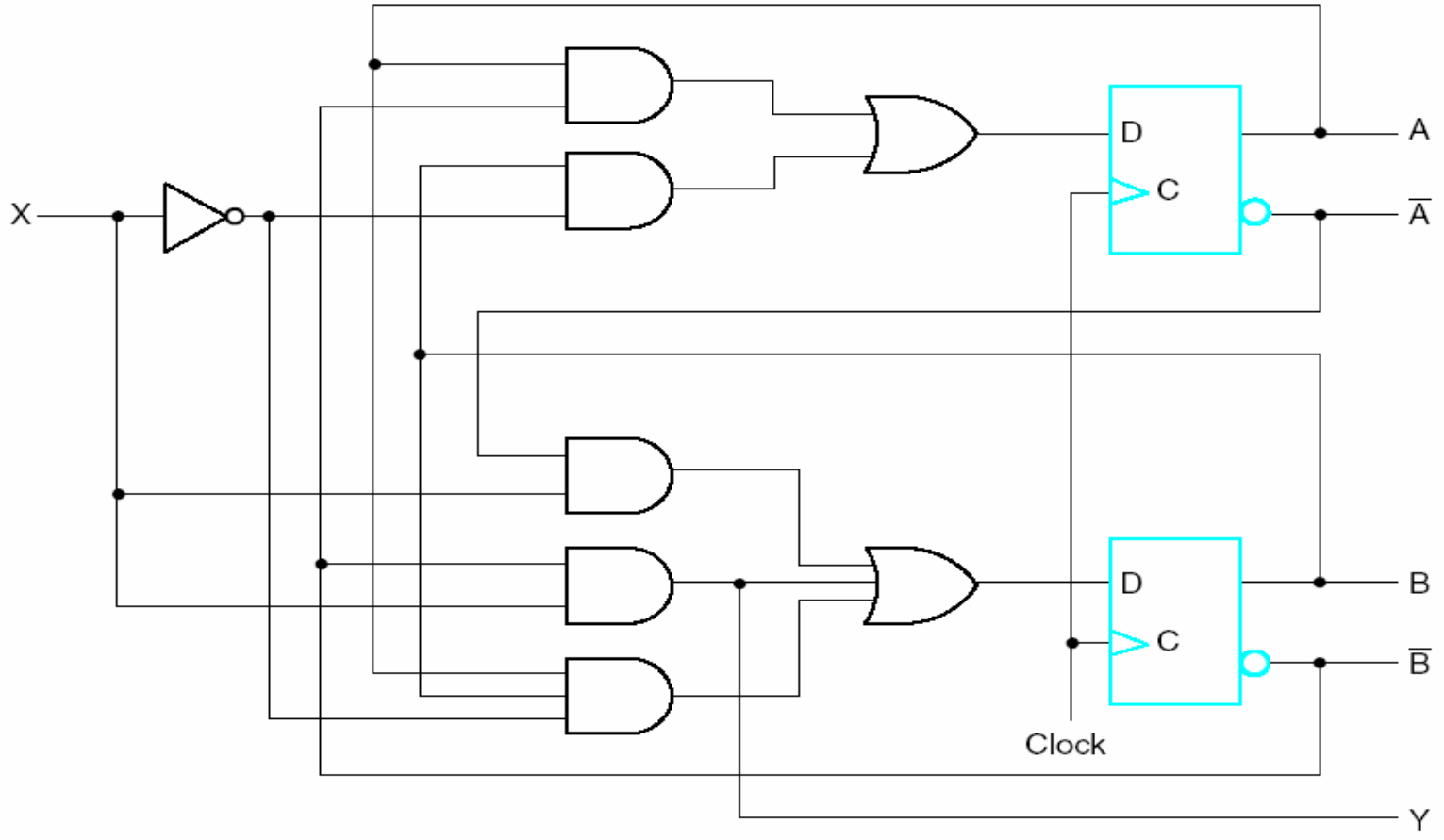



Fig. 4-25 Logic Diagram for Sequential Circuit with *D* Flip-Flops

Similarly, the design with JK FFs can be done.