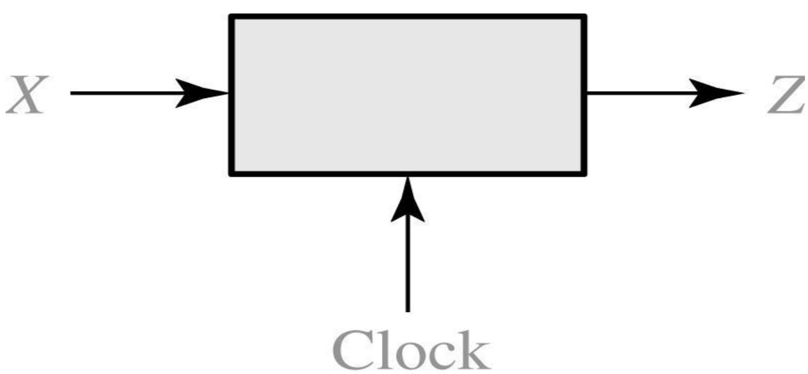
Lecture 13 Derivation of State Graphs and Tables

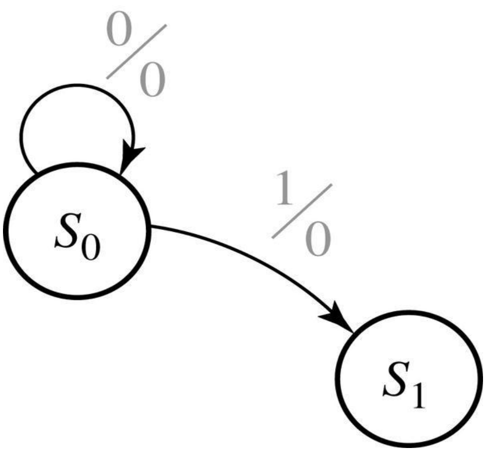
- Problem: a sequence detector. If 101 is detected, Z = 1. We use a clocked Mealy machine to design the network.

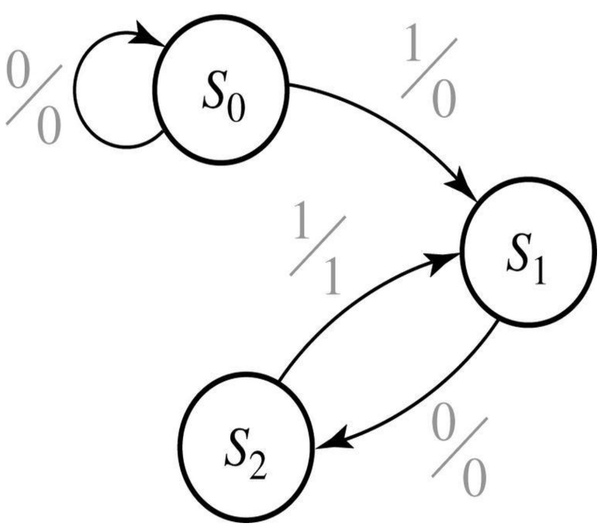


$X =$	0	0	1	1	0	1	1	0	0	1	0	1	0	1	0	0	
$Z =$	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	(14–1)
(time:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15)	

Sequence Detector

- Reset state: S_0
  - Stay in S_0 if 0 is received, go to S_1 if 1 is received. (Remember the first 1 in S_1)
  - 0/0 and 1/0 (Input/Output)
  - In S_1, if we receive a 0, then we go to another state S_2 to remember that 10 has been received.
  - In S_2, if we receive 1, then 101 is received. We must output 1. Then where should we go? Not reset (S_0). But the <u>1</u> in 10<u>1</u> may be the first 1 of the next 101. So we go back to S_1





Sequence Detector (cont.)

- In S_1, if we receive a 1, this means the restart of the <u>1</u>01 sequence, so we stay at S_1.
- In S_2 (we remember 10), we also need to consider what to do if we receive a 0. This means that a 100 is received. In this case, we go back to S_0 to reset.
- Each state has two exit lines.

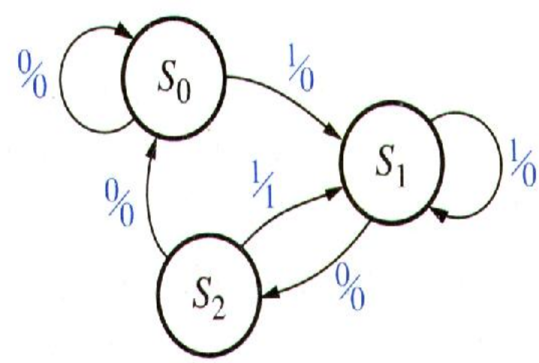


Figure 14–4
Mealy State Graph for Sequence Detector

Sequence Detector (cont.)

– Convert the state graph to a state table. For the arc between S_2 and S_1, 1/1 means that 1 output is present as soon as X becomes 1 (before the state change occurs.)

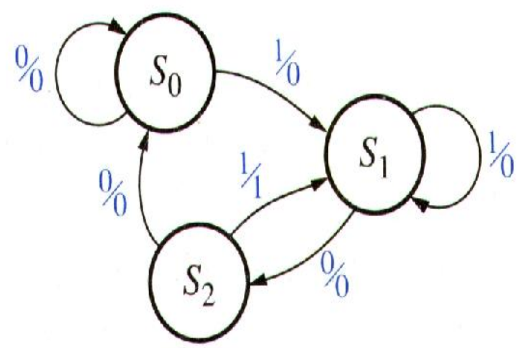


Figure 14–4
Mealy State Graph for Sequence Detector

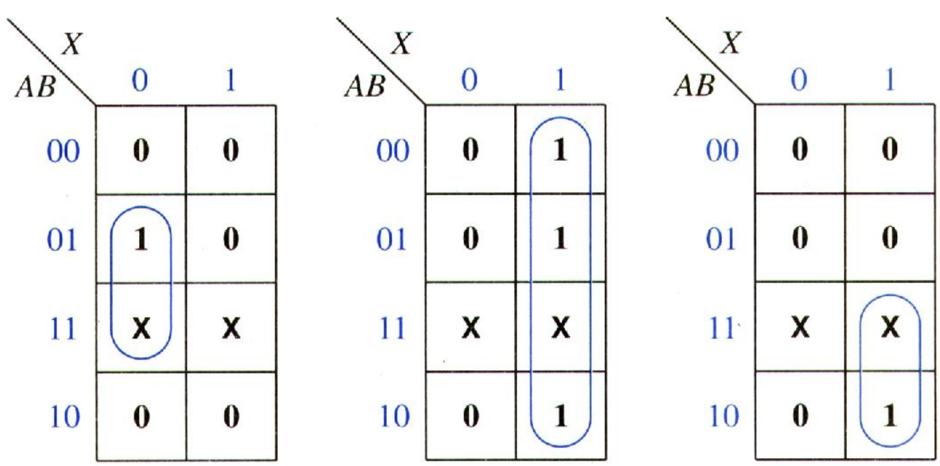
Present State	Next State $X=0$	Next State $X=1$	Present Output $X=0$	Present Output $X=1$
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_1	0	1

Sequence Detector (cont.)

