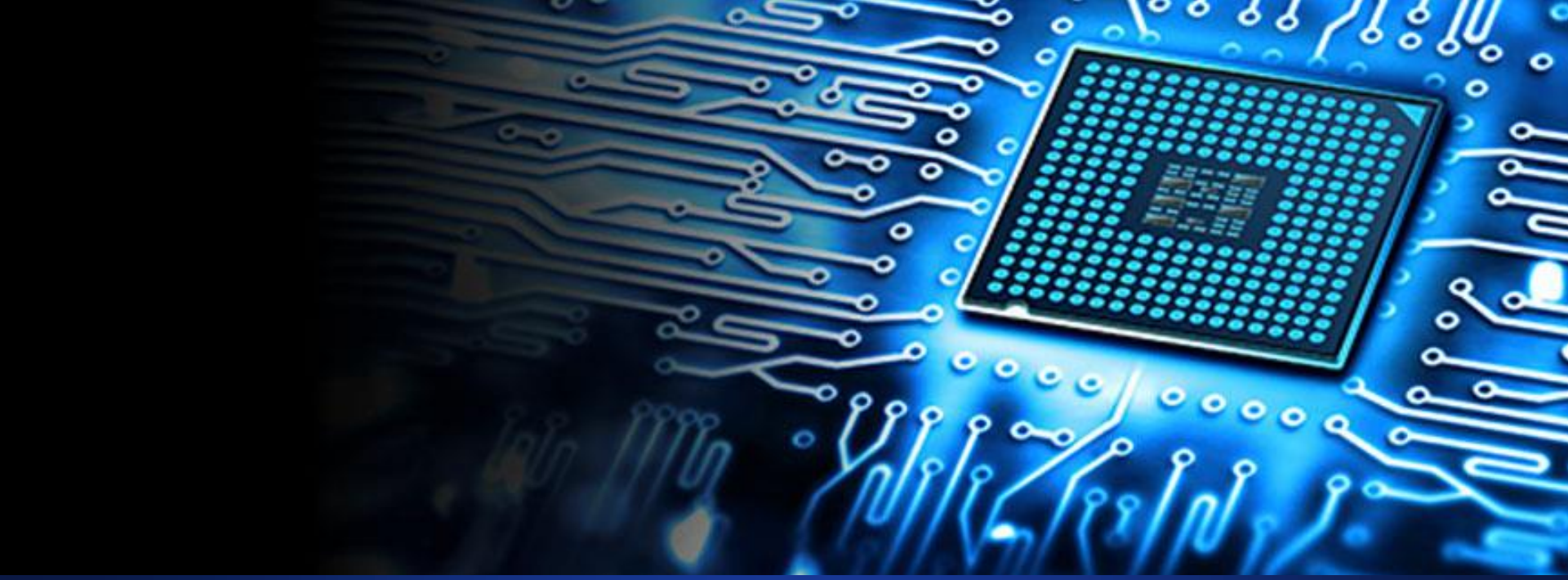


Chapter 1.6-11
Review of Logic Design Fundamentals

Version: 2023/11/15

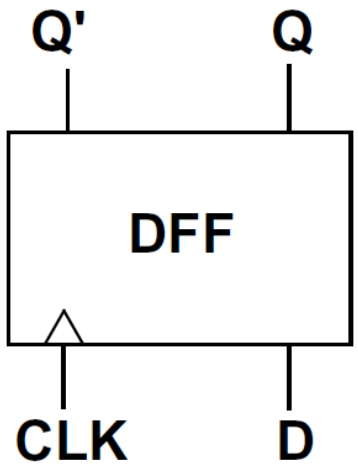
	Contents
1.1	Combinational Logic（组合逻辑）
1.2	Boolean Algebra and Algebra Simplification
1.3	Karnaugh Maps
1.4	Designing with NAND and NOR Gates
1.5	Hazards in Combinational Circuits
1.6	**Flip-Flops and Latches**
1.7	Mealy Sequential Circuit Design
1.8	Moore Sequential Circuit Design
1.9	Equivalent States and Reduction of State Tables
1.10	Sequential Circuit Timing
1.11	Tristate Logic and Busses

