Switching Circuits & Logic Design

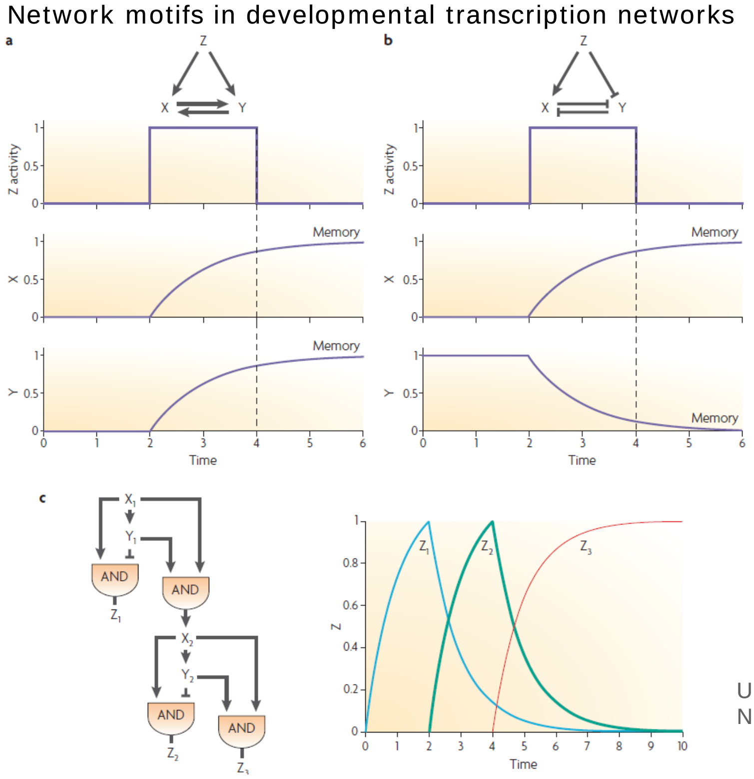
Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University

Fall 2013

§14 Derivation of State Graphs and Tables

Network motifs in developmental transcription networks



Uri Alon
Nature Reviews Genetics, June 2007

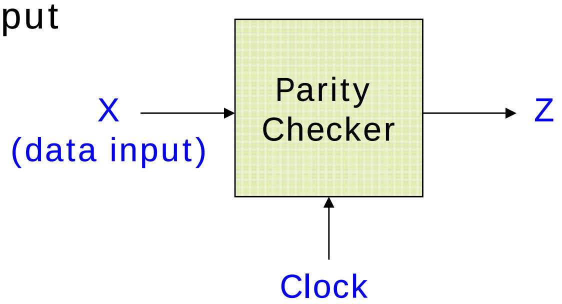
Outline

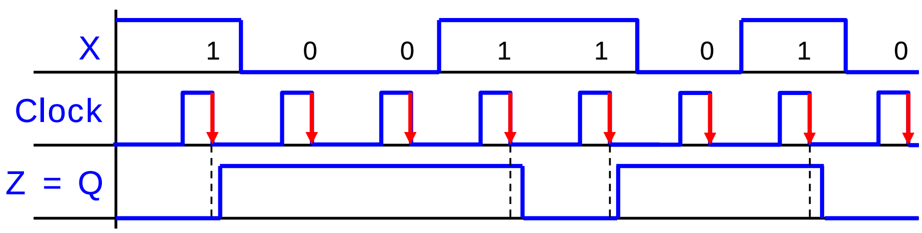
- Design of a sequence detector
- More complex design problems
- Guidelines for construction of state graphs
- Serial data code conversion
- Alphanumeric state graph notation
- Conversion between Mealy and Moore State Graphs

Design of a Sequence Detector
Sequential Parity Checker (recap)

- A parity checker for serial data
  - Z = 1 ⇔ the total number of 1 inputs received is odd (i.e., input parity is odd)
  - Z = 0 initially

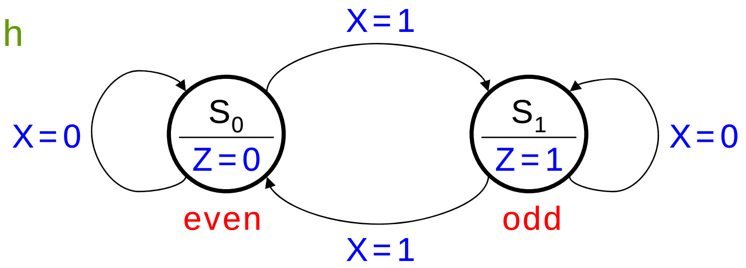
Block diagram





Design of a Sequence Detector
Sequential Parity Checker (recap)