– We need two FFs for 3 states.

	A^+B^+		Z	
AB	$X=0$	$X=1$	$X=0$	$X=1$
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

– We plot the next state table.



$A^+ = X'B$ $B^+ = X$ $Z = XA$

Sequence Detector (cont.)

- Suppose we use D FFs. Then the input equation of a D FF is : $D = Q^+$
- $D_A = A^+ = X'B$
- $D_B = B^+ = X$
- $Z = XA$

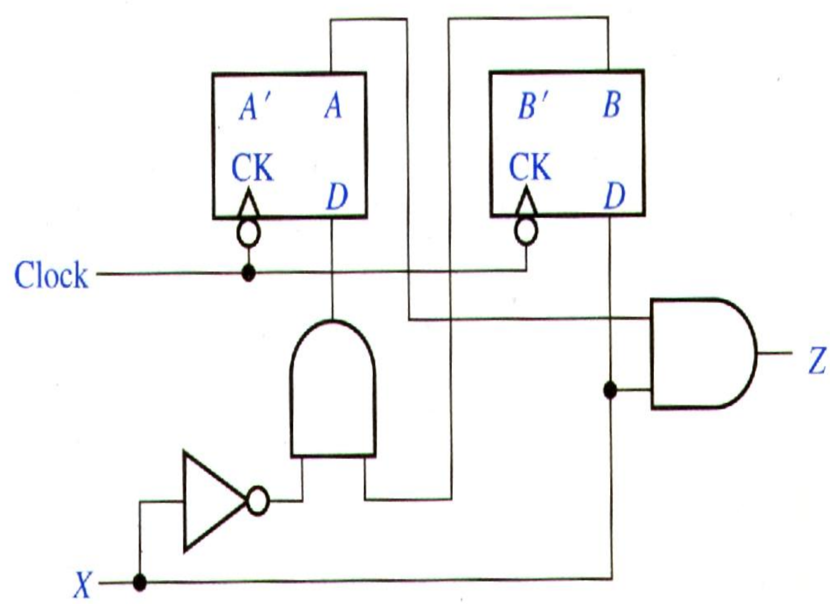
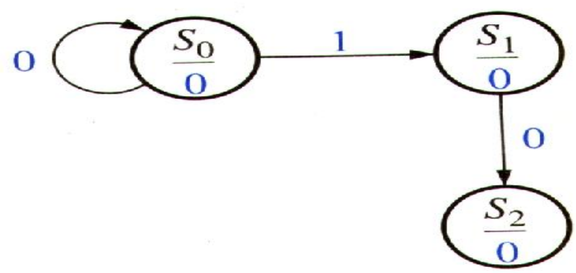


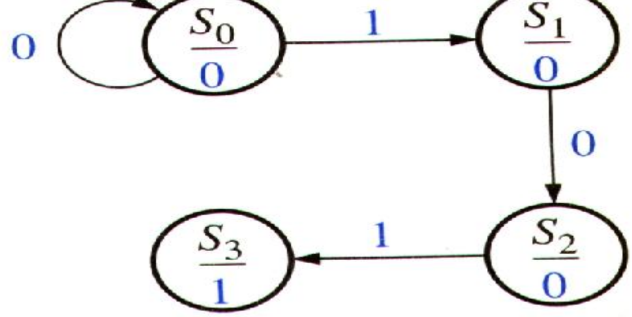
Figure 14–5

101 Detection Using a Moore Network

- First reset in S_0, if 1 is received, go to S_1.
- If a 0 is received in S_1, go to S_2 to remember 10.

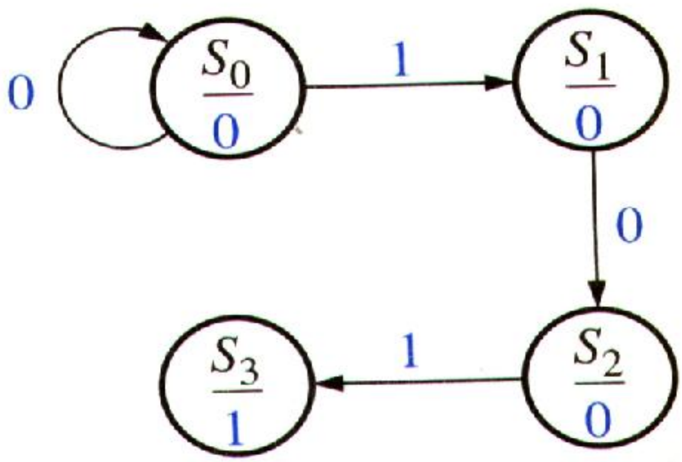


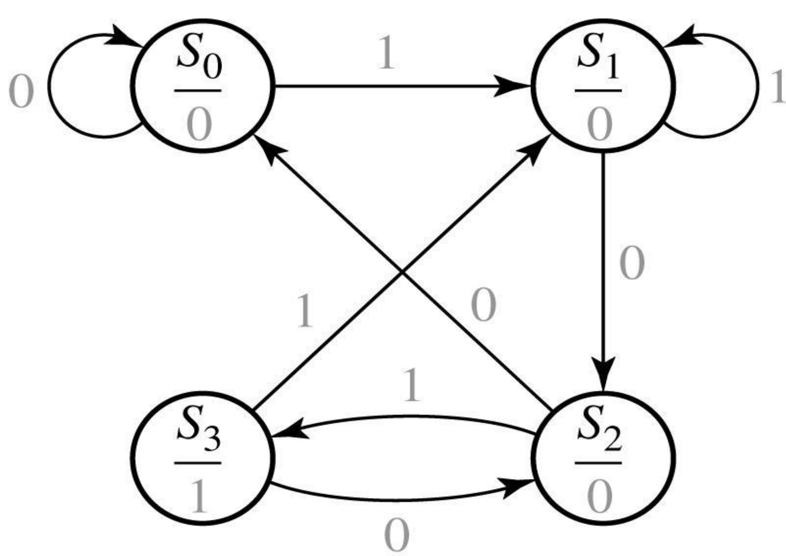
- If a 1 occurs to complete 101, we can not go back to S_1 because in S_1 the output is 0. We need to create another state S_3.



101 Detection (cont.)

– Now we complete each state with the rest of cases that have not considered yet.

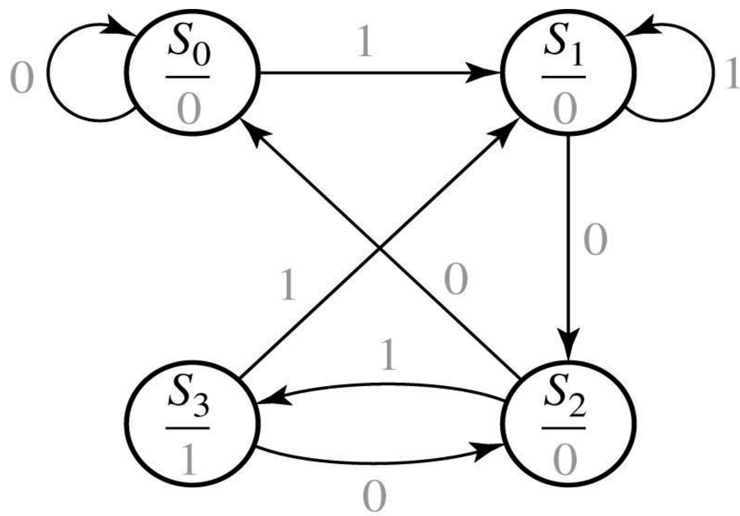




Receive 10
Come to here

101 Detection (cont.)

– Find the state table from state graph. Z = AB'.



