Chapter 6 The Design of Sequential Systems

- Design procedure
  - Example of finding pattern 1101
- Finite state machine design
- Finite state machine example
- Design of synchronous counters

Design procedure

- Specification
- Formulation - Obtain a state diagram or state table
- State Assignment - Assign binary codes to the states
- Flip-Flop Input Equation Determination - Select flip-flop types and derive flip-flop equations from next state entries in the table
- Output Equation Determination - Derive output equations from output entries in the table
- Optimization - Optimize the equations
- Technology Mapping - Find circuit from equations and map to flip-flops and gate technology
- Verification - Verify correctness of final design

Specification

- Component Forms of Specification
  - Written description
  - Mathematical description
  - Hardware description language*
  - Tabular description*
  - Equation description*
  - Diagram describing operation (not just structure)*

- Relation to Formulation
  - If a specification is rigorous at the binary level (marked with * above), then all or part of formulation may be completed

Formulation: Finding a state diagram

- A state is an abstraction of the history of the past applied inputs to the circuit (including power-up reset or system reset).
  - The interpretation of "past inputs" is tied to the synchronous operation of the circuit. E. g., an input value (other than an asynchronous reset) is measured only during the setup-hold time interval for an edge-triggered flip-flop.
- Examples:
  - State A represents the fact that a 1 input has occurred among the past inputs.
  - State B represents the fact that a 0 followed by a 1 have occurred as the most recent past two inputs.

Formulation: Finding a state diagram

- In specifying a circuit, we use states to remember meaningful properties of past input sequences that are essential to predicting future output values.

- A sequence recognizer is a sequential circuit that produces a distinct output value whenever a prescribed pattern of input symbols occur in sequence, i.e, recognizes an input sequence occurrence.

- We will develop a procedure specific to sequence recognizers to convert a problem statement into a state diagram.

- Next, the state diagram, will be converted to a state table from which the circuit will be designed.

Sequence Recognizer Procedure

- To develop a sequence recognizer state diagram:
  - Begin in an initial state in which NONE of the initial portion of the sequence has occurred (typically "reset" state).
  - Add a state that recognizes that the first symbol has occurred.
  - Add states that recognize each successive symbol occurring.
  - The final state represents the input sequence occurrence.
  - Add state transition arcs which specify what happens when a symbol *not* in the proper sequence has occurred.
  - Add state transition arcs which specify substrings.

- The last step is required because the circuit must recognize the input sequence *regardless of where it occurs within the overall sequence applied since "reset."*.

State Assignment

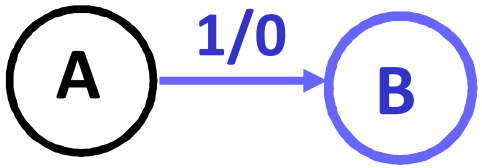
- Each of the m states must be assigned a unique code
- Minimum number of bits required is n such that
  $n \geq \lceil \log_2 m \rceil$
  where $\lceil x \rceil$ is the smallest integer $\geq x$
- There are useful state assignments that use more than the minimum number of bits
- There are 2^n - m unused states

Sequence Recognizer Example

- Example: Recognize the sequence 1101
  - Note that the sequence 1111101 contains 1101 and "11" is a proper sub-sequence of the sequence.
- Thus, the sequential machine must remember that the first two one's have occurred as it receives another symbol.
- Also, the sequence 1101101 contains 1101 as both an initial subsequence and a final subsequence with some overlap, i. e., 1101101 or 1101101.
- And, the 1 in the middle, 1101101, is in both subsequences.
- The sequence 1101 must be recognized each time it occurs in the input sequence.

Example: Recognize 1101

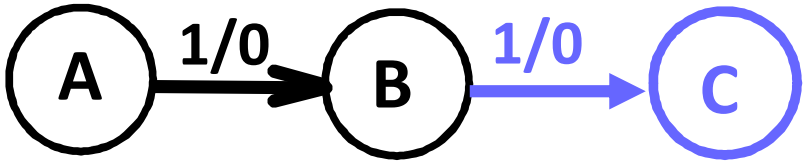
- Define states for the sequence to be recognized:
  - assuming it starts with first symbol,
  - continues through each symbol in the sequence to be recognized,
  - uses output 1 to mean the full sequence has occurred,
  - with output 0 otherwise.