State graph



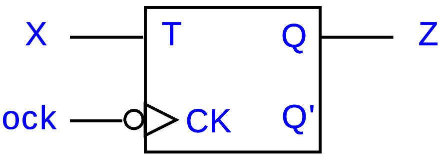
State table

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

state encoding/assignment

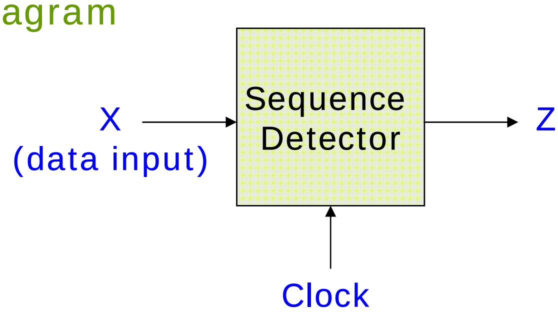
Q	Q^+ X=0	Q^+ X=1	T X=0	T X=1	Z
0	0	1	0	1	0
1	1	0	0	1	1

Logic circuit



Design of a Sequence Detector
{101}-Sequence Detector

Block diagram



Z=1 ⇔ input sequence "101" is received

Input/output sequence example

X = 0 0 1 1 0 1 1 0 0 1 0 1 0 1 0 0

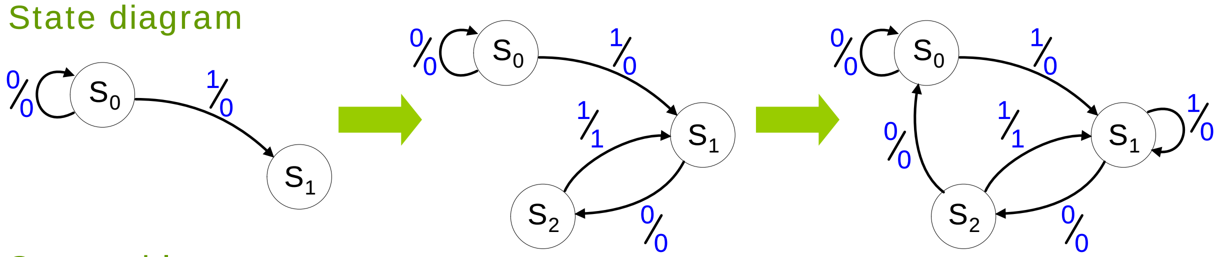
Z = 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0

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

Design of a Sequence Detector {101}-Sequence Detector

- Mealy machine

State diagram



State table

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

AB	A^+B^+ X=0	A^+B^+ X=1	Z X=0	Z X=1
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1
11	-	-	-	-

S_0: initial state
S_1: sequence ending with 1 received
S_2: sequence ending with 10 received