AB	Present State	Next State $X = 0$	Next State $X = 1$	Present Output (Z)
00	S_0	S_0	S_1	0
01	S_1	S_2	S_1	0
11	S_2	S_0	S_3	0
10	S_3	S_2	S_1	1

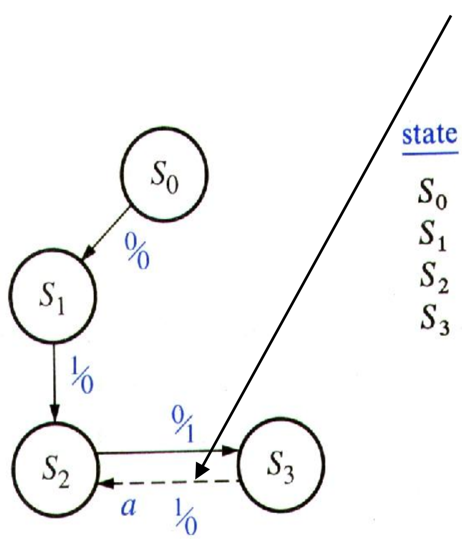
More Complex Example

- Output Z = 1 if the input sequence ends in 010 or 1001.
  - Construct some sample input and output sequences to make sure we understand the problem statement.

```
X = 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 1 0
          ↑   ↑   ↑ ↑       ↑   ↑
          a   b   c d       e   f
Z = 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0
```

More Complex (cont.)

- 010 or 1001
  - First work on the sequences that lead to a 1 output. 010 first.
    - Start at a reset state S_0 (no inputs received).
    - In S_2, the sequence ends in 01. In S_3, the sequence ends in 010.
    - In S_3, if we receive a 1, then we are in the sequence ending in 01 . 01 is remembered in S_2. So we go back to S_2.

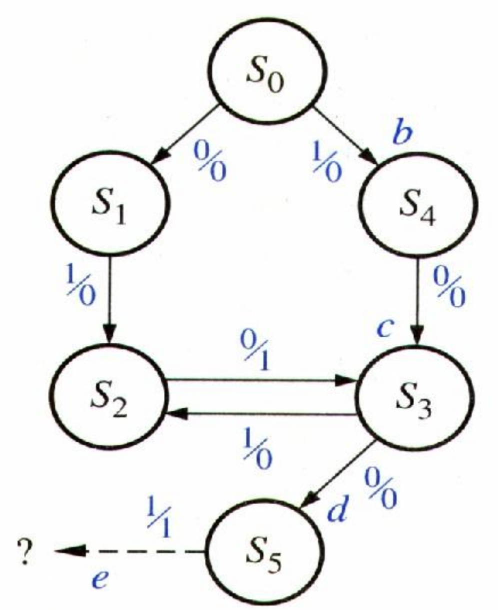


state	sequence received
S_0	reset
S_1	0
S_2	01
S_3	010

Figure 14–7

More Complex (cont.)

- Now 1001
  - Start at reset state S_0. If we receive 1, we go to S_4 to remember that the first 1 in 1001 is received.
  - Then 0 is received; this means a sequence ending in 10.
    - » Since S_3 represents sequences ends in (0)10, so we go to S_3 instead of creating a new state.
  - In S_3, 0 is received. We create a new state S_5 to remember 100.

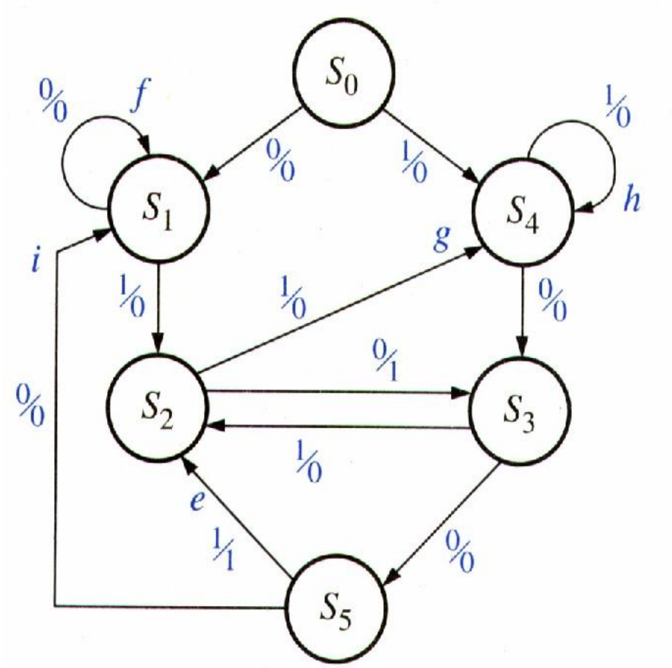


state	sequence ends in
S_0	reset
S_1	0 **(but not 10)**
S_2	01
S_3	010 or 10
S_4	1
S_5	100

Complex Example (cont.)

- In S_5, if we received a 1, we complete the sequence 1001. Since 1001 ends in 01, we go back to S_2 from S_5 if 1 is received.

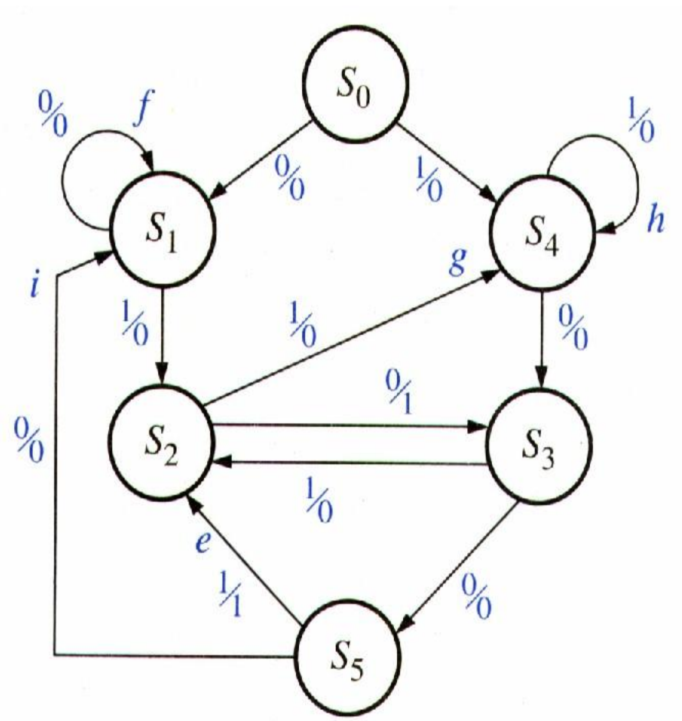
- Patch up the rest:
  - In S_1, 1 is already considered. 0 occurs for input (x). This is 00. No matter how many 0's occur, the sequence ends in 0. So we stay at S_1. The same as in S_4.



state	sequence ends in
S_0	reset
S_1	0
S_2	01
S_3	010 or 10
S_4	1
S_5	100

Complex Example (cont.)