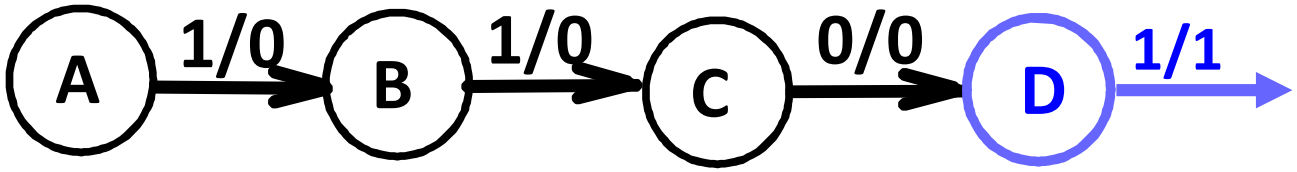
- Starting in the initial state (Arbitrarily named "A"):
  - Add a state that recognizes the first "1."
  - State "A" is the initial state, and state "B" is the state which represents the fact that the "first" one in the input subsequence has occurred. The output symbol "0" means that the full recognized sequence has not yet occurred.

Example: Recognize 1101 (continued)

- After one more 1, we have:
  - C is the state obtained when the input sequence has two "1"s.

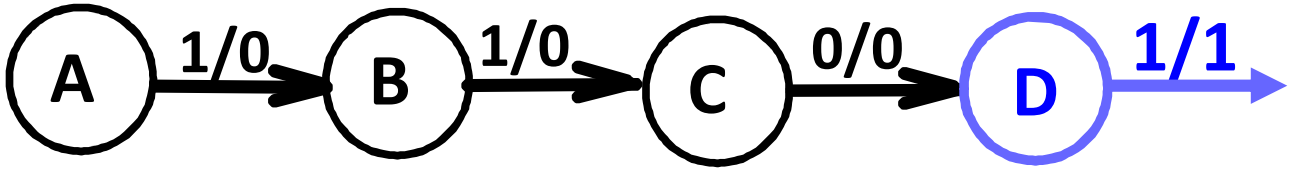


- Finally, after 110 and a 1, we have:

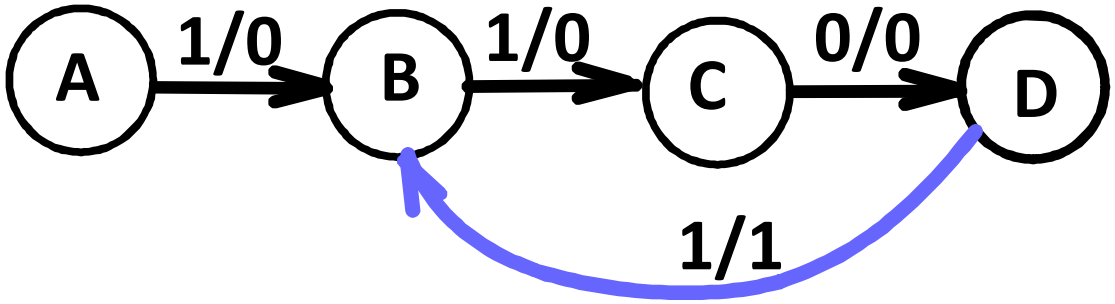


  - Transition arcs are used to denote the output function (Mealy Model)
  - Output 1 on the arc from D means the sequence has been recognized
  - To what state should the arc from state D go? Remember: 1101<u>1101</u> ?
  - Note that D is the last state but the output 1 occurs for the input applied in D. This is the case when a *Mealy model* is assumed.