Design of a Sequence Detector {101}-Sequence Detector

- Mealy machine

Next-state maps

AB \ X	0	1
00	0	0
01	1	0
11	x	x
10	0	0

$A^+ = X'B$

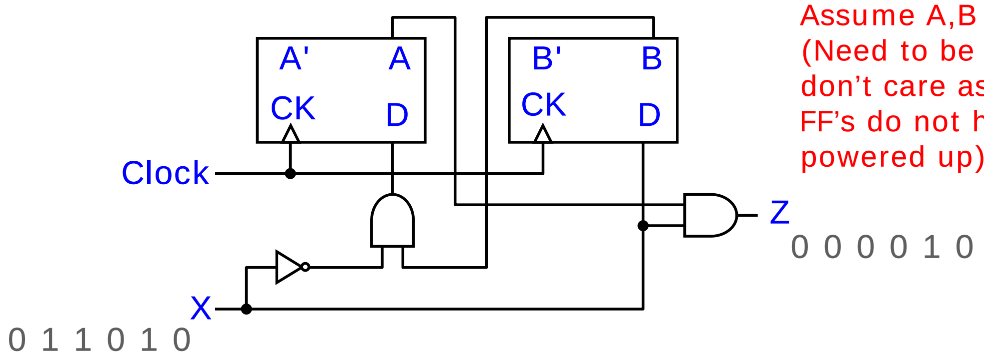
AB \ X	0	1
00	0	1
01	0	1
11	x	x
10	0	1

$B^+ = X$

AB \ X	0	1
00	0	0
01	0	0
11	x	x
10	0	1

$Z = XA$

Circuit realization

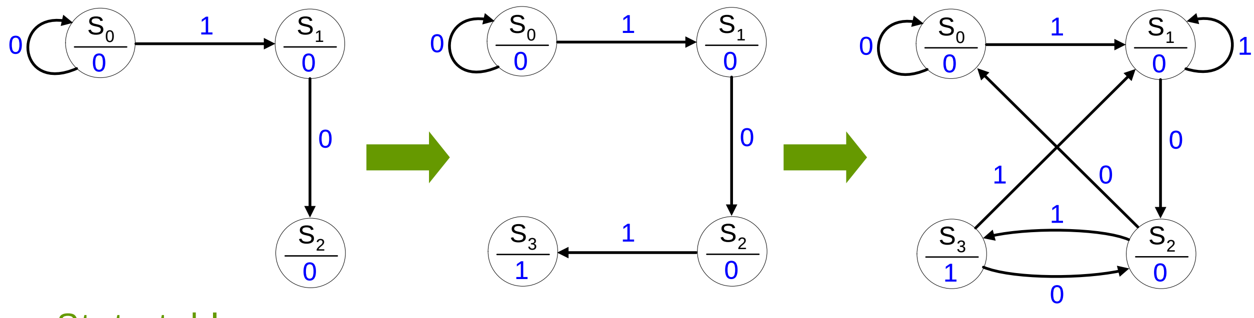


Assume A,B can be reset to 0 (Need to be careful about the don't care assignment if the FF's do not have reset when powered up)

Design of a Sequence Detector
{101}-Sequence Detector

Moore machine

State diagram



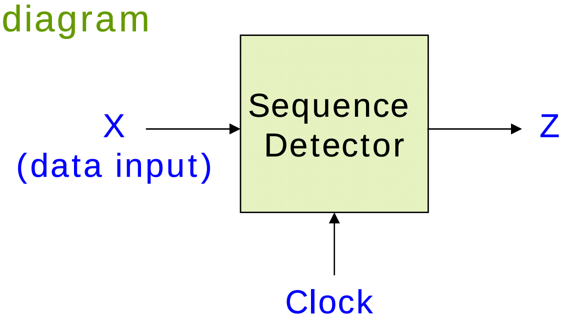
State table

Present State	Next State X=0	Next State X=1	Present Output Z
S_0	S_0	S_1	0
S_1	S_1	S_2	0
S_2	S_2	S_0	0
S_3	S_3	S_1	1

AB	A^+B^+ X=0	A^+B^+ X=1	Z
00	00	01	0
01	11	01	0
11	00	10	0
10	11	01	1

More Complex Design Problems
{010,1001}-Sequence Detector

Block diagram



Z=1 ⇔ input sequence "010" or "1001" is received

Input/output sequence example

X = 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 1 0

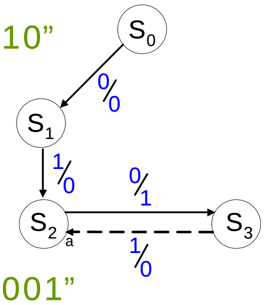
Z = 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0

Time: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

More Complex Design Problems
{010,1001}-Sequence Detector

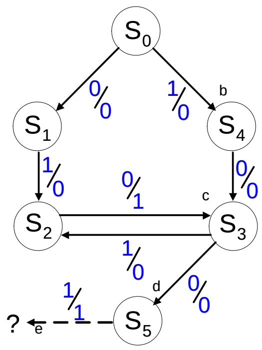
Mealy machine implementation

(1) Partial graph for “010”



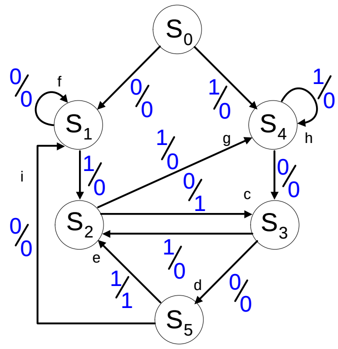
State	Sequence received
S_0	reset
S_1	0
S_2	01
S_3	010

(2) Partial graph for “1001”



State	Sequence ends in
S_0	reset
S_1	0 (but not 10)
S_2	01
S_3	10
S_4	1 (but not 01)
S_5	100

(3) Complete state graph



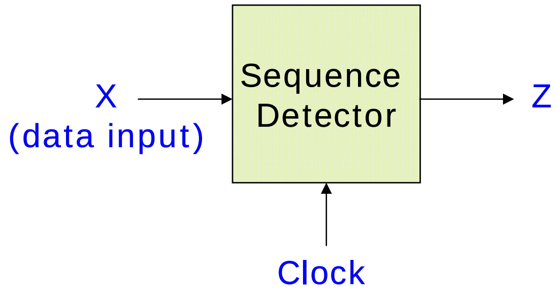
More Complex Design Problems
{010,1001}-Sequence Detector