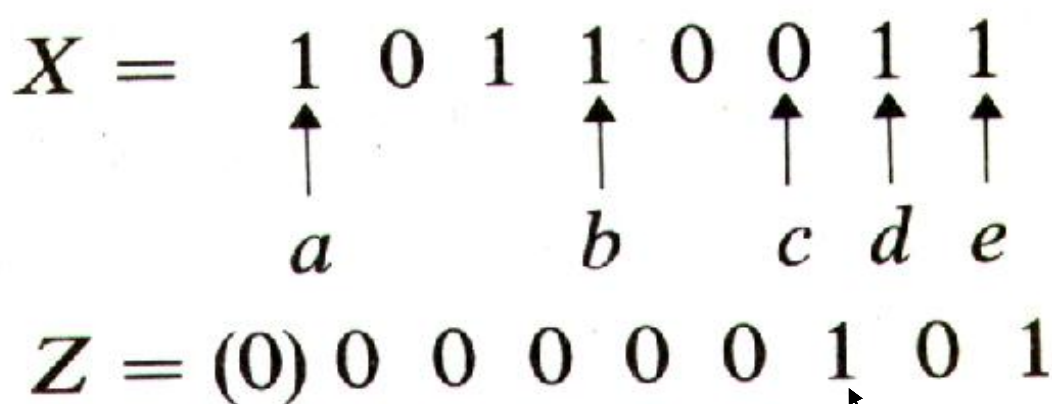
- Patch up the rest: (out-going line)
  - In S_2, input 0 has considered. For input 1, then 11 occurs. 11 does not appear in 010 or 1001. So we don't need another state. Since 11 ends in 1, so we go to S_4.
  - In S_5, if we get a 0 input, then the sequence ends in 000. 000 is not in 010 or 1001. 000 ends in 0. So we go back to S_1.



state	sequence ends in
S_0	reset
S_1	0
S_2	01
S_3	010 or 10
S_4	1
S_5	100

Moore Network Example

- Problem: input X and output Z. Z = 1 if the total number of 1's received is odd and at least two consecutive 0's have been received.
  - Z only changes after the next active clock edge. (Moore machine example)



Wait until clock edge

Moore Network Example (cont.)

- Start with a reset state S_0 with 0 output.
  - Two states to remember odd number of 1's and even number of 1's received respectively.
    - Output of S_1 is 0 since two consecutive 0's have not been received.

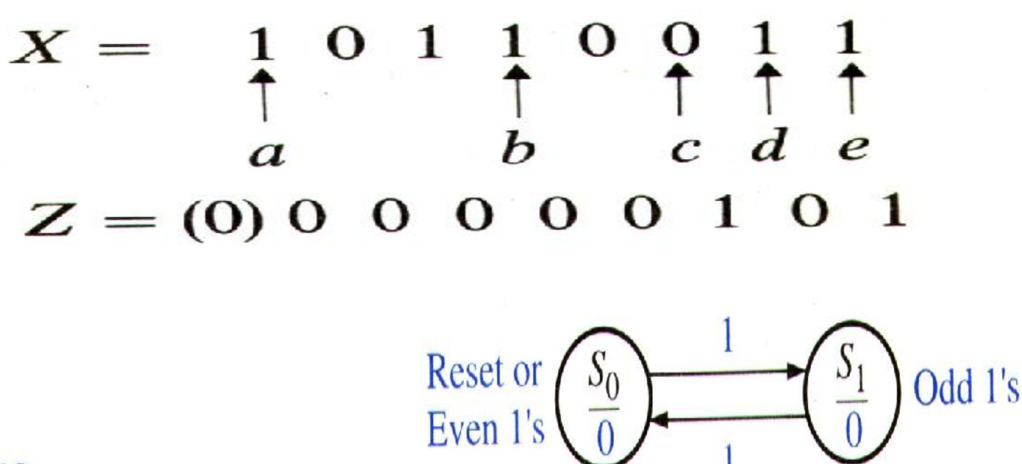


Figure 14–10

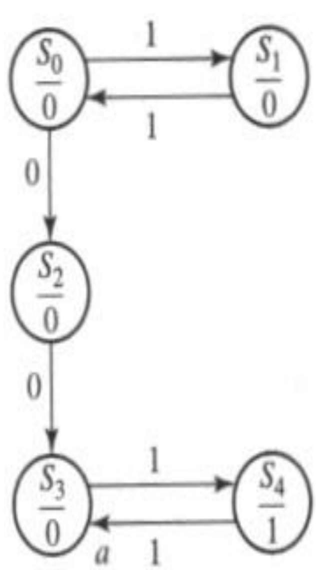
Moore Network Example (cont.)

– In S_0, if we receive a 0, then the first 0 of sequence of 00 starts. We go to S_2. (S_2: even 1's and 0).

– Another 0 takes us to S_3 (even 1's and 00).

– In S_3, if we receive 1, then 00 and odd number of 1's occurs. We go to S_4 and set output =1.

FIGURE 14-11



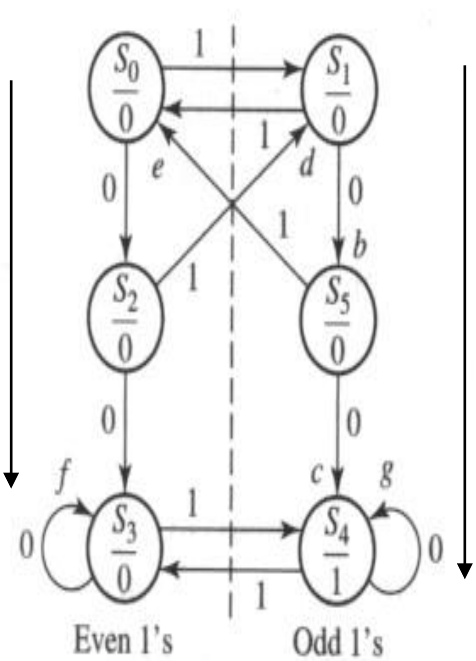
state	sequence received
S_0	reset or even 1's
S_1	odd 1's
S_2	even 1's and ends in 0
S_3	even 1's and 00 has occurred
S_4	00 has occurred and odd 1's

Moore Network Example (cont.)

- In S_4, if we receive 1, then 00 and even number of 1's occurs. We go back to S_3. In the same way, we can construct S_5 and the rest of the output going line in each state.

FIGURE 14-12

Get 00 first then odd one



state	input sequences
S_0	reset or even 1's
S_1	odd 1's
S_2	even 1's and ends in 0
S_3	even 1's and 00 has occurred
S_4	odd 1's and 00 has occurred
S_5	odd 1's and ends in 0

Get odd one first then 00.