1.6 Flip-Flops and Latches
(触发器和锁存器)

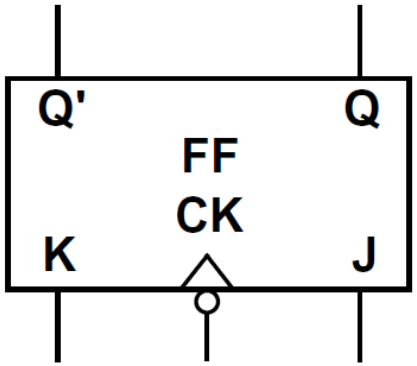
- **Latches** and **flip-flops** are the basic elements for storing information
- One latch or flip-flop can store one bit of information

	Difference
Latch (锁存器)	The outputs are constantly affected by their inputs as long as the enable signal is asserted, i.e., when they are enabled, their content changes immediately when their inputs change
Flip-flop (触发器)	Their content change only either at the rising or failing edge of the enable signal. This enable signal is usually the controlling clock signal. After the rising or falling edge of the clock, the flip-flop content remains constant even if the input changes

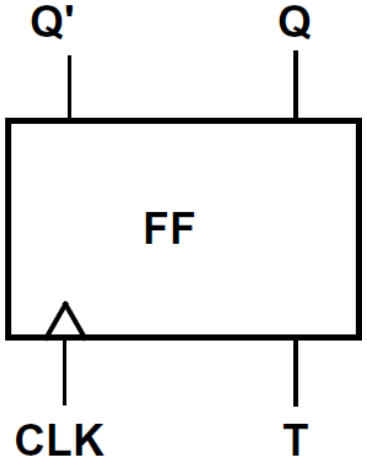
1.6 Flip-Flops and Latches
(触发器和锁存器)



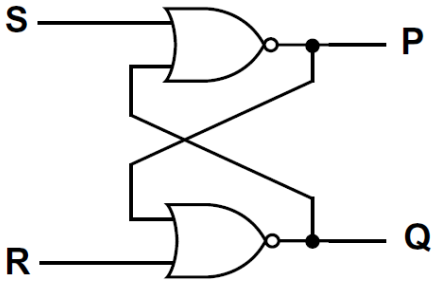
Clocked D flip-flop



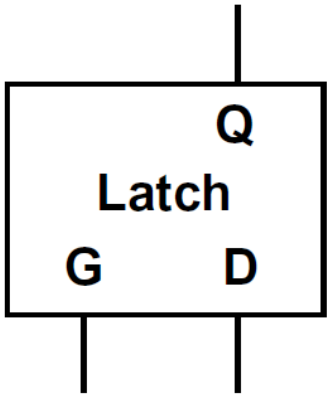
Clocked J-K flip-flop



Clocked T flip-flop

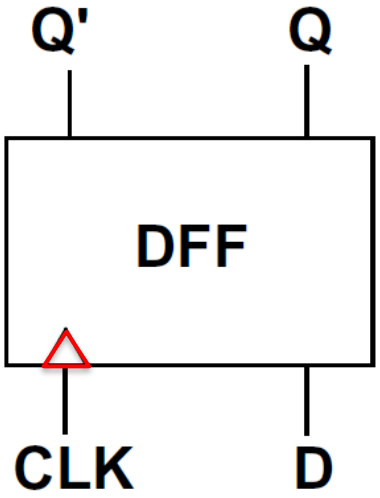


SR latch



Gated D latch

Clocked D flip-flop (时钟D触发器)



D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

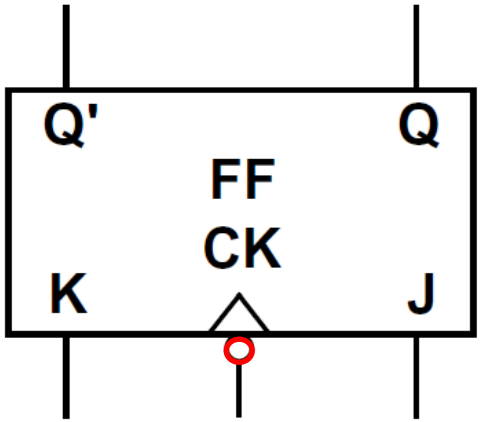
Q+: the next state of Q after the active edge of the clock