Exercise

- Moore machine implementation

More Complex Design Problems
Modified Parity Sequence Detector

Block diagram



Z=1 ⇔ the total number of 1's received is odd and at least two consecutive 0's have been received

Input/output sequence example

X = 1 0 1 1 0 0 1 1

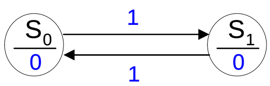
odd odd odd odd odd odd

Z = (0) 0 0 0 0 0 1 0 1

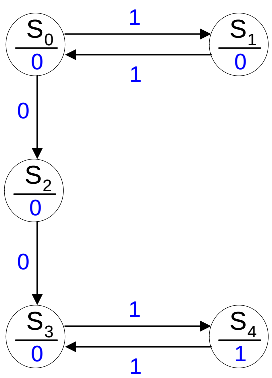
More Complex Design Problems
Modified Parity Sequence Detector

Moore machine implementation

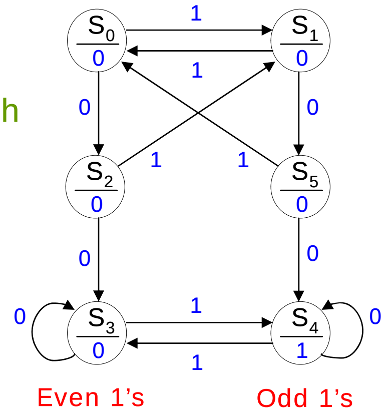
(1) Partial graph



(2) Partial graph



(3) Complete state graph



State	Sequence received
S_0	reset on even 1's
S_1	odd 1's
S_2	even 1's and ends in 0
S_3	even 1's and 00 occurred
S_4	odd 1's and 00 occurred
S_5	odd 1's and ends in 0

More Complex Design Problems
Modified Parity Sequence Detector

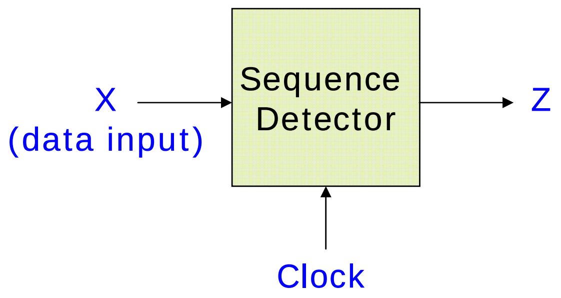
- Exercise
  - Mealy machine implementation

Construction of State Graphs

- Guidelines
  1. Construct sample input/output sequences
  2. Determine under what conditions, if any, the circuit should reset to its initial state
  3. If only one or two sequences lead to a nonzero output, construct a partial state graph for those sequences
  4. Alternatively, determine what sequences or groups of sequences must be remembered by the circuit and set up states accordingly
  5. Each time an arrow is added, determine whether it can go to one of the previously defined states or whether a new state must be added
  6. Check there is only one outgoing edge leaving each state for each input value
  7. Test the completed graph and make sure correct

Construction of State Graphs
Example 1

Block diagram



Z=1 ⇔ input sequence 0101 or 1001 occurs

The circuit examines groups of 4 consecutive inputs, and resets after every 4 inputs