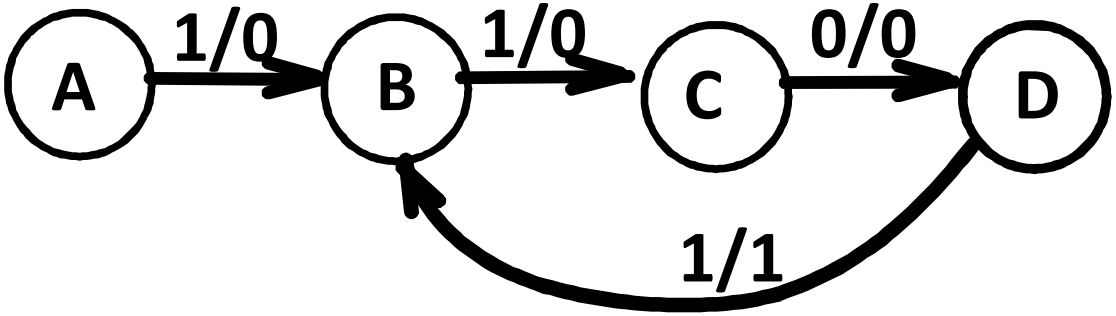
Example: Recognize 1101 (continued)



- Clearly the final 1 in the recognized sequence 1101 is a sub-sequence of 1101. It follows a 0 which is not a sub-sequence of 1101. Thus it should represent *the same state reached from the initial state after a first 1 is observed.* We obtain:



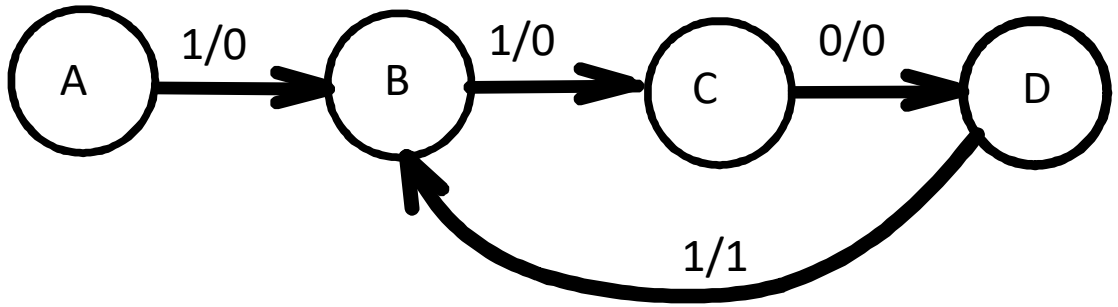
Example: Recognize 1101 (continued)



- The state have the following abstract meanings:
  - A: No proper sub-sequence of the sequence has occurred.
  - B: The sub-sequence 1 has occurred.
  - C: The sub-sequence 11 has occurred.
  - D: The sub-sequence 110 has occurred.
  - The 1/1 on the arc from D to B means that the last 1 has occurred and thus, the sequence is recognized.

Example: Recognize 1101 (continued)

- The other arcs are added to each state for inputs not yet listed. Which arcs are missing?



- Answer:

"0" arc from A

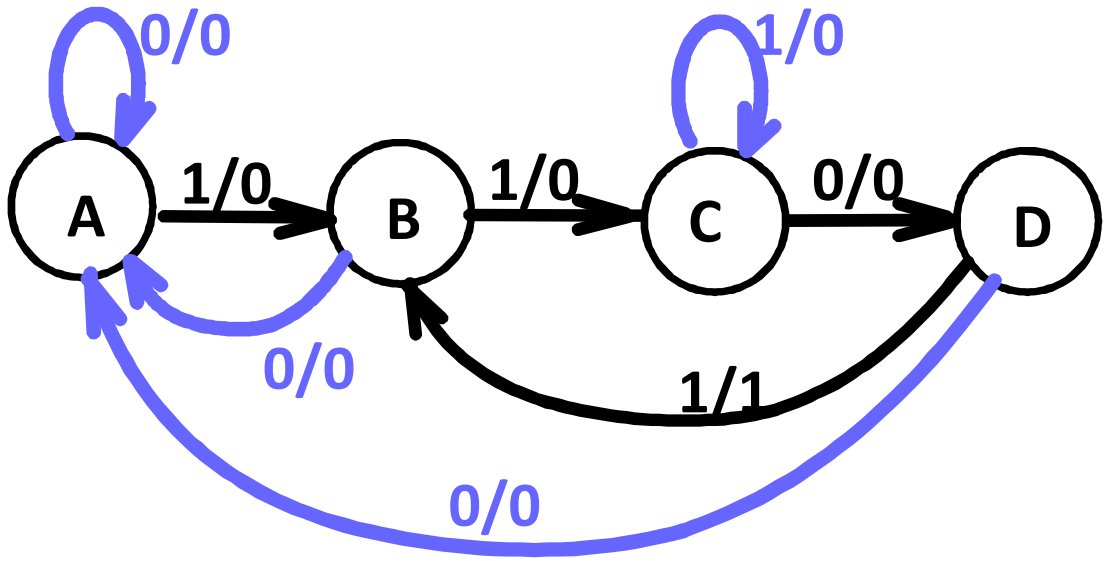
"0" arc from B

"1" arc from C

"0" arc from D.

Example: Recognize 1101 (continued)

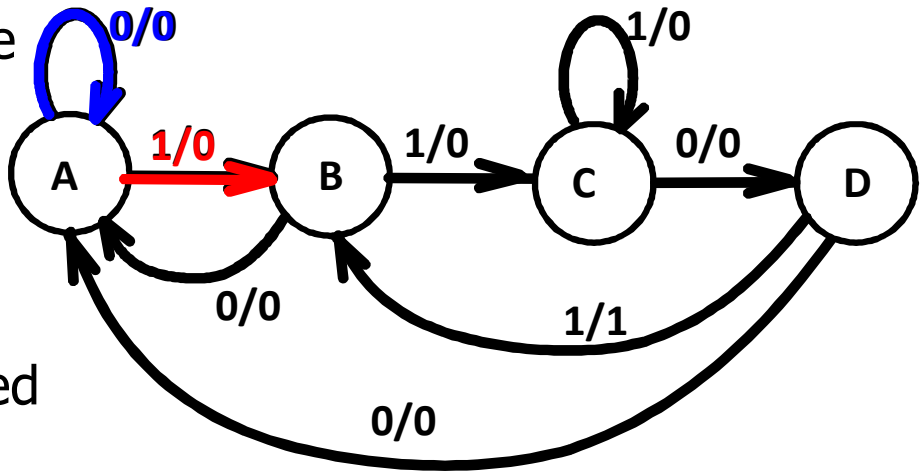
- State transition arcs must represent the fact that an input subsequence has occurred. Thus we get:



- Note that the 1 arc from state C to state C implies that State C means *two or more 1's have occurred*.

Formulation: Find State Table

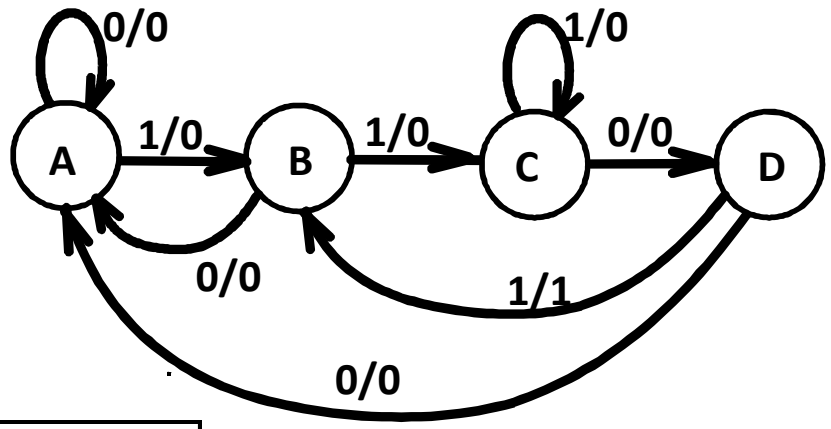
- From the State Diagram, we can fill in the State Table.
- There are 4 states, one input, and one output. We will choose the form with four rows, one for each current state.
- From State A, the 0 and 1 input transitions have been filled in along with the outputs.



Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B				
C				
D				

Formulation: Find State Table

- From the state diagram, we complete the state table.