$Q^+ = D$

The next state of Q is equal to the D input before the rising edge

This D flip-flop changes state in response to the rising edge of the clock input

Clocked J-K flip-flop

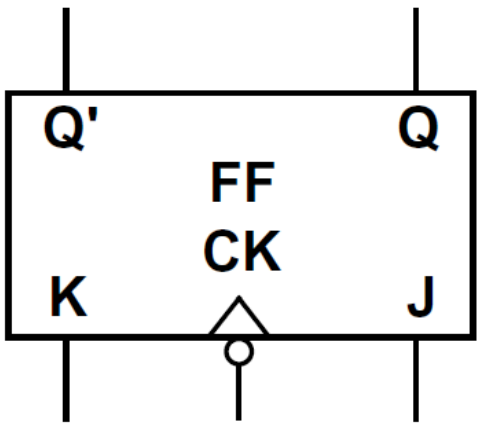


J	K	Q+
0	0	Q
0	1	0
1	0	1
1	1	Q'

All state changes occur following the falling edge of the clock input

下降沿触发

Clocked J-K flip-flop



J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

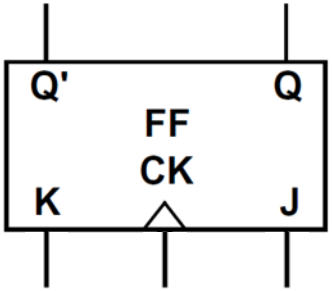
\JK Q\	00	01	11	10
0	0	0	1	1
1	1	0	0	1

JQ'

K'Q

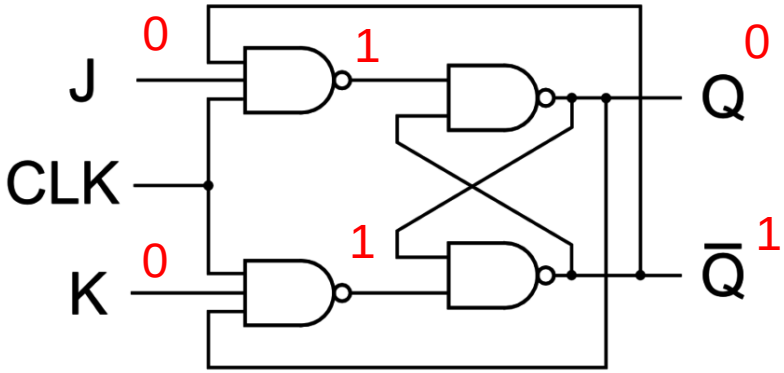
$$Q^+ = JQ' + K'Q$$

Clocked J-K flip-flop

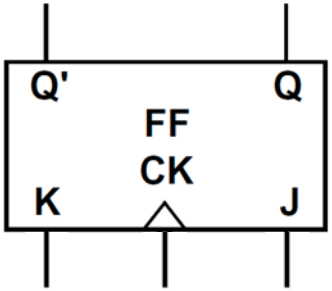


J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$$Q^+ = JQ' + K'Q$$

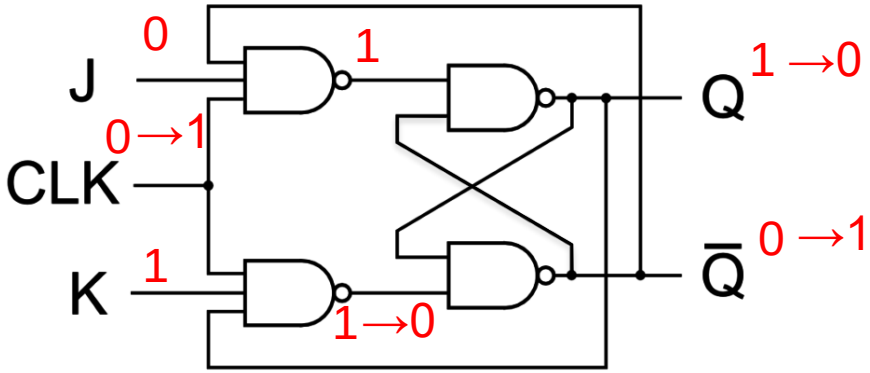


Clocked J-K flip-flop



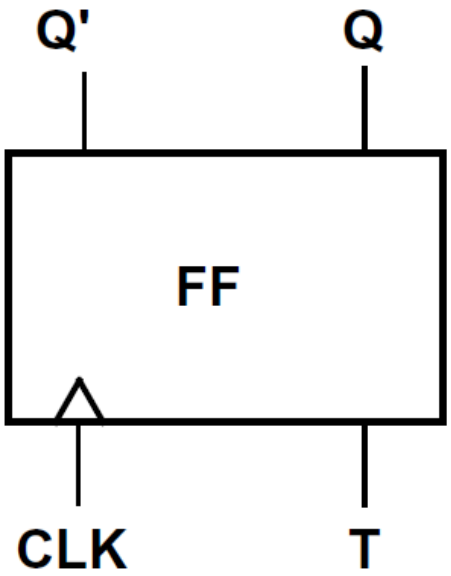
J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$$Q^+ = JQ' + K'Q$$