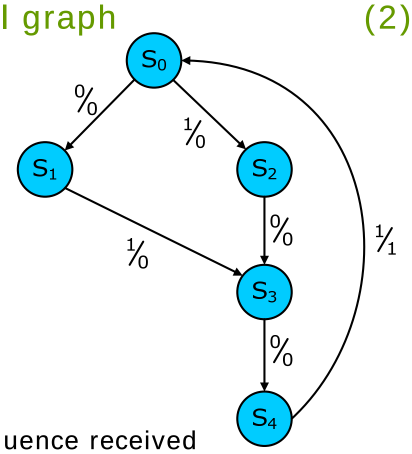
Input/output sequence example

X =	0101	0010	1001	0100
Z =	0001	0000	0001	0000

Construction of State Graphs
Example 1

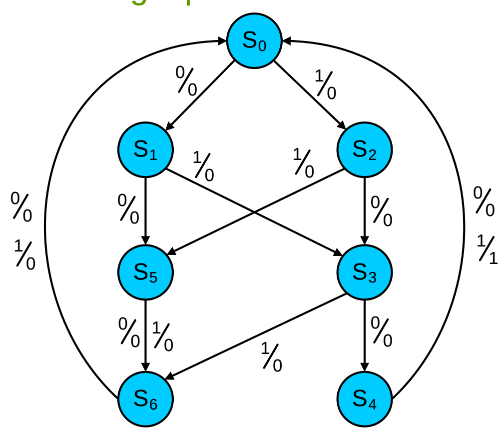
Mealy machine implementation

(1) Partial graph



State	Sequence received
S_0	reset
S_1	0
S_2	1
S_3	01 or 10
S_4	010 or 100

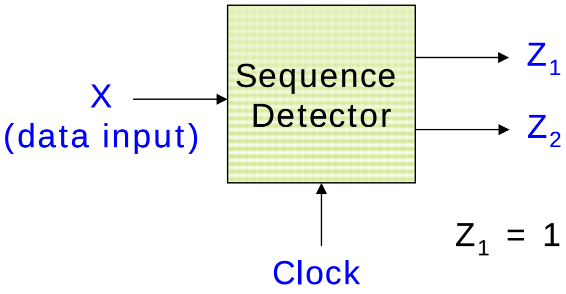
(2) Complete state graph



State	Sequence received
S_5	two inputs received, no 1 output is possible
S_6	three inputs received, no 1 output is possible

Construction of State Graphs
Example 2 (omitted)

Block diagram



Z_1 = 1: every time sequence 100 is completed and 010 has never occurred
Z_2 = 1: every time sequence 010 is completed

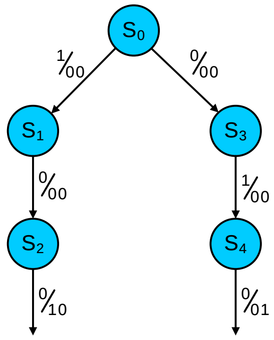
Input/output sequence example

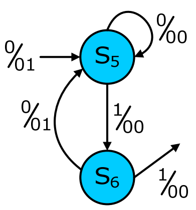
X = 1 0 0 1 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 1 0 0
Z_1 = 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Z_2 = 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 1 0 0 0 0 1 0

Construction of State Graphs
Example 2 (omitted)

Mealy machine implementation

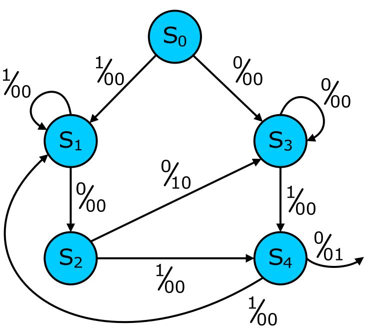
(1) Partial graph

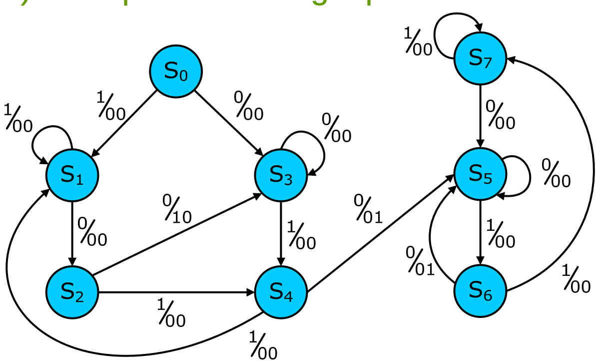




partial graph for 010

(2) Complete state graph





State	Description		
S_0	No progress on 100	No progress on 010	010 has never occurred
S_1	Progress of 1 on 100	No progress on 010	
S_2	Progress of 10 on 100	Progress of 0 on 010	
S_3	No progress on 100	Progress of 0 on 010	
S_4	Progress of 1 on 100	Progress of 01 on 010	
S_5		Progress of 0 on 010	010 has occurred
S_6		Progress of 01 on 010	
S_7		No progress on 010	

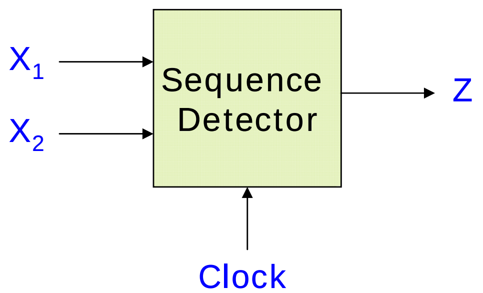
Construction of State Graphs Example 2 (omitted)