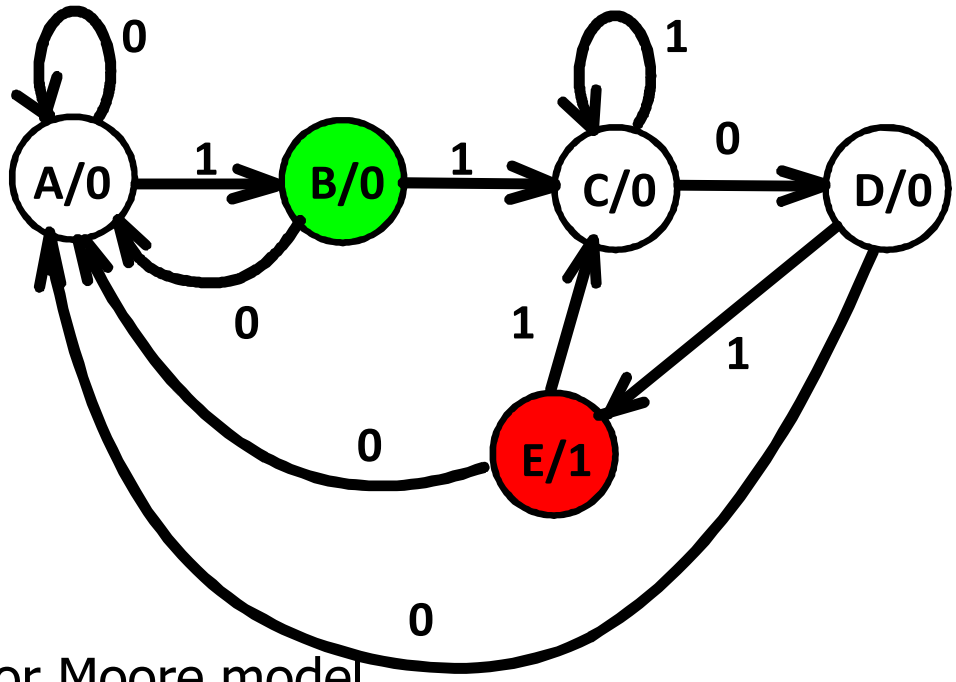
Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

- What would the state diagram and state table look like for the Moore model?

Example: Moore Model for Sequence 1101

- For the Moore Model, outputs are associated with states.
- We need to add a state "E" with output value 1 for the final 1 in the recognized input sequence.
  - This new state E, though similar to B, would generate an output of 1 and thus be different from B.
- The Moore model for a sequence recognizer usually has *more states* than the Mealy model.

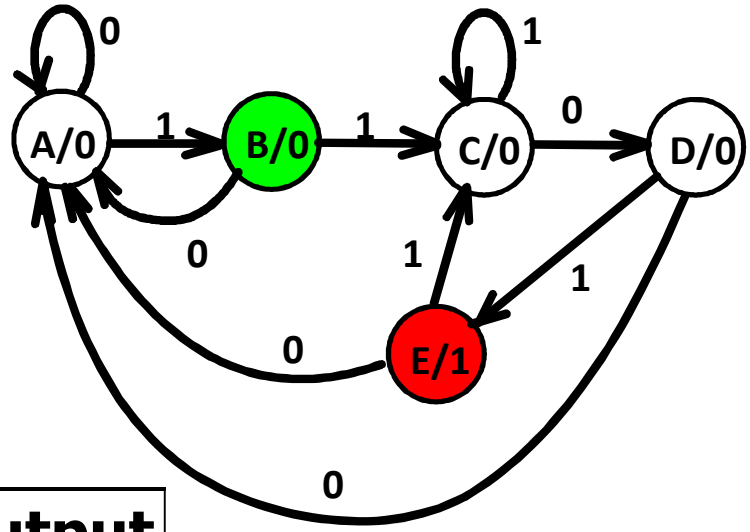
Example: Moore Model (continued)



- We mark outputs on states for Moore model
- Arcs now show only state transitions
- Add a new state E to produce the output 1
- Note that the new state, E produces the same behavior in the future as state B. But it gives a different output at the present time. Thus these states do represent a *different abstraction* of the input history.

Example: Moore Model (continued)

- The state table
- "Moore is More."



Present State	Next State x=0	Next State x=1	Output y
A	A	B	0
B	A	C	0
C	D	C	0
D	A	E	0
E	A	C	1

Finite State Machines

- A finite state machine (FSM) consists of three sets I, O, and S and two functions f and g in which:
  - I is a set of input combinations,
  - O is a set of output combinations,
  - S is a set of states
  - f is the next state function f(I, S), and
  - g is the output function f(S) [Moore model] or the output function f(I, S) [Mealy model].