Clocked T flip-flop

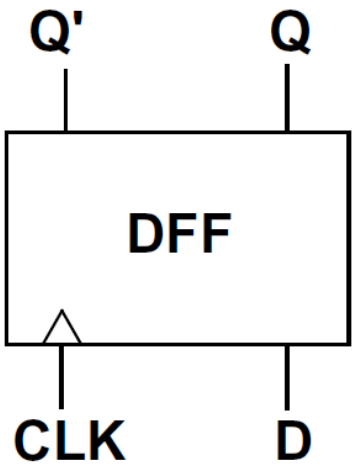
T	Q+
0	Q
1	Q'



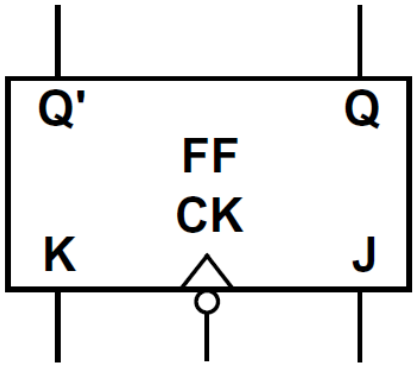
T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

$$Q^+ = QT' + Q'T = Q \oplus T$$

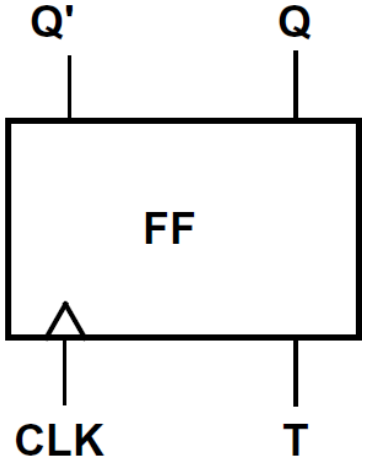
1.6 Flip-Flops and Latches
(触发器和锁存器)