State table

Present State	Next state		Output Z_1Z_2	
	X=0	X=1	X=0	X=1
S_0	S_3	S_1	00	00
S_1	S_2	S_1	00	00
S_2	S_3	S_4	10	00
S_3	S_3	S_4	00	00
S_4	S_5	S_1	01	00
S_5	S_5	S_6	00	00
S_6	S_5	S_7	01	00
S_7	S_5	S_7	00	00

Construction of State Graphs Example 3 (omitted)

Block diagram



Z remains a constant value unless one of the following input sequences occurs

(a) Input sequence X_1X_2 = 01, 11 causes Z=0

(b) Input sequence X_1X_2 = 10, 11 causes Z=1

(c) Input sequence X_1X_2 = 10, 01 causes Z to change value

(X_1X_2 = 01, 11 means X_1 = 0, X_2 = 1 followed by X_1 = 1, X_2 = 1)

Construction of State Graphs Example 3 (omitted)

- Moore machine implementation
  - Observation:
    - Only the previous and present inputs (input sequence of length 2) will determine the output
    - Unnecessary to use a separate state for 00 and 11 because neither input starts a sequence which leads to an output change

State designation

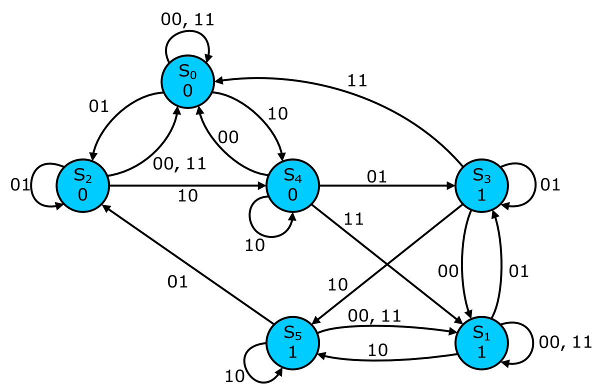
Previous Input (X_1X_2)	Output (Z)	State Designation
00 or 11	0	S_0
00 or 11	1	S_1
01	0	S_2
01	1	S_3
10	0	S_4
10	1	S_5

Construction of State Graphs Example 3 (omitted)

State table

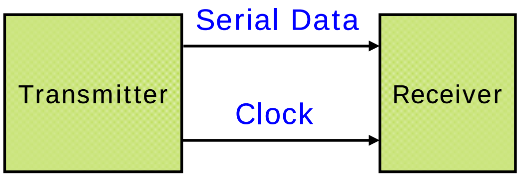
Present State	Z	Next State X_1X_2 = 00	01	11	10
S_0	0	S_0	S_2	S_0	S_4
S_1	1	S_1	S_3	S_1	S_5
S_2	0	S_0	S_2	S_0	S_4
S_3	1	S_1	S_3	S_0	S_5
S_4	0	S_0	S_3	S_1	S_4
S_5	1	S_1	S_2	S_1	S_5

State graph

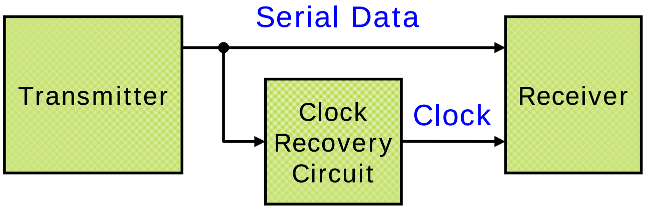


Serial Data Code Conversion

- Transmission of serial bit streams
  - Two common approaches
    1. Clock signal transmitted along with the data