Guideline for Construction of State Graph

- Construct sample input and output to understand the problem.
- Determine the initial state.
- Construct partial graph according to the sequences that lead to a nonzero output.
- Check to see if an arrow should go a new state or a previously defined state.
- Check if the input sequences and output sequences match the requirement when the graph is complete.

More example

- Problem: input X and output Z. If input forms 0101 or 1001, then Z = 1. The network resets after every four inputs. Find the Mealy state graph.

X = <u>01</u>01   0010   <u>10</u>01   0100

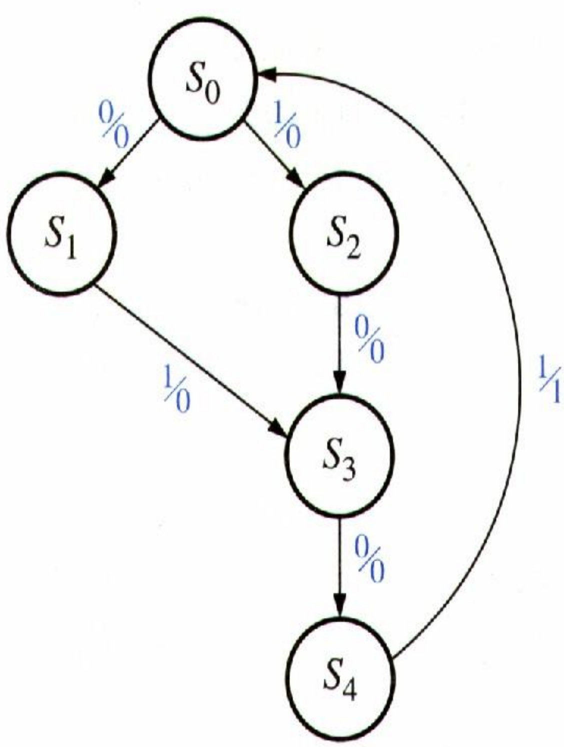
Z = 0001 0000   0001   0000

reset   reset   reset

Note: If 01 or 10 followed by 01, then Z = 1.

More example (cont.)

- 0101 or 1001. If 01 or 10 followed by 01, then Z = 1.
- This partial graph shows 0101 and 1001 sequences.

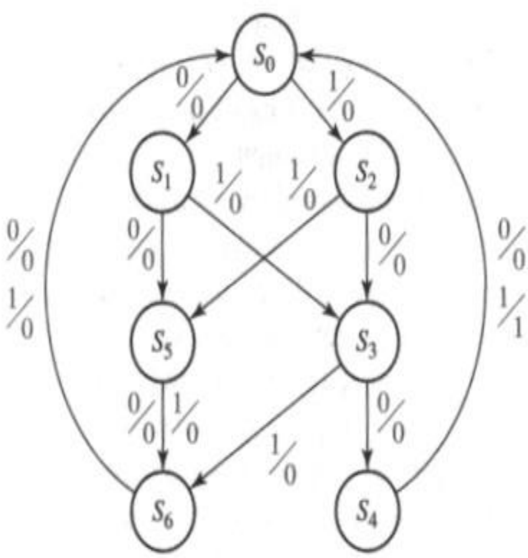


state	sequence received
S_0	reset
S_1	0
S_2	1
S_3	01 or 10
S_4	010 or 100

More example (cont.)

- Wrap up the rest
  - Use S_5 and S_6 to accommodate the rest of 4-bit sequences. For S_5, either 00 or 11 is received. No output of 1 is possible until the network is reset.

FIGURE 14-14 Complete State Graph for Example 1



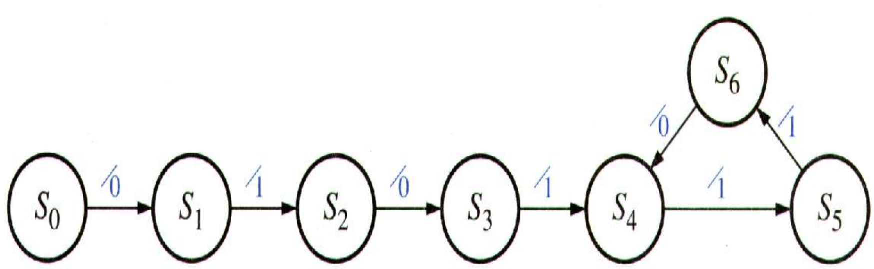
state	sequence received
S_0	reset
S_1	0
S_2	1
S_3	01 or 10
S_4	010 or 100
S_5	two inputs received, no 1 output is possible
S_6	three inputs received, no 1 output is possible

Mealy Machine Example

- A sequential network that generates the output sequence 0101 110 110 110 …..
- Homework: Realize the network.

EXAMPLE 2 Find the Mealy state graph for a sequential network which generates the output sequence 0101 110 110 110 . . .

Solution:



(A blank space above the slash indicates that the network has no input other than the clock.)

Moore Machine Example

- Multiple inputs
  - Assign previous inputs to states.

EXAMPLE 3 A sequential network has two inputs (X_1, X_2) and one output (Z). The output remains a constant value unless one of the following input sequences occurs:

(a) The input sequence $X_1X_2 = 01, 11^2$ causes the output to become 0.
(b) The input sequence $X_1X_2 = 10, 11$ causes the output to become 1.
(c) The input sequence $X_1X_2 = 10, 01$ causes the output to change value.

Derive a Moore state graph for the network.

Solution: The only sequences of input pairs which affect the output are of length two. Therefore, the previous and present inputs will determine the output, and the network must "remember" only the previous input pair. At first it appears that three states are required, corresponding to the last input received being $X_1X_2 = 01$, 10 and (00 or 11). Note that it is unnecessary to use a separate state for 00 and 11 since neither input starts a sequence which leads to an output change. However, for each of these states the output could be either 0 or 1, so we will initially define six states as follows:

Previous Input (X_1X_2)	Output (Z)	State Designation
00 or 11	0	S_0
00 or 11	1	S_1
01	0	S_2
01	1	S_3
10	0	S_4
10	1	S_5

Moore Machine Example (cont.)

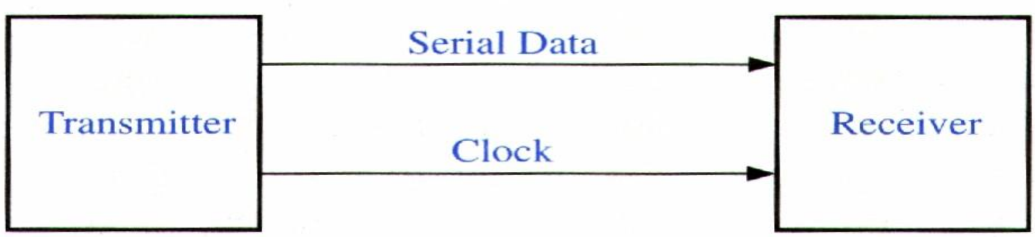
- Derive state table
  - For S_4, if 00 is received, the input sequence is 10,00, the output does not change. We go to S_0 to remember that the last input received was 00.
  - If 01 is received at S_4, then 10,01 is received. Then Z (= 0) is changed to 1. And we go to S_3 to remember that last input was 01.