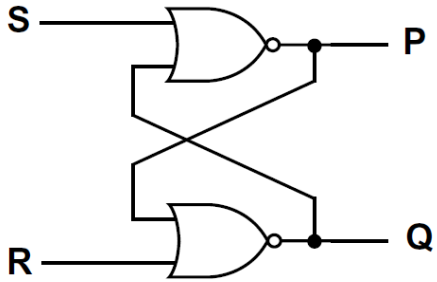
Clocked D flip-flop



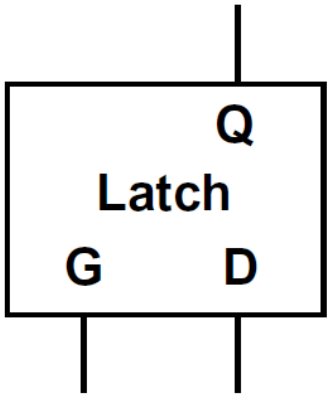
Clocked J-K flip-flop



Clocked T flip-flop

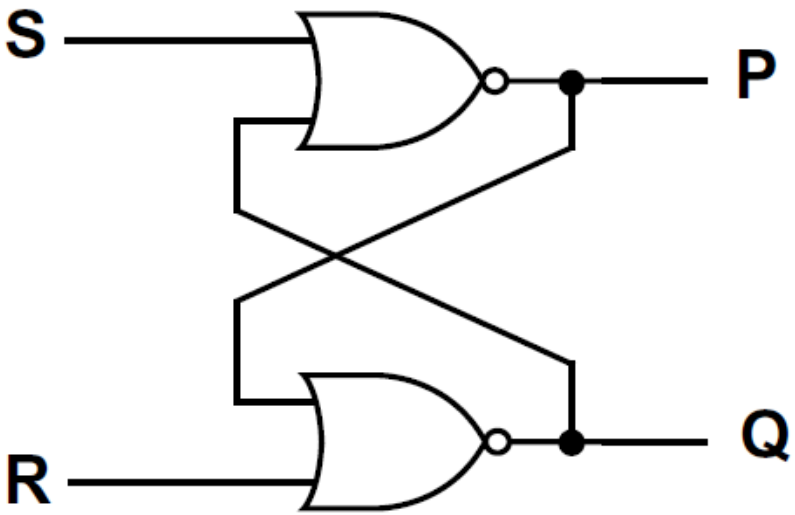


SR latch



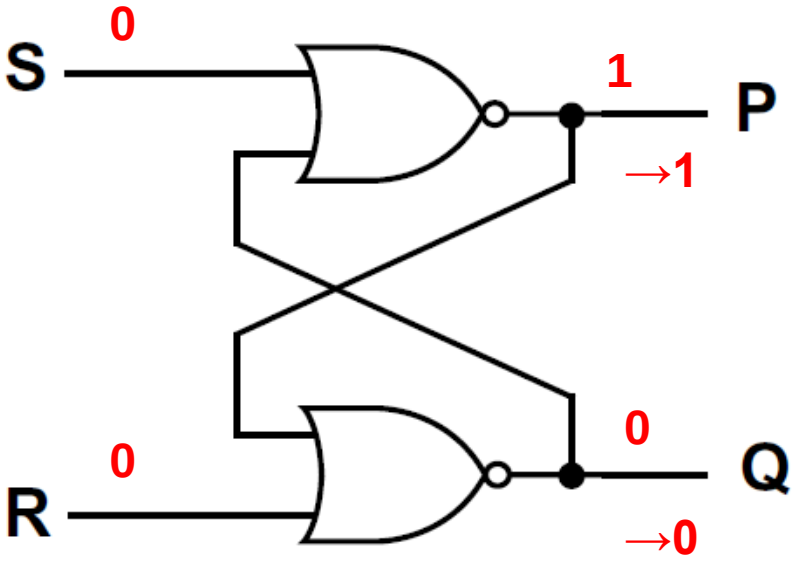
Gated D latch

S-R (set-reset) latch (置位-复位锁存器)



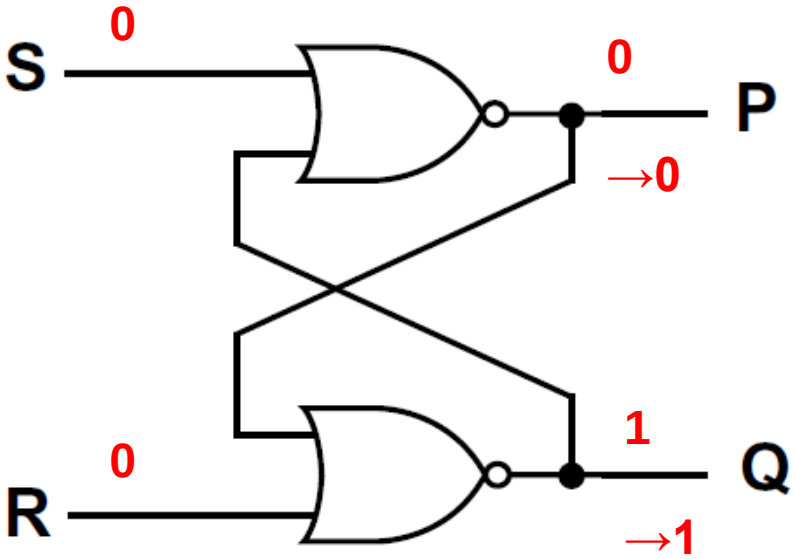
S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

S-R (set-reset) latch (置位-复位锁存器)