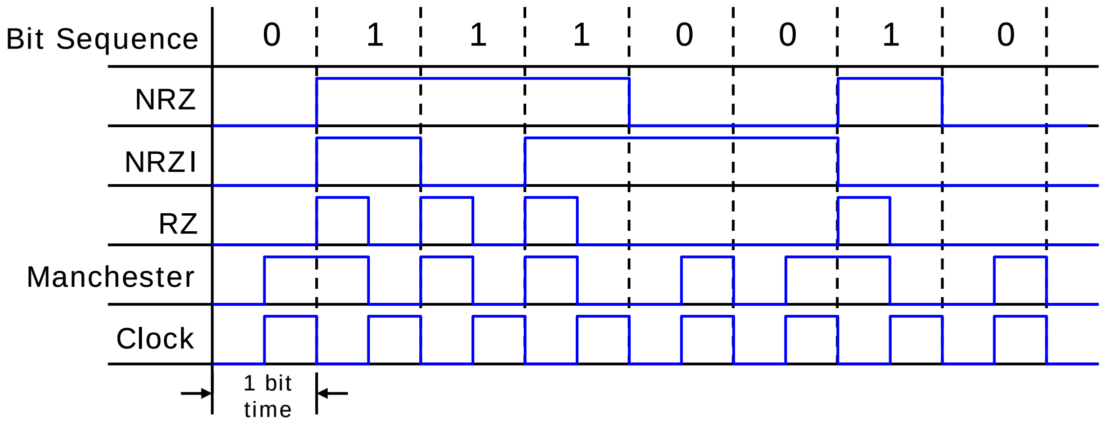
    2. Clock recovery circuit used



Serial Data Code Conversion

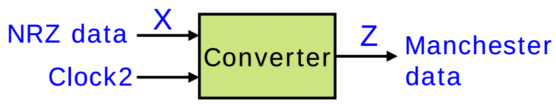
- Four typical coding schemes
  - NRZ (non-return-to-zero) code
  - NRZI (non-return-to-zero-inverted) code
  - RZ (return-to-zero) code
  - Manchester code
    - Easy to recover the clock signal

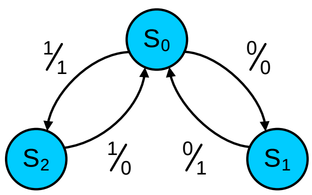
Example

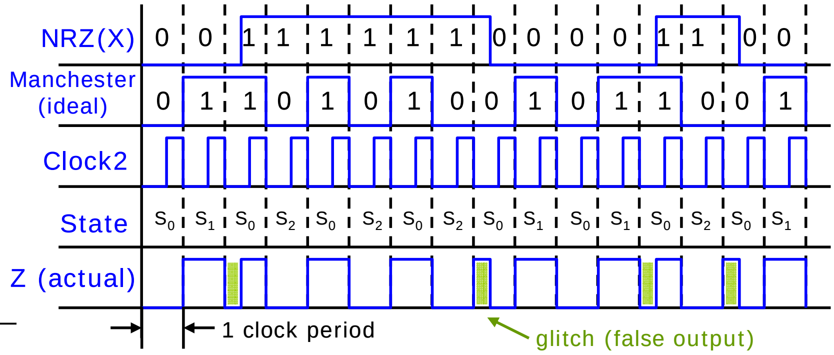


Serial Data Code Conversion
NRZ-Code to Manchester-Code

- Mealy machine implementation
  - Use Clock2, twice the frequency of the basic clock
    - If the NRZ bit is 0 (1), it will be 0 (1) for two Clock2 periods



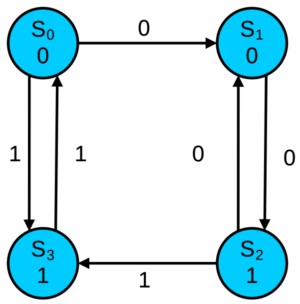


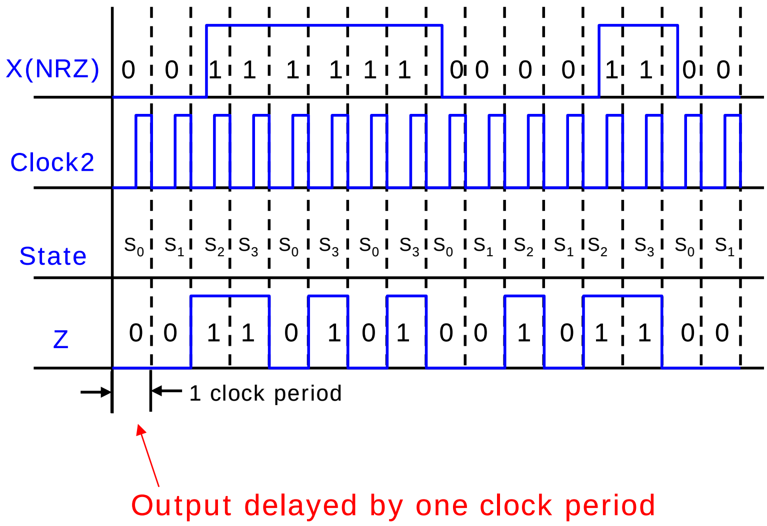


Present State	Next State X=0	Next State X=1	Output Z X=0	Output Z X=1
S_0	S_1	S_2	0	1
S_1	S_0	-	1	-
S_2	-	S_0	-	0