Previous Input (X_1X_2)	Output (Z)	State Designation
00 or 11	0	S_0
00 or 11	1	S_1
01	0	S_2
01	1	S_3
10	0	S_4
10	1	S_5

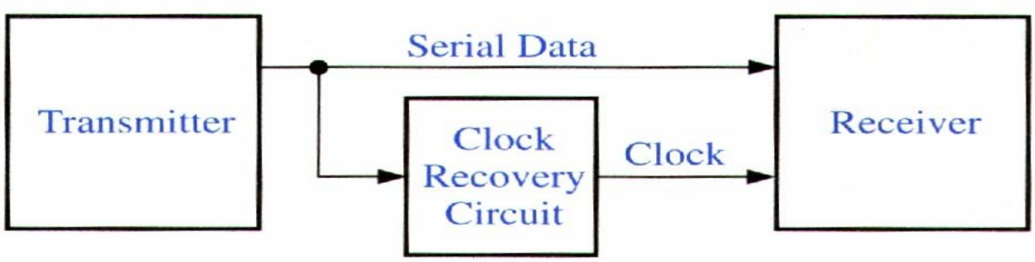
Present State	Z	Next State $X_1X_2 = 00$	01	11	10
S_0	0	S_0	S_2	S_0	S_4
S_1	1	S_1	S_3	S_1	S_5
S_2	0	S_0	S_2	S_0	S_4
S_3	1	S_1	S_3	S_0	S_5
S_4	0	S_0	S_3	S_1	S_4
S_5	1	S_1	S_2	S_1	S_5

Serial Data Code Conversion

- General block diagram
  - Data and clock transmitted separately.
  - (Clock + Data) transmitted as a signal.
    - Need a digital phase-locked loop circuit to regenerate the clock signal at the receiver end.



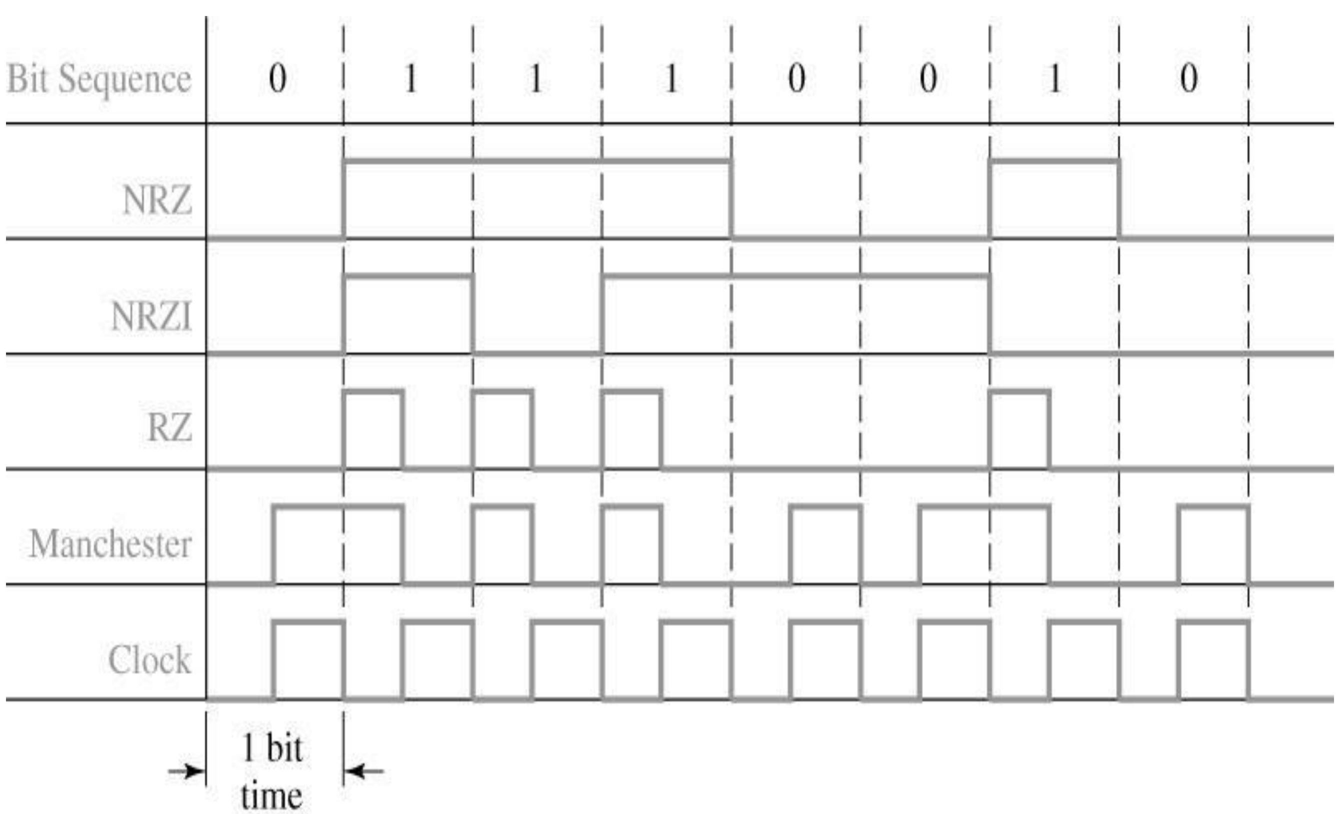
(a)



(b)