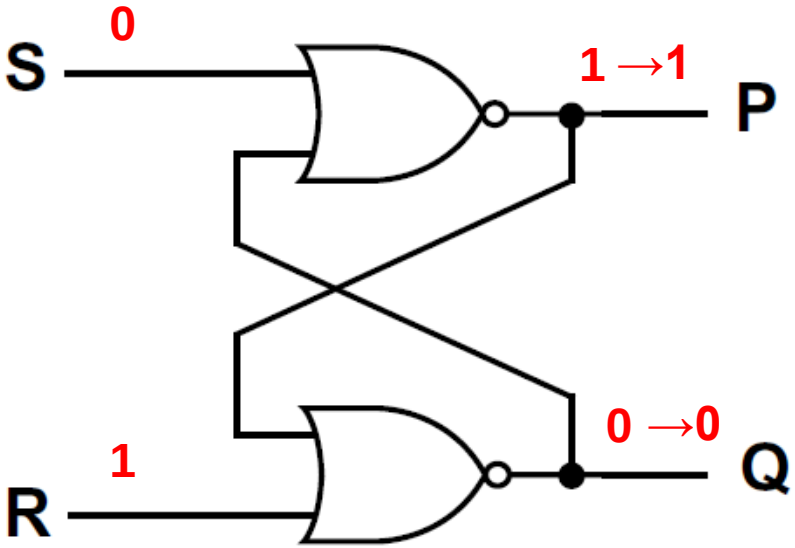
S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

S-R (set-reset) latch (置位-复位锁存器)



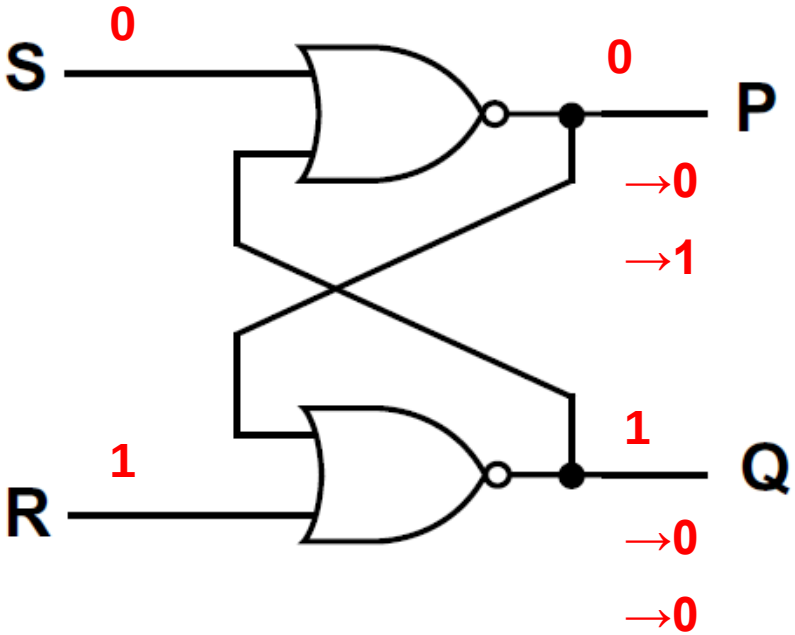
S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

S-R (set-reset) latch (置位-复位锁存器)



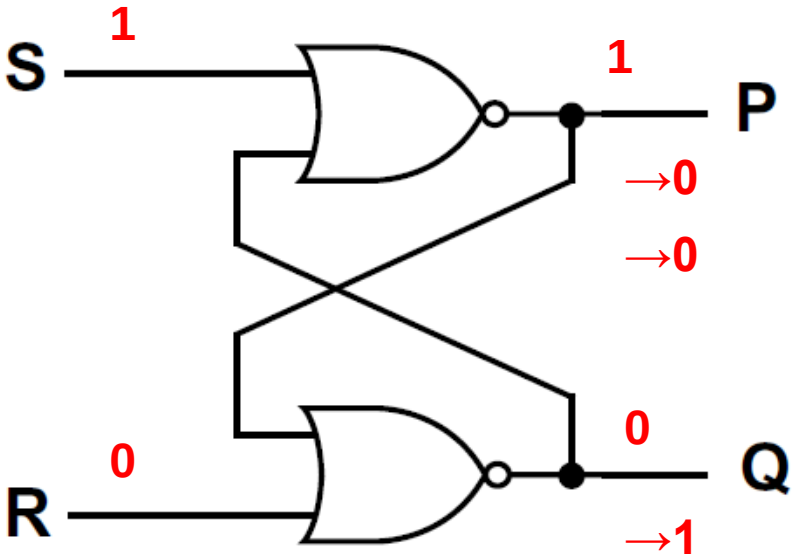
S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

S-R (set-reset) latch (置位-复位锁存器)