Serial Data Code Conversion
NRZ-Code to Manchester-Code

- Moore machine implementation



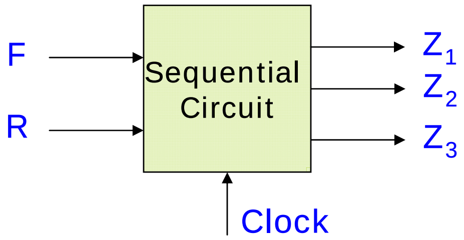


Present State	Next State X=0	Next State X=1	Present Output Z
S_0	S_1	S_3	0
S_1	S_2	-	0
S_2	S_1	S_3	1
S_3	-	S_0	1

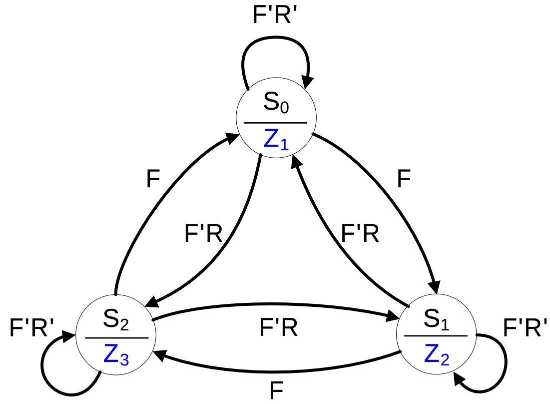
Alphanumeric State Graph Notation

- State graphs with variable names on arc labels (and in states for Moore machine)

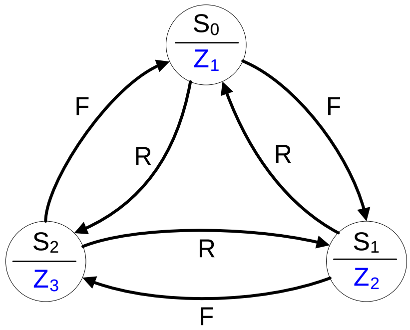
Example



Completely specified state graph

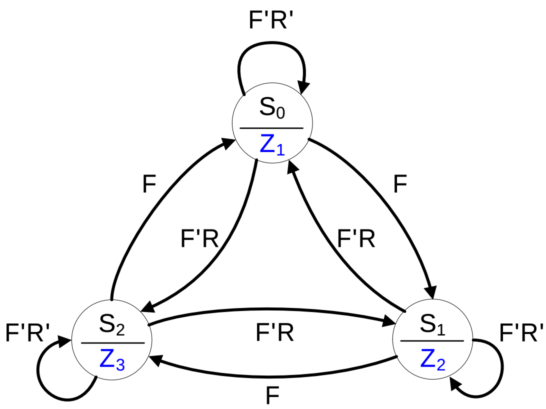


Incompletely specified state graph



Alphanumeric State Graph Notation

State graph



State table

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

Check input signals (for every state):

$F + F'R + F'R' = F + F' = 1$

$\Rightarrow$ Transition defined for every input combination

$F \cdot F'R = 0,\ F \cdot F'R' = 0,\ F'R \cdot F'R' = 0$

$\Rightarrow$ At most one next state for every input combination

Alphanumeric State Graph Notation

- A completely specified state graph has the following properties
  1. ORing together all input labels on arcs outgoing from a state reduces to 1 (i.e., complete transition)
     - For every input combination, at least one next state is defined
  2. ANDing together any pair of input labels on arcs outgoing from a state reduces to 0 (i.e., deterministic transition)
     - For every input combination, no more than one next state is defined

  - If both properties are true, then exactly one next state is defined

Alphanumeric State Graph Notation

- Convention for Mealy machine
  - The label X_iX_j/Z_pZ_q on an arc means if X_i and X_j are 1 (we don't care what the other input values are), the outputs Z_p and Z_q are 1 (and the other outputs are 0)

E.g., for a circuit with 4 inputs (X_1, X_2, X_3, X_4) and 4 outputs (Z_1, Z_2, Z_3, Z_4)

X_1X_4'/Z_2Z_3 is equivalent to 1--0/0110

Conversion between Mealy and Moore State Graphs

- Convert Mealy to Moore
  1. Push the output label on an edge to its next state (so delay introduced!)
  2. If a state receives different output labels, duplicate the state such that every copy has exactly one output label
  3. Connect every edge properly to the state with correct output label
- Convert Moore to Mealy
  1. Distribute the output label of a state to its incoming edges
  2. Simplify the state graph by merging equivalent states

Mealy-type implementation of a circuit can have fewer states than Moore-type implementation

Conversion between Mealy and Moore State Graphs

- Exercise