- The FSM is a fundamental mathematical model used for sequential circuits.

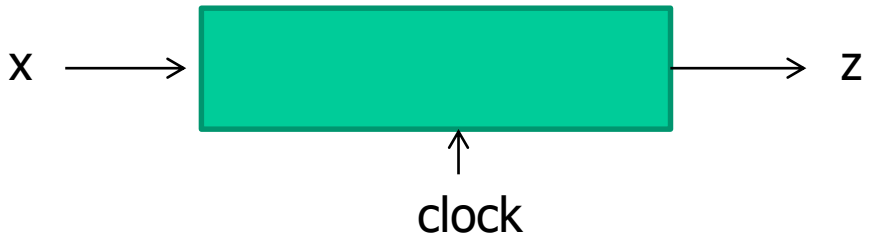
- The details of the traditional state diagrams and state tables as we have defined them are just two of many ways of representing FSMs.

Issues with Traditional State Diagram and Table Representations

- Both of these traditional representations require:
  - Enumeration of all input combinations for each state in defining next states
  - Enumeration of all input combinations for each state in defining Mealy outputs
  - Enumeration of all applicable output combinations for each state (Moore) and for each input combination-state pair (Mealy).

- For state diagrams, all Mealy outputs must be specified on transition arcs
- These requirements may be acceptable for sequential circuits with relatively few inputs, and outputs.
- For larger numbers of inputs and outputs both representations become intractable.

Finite state machine example: design a sequence detector



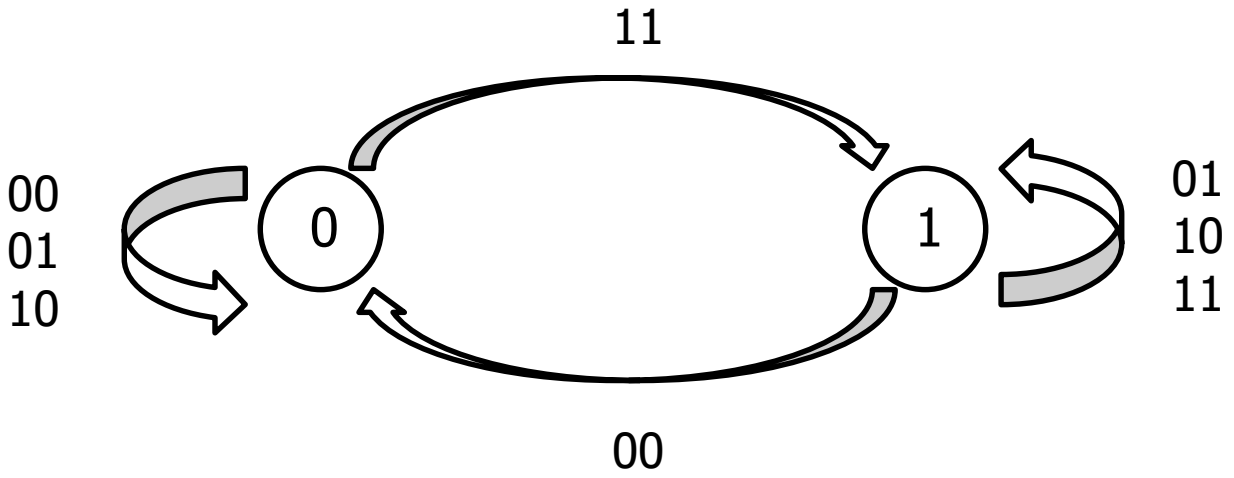
Any input sequence ending 101 will produce an output z = 1