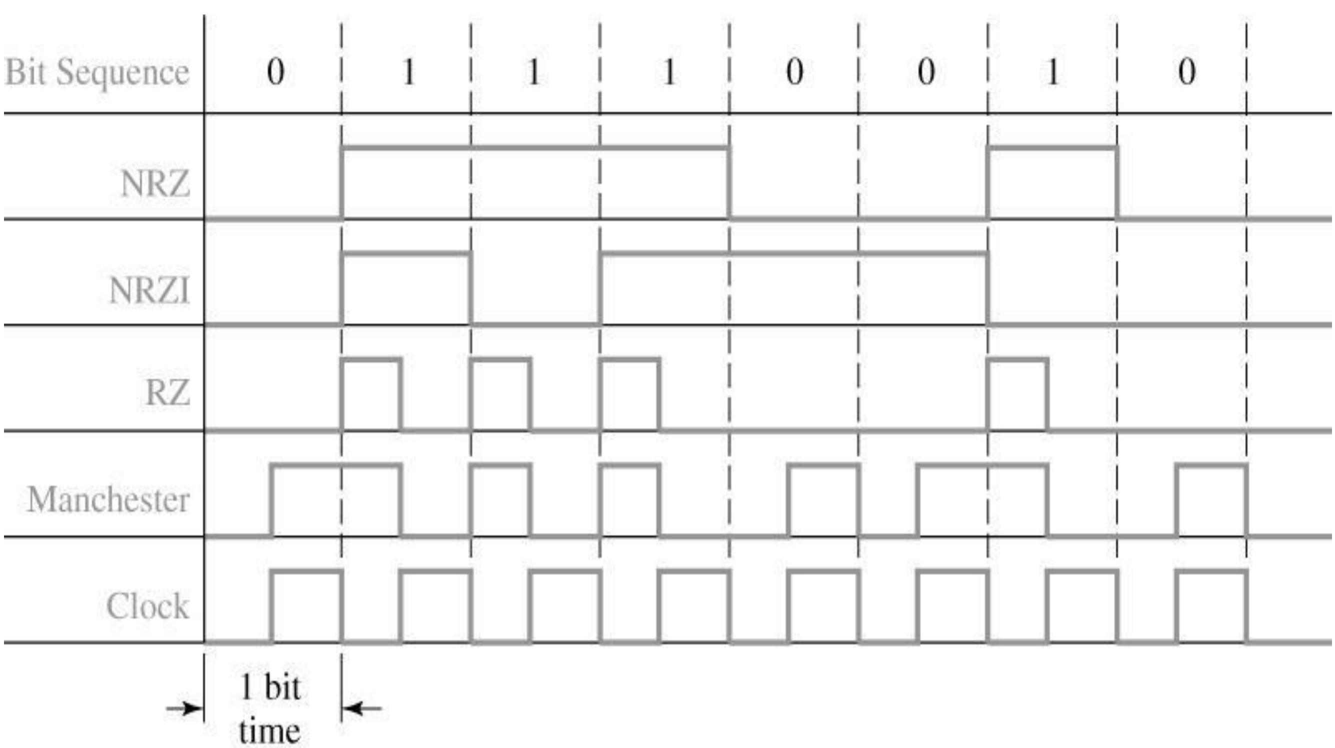
Coding Schemes

- NRZ (non-return-to-zero):
  - Each bit is transmitted for one bit time without any change.
- NRZI (non-return-to-zero Inverted-on-1s)
  - A 0 is encoded by no change in the transmitted value, and a 1 is encoded by inverting the previous transmitted value. (0 no change of previous value, 1 inverting previous value)



Coding Schemes (cont.)

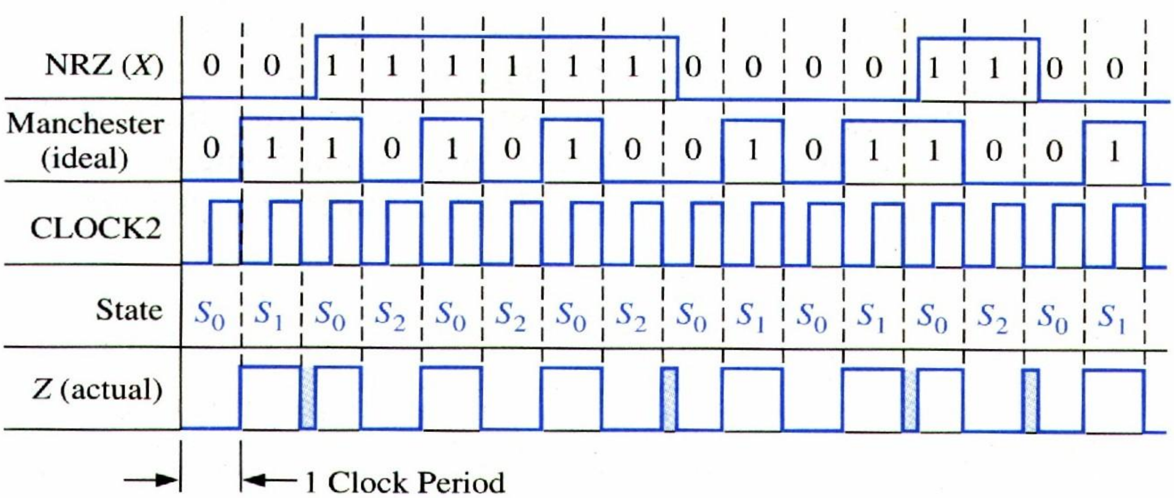
- RZ (return-to-zero):
  - A 0 is transmitted as NRZ, but a 1 is transmitted as a 1 for the first half of the bit time and signal returns to 0 for the second half.
- Manchester
  - A 0 is encoded as a 0-to-1 transition in the middle of the bit time and a 1 is encoded as a 1-to-0 transition.
    - » Ethernet (10/100 Mbps)



Code conversion Network
(Mealy machine)

- Convert a NRZ-coded bit stream to a Manchester-coded bit stream.
  - Clock2 is twice the frequency of the basic block.
  - After each conversion, reset to S_0.

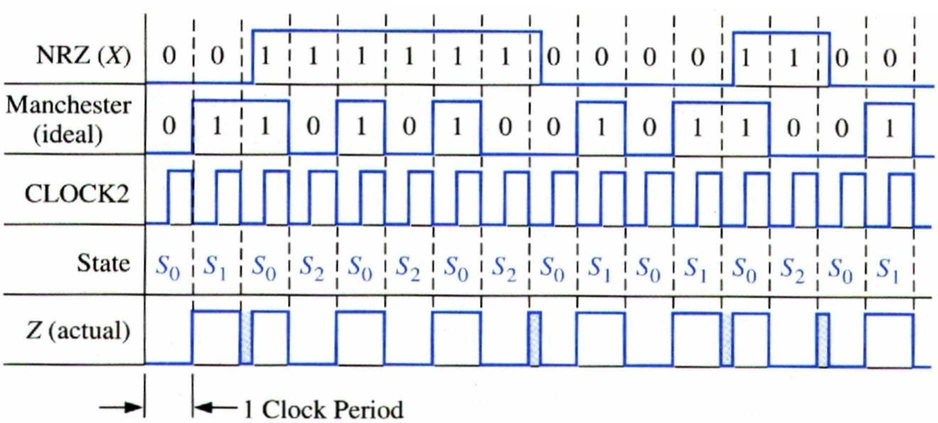
(a) Conversion network



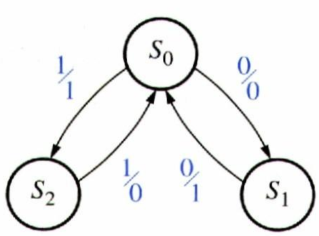
(b) Timing chart

Code conversion Network

- State graph and table
  - In S_1, X = 1 does not occurs because X = 00 seen from CLOCK2.
  - In S_2, X = 0, does not occurs because X = 11. Treat them as don't care.
    - » Note that glitch occurs if X is delayed w.r.t. basic clock.