X=001101100101010 0
Z=000001000001010 0

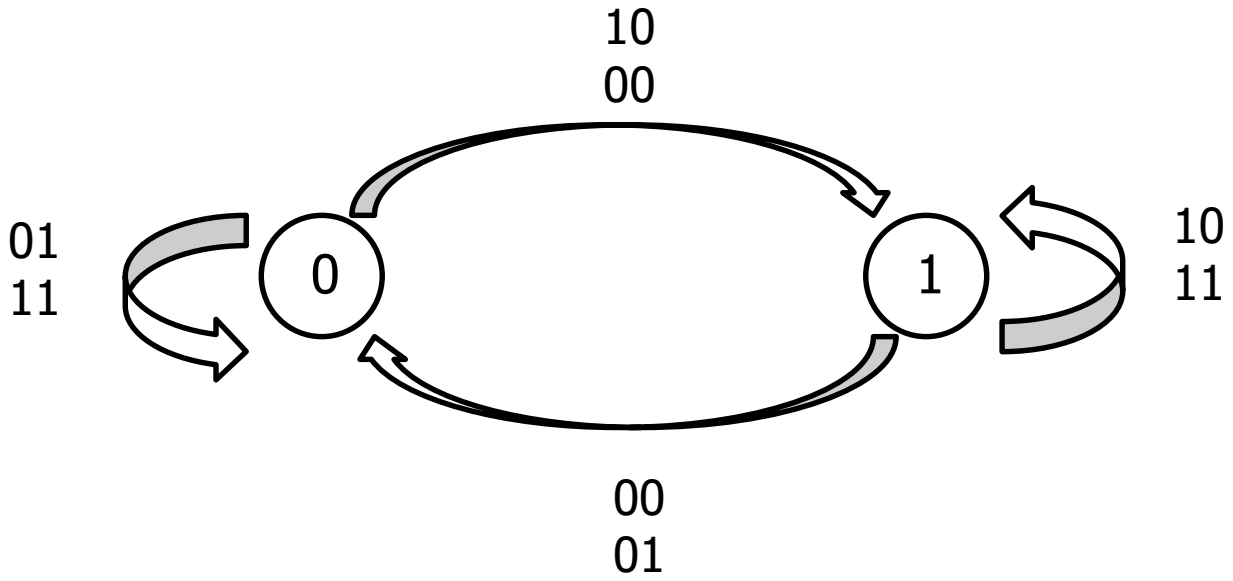
Find state diagram and state table, and design circuit

More example

Design the circuit and draw logic diagram of the sequential circuit specified by the following state diagram. Use an SR flip-flop.



Hint: This is just like a moore machine with two inputs!


10
00
01
11
0
1
10
11
00
01

Design of synchronous counter

- 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1 2 3 ……

Table 6.8 A base-16 counter.

D	*C*	*B*	*A*	*D**	*C**	*B**	*A**
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