(b) Timing chart



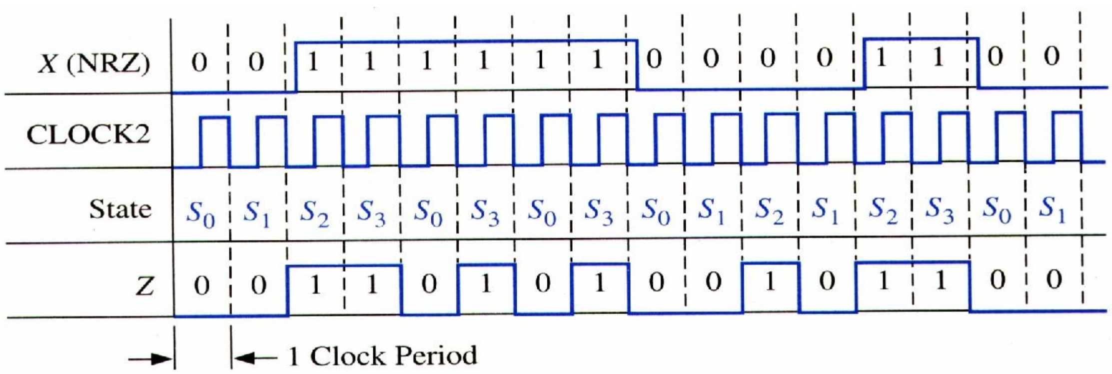
(c) State graph

Present State	Next State $X = 0$	Next State $X = 1$	Output (Z) $X = 0$	Output (Z) $X = 1$
S_0	S_1	S_2	0	1
S_1	S_0	–	1	–
S_2	–	S_0	–	0

(d) State table

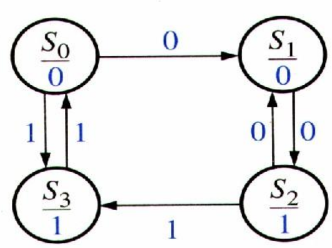
Code conversion Network
(Moore Machine)

- Moore State graph and table
  - Output is delayed by one clock. (why?)
  - 1 input cannot occur in S_1.
  - 0 input can not occur in S_3.
  - Work on 00 then work on 11. Then patch up the rest.



(a) Timing chart

This can not be the first state for the first NRZ = 1. So either for the first 1 or 0, the output is delayed for one clock.



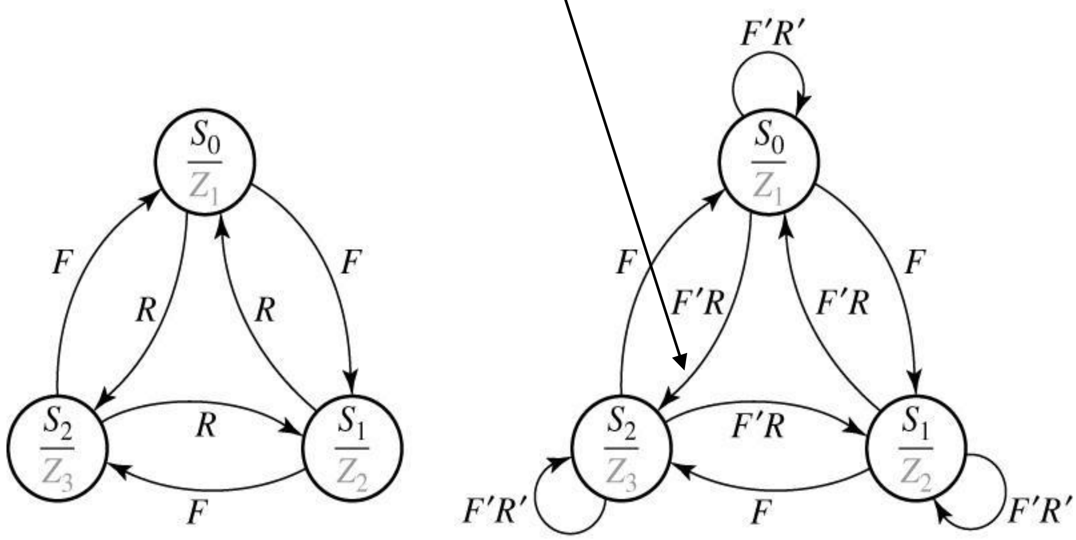
(b) State graph

Present State	Next State $X = 0$	Next State $X = 1$	Present Output (Z)
S_0	S_1	S_3	0
S_1	S_2	–	0
S_2	S_1	S_3	1
S_3	–	S_0	1

(c) State table

State graph with variable names

- In (a), all F’s (forward) for input sequence, output = $Z_1Z_2Z_3Z_1Z_2Z_3$… and all R’s for reverse output. (a) is not properly specified.
- In state S0 what if F = 1 and R =1? We must resolve this by assuming F has a high priority, for instance.
- In (b), R is changed to F’R. S0 to S2 if F = 0 and R = 1.



(a) (b)

Assuming input F takes priority over input R

TABLE 14-8
State Table for
Figure 14-22

PS	NS FR = 00	01	10	11	Output $Z_1Z_2Z_3$
S_0	S_0	S_2	S_1	S_1	1 0 0
S_1	S_1	S_0	S_2	S_2	0 1 0
S_2	S_2	S_1	S_0	S_0	0 0 1

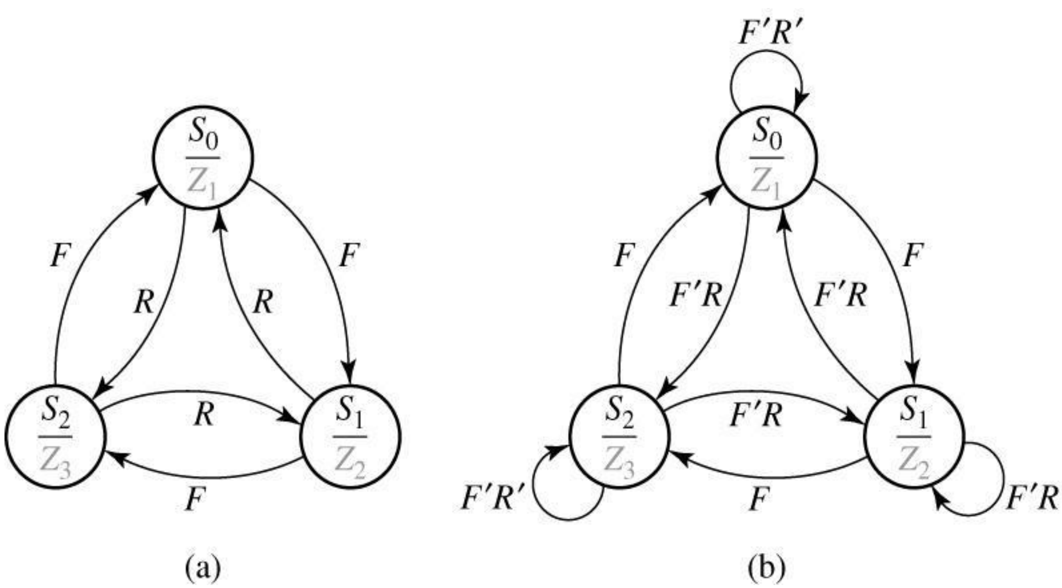
The same

Completely specified state graph

– OR all the labels emanating from a state, the result is 1. (output arcs of a state)
  - In S_0, F + F'R + F'R' = 1
  - For every input combination, at least one next state is defined. One of the labels must be true.

– AND any pair of the labels on arcs emanating from a state, the result is 0.
  - In S_0, F. F'R = 0, F.F'R' = 0, F'R.F'R' = 0
  - For every input combination, no more than one next state is defined. (no more than two 1's is defined, i.e., so will not go to two states.)

– If both are true, then exactly one next state is defined.



Incompletely Specified Graph

- If we know certain input combinations cannot occur, then an incompletely specified graph is acceptable!
  - For example, if $F = 1$, R must be 0 and if $R = 1$, F must be 0.