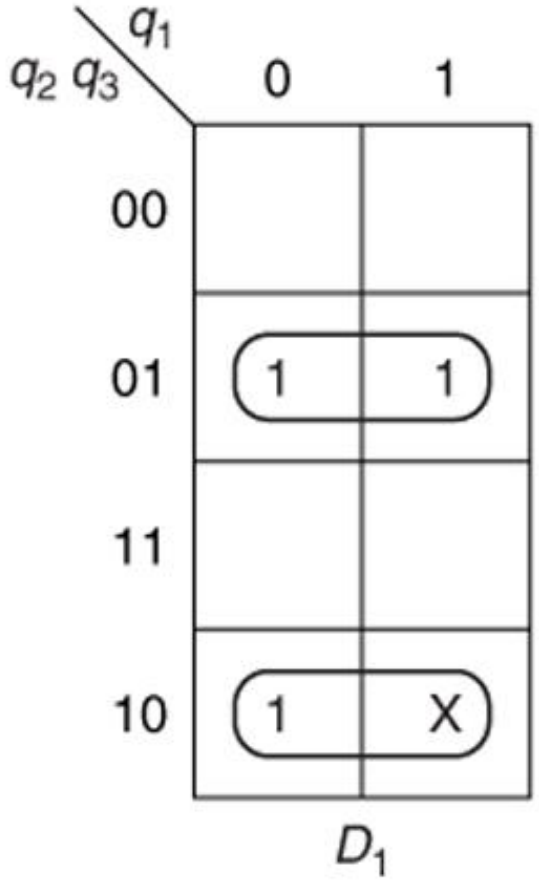
D flip-flop inputs for 16 state Counter

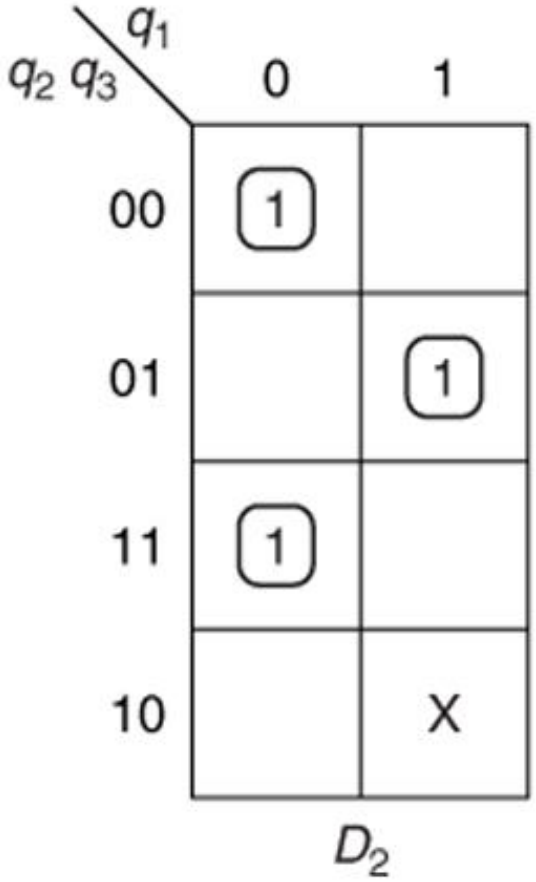
Dd = DC′ + DB′ + DA′+ D′CBA
Dc = CB′ + CA′ + C′BA
Db = B′A + BA′
Da = A′

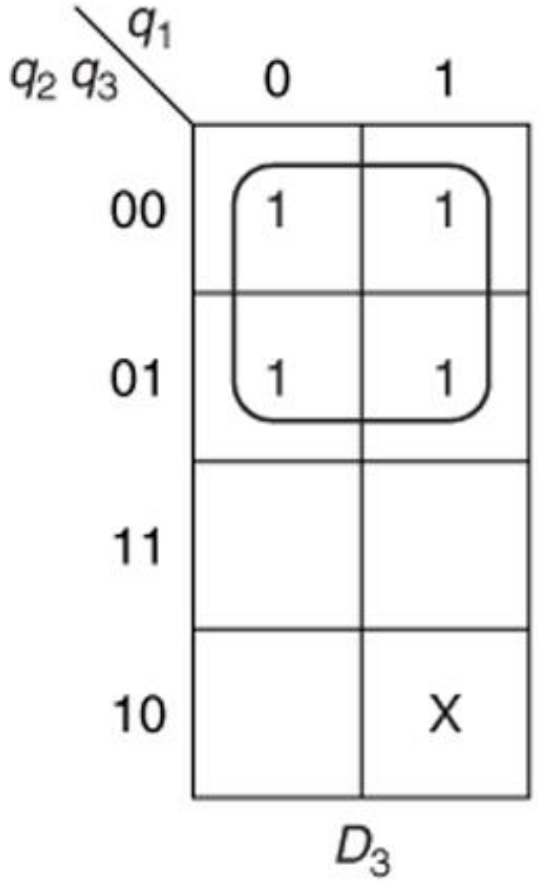
Redesign with JK flip-flops

0, 3, 2, 4, 1, 5, 7, and repeat

q_1	q_2	q_3	q_1^*	q_2^*	q_3^*
0	0	0	0	1	1
0	0	1	1	0	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	1	1	1
1	1	0	X	X	X
1	1	1	0	0	0







$$D_1 = q_2'q_3 + q_2q_3'$$

$$D_2 = q_1'q_2'q_3' + q_1'q_2q_3 + q_1q_2'q_3$$

$$D_3 = q_2'$$

state 6?

$q_1 = 1, q_2 = 1,$ and $q_3 = 0$

$D_1 = q_2' q_3 + q_2 q_3' = 00 + 11 = 1$

$D_2 = q_1' q_2' q_3' + q_1' q_2 q_3 + q_1 q_2' q_3 = 001 + 011 + 100 = 0$

$D_3 = q_2' = 0$