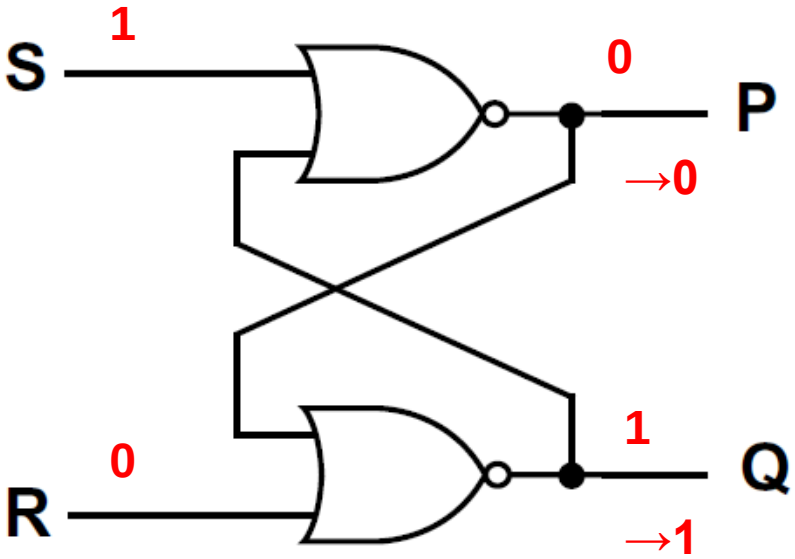
S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

S-R (set-reset) latch (置位-复位锁存器)



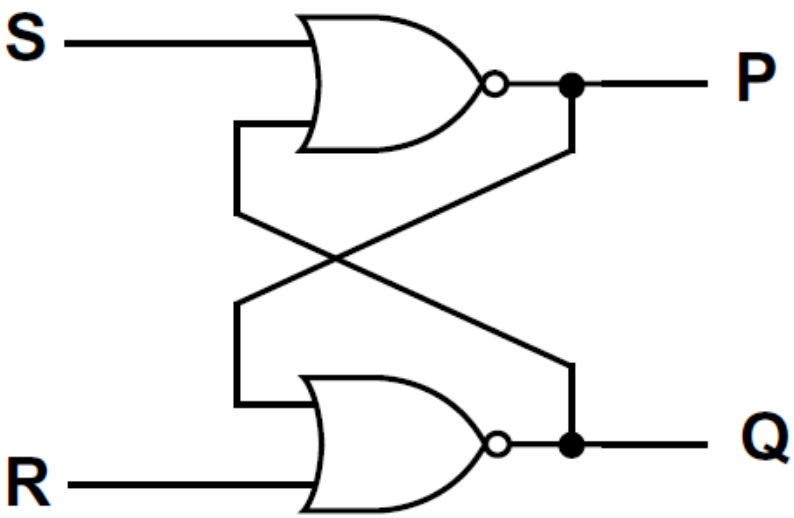
S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

S-R (set-reset) latch (置位-复位锁存器)



S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

S-R (set-reset) latch (置位-复位锁存器)



S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

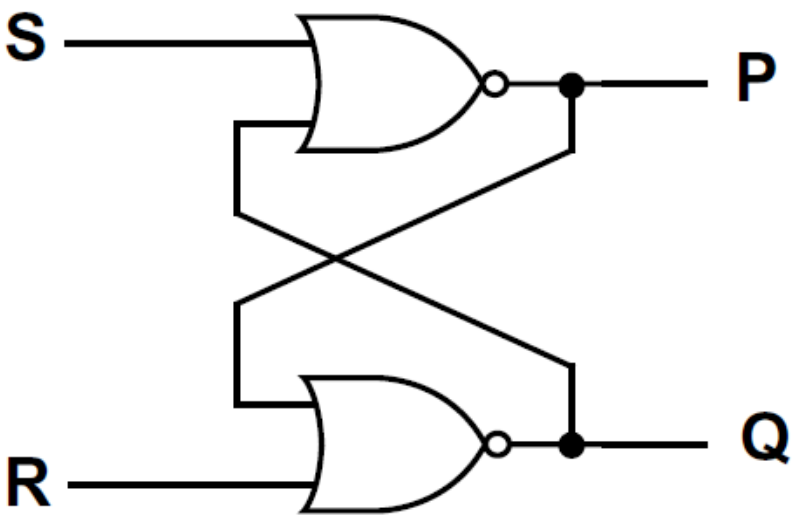
For S-R latch, S and R are not allowed to be 1 at the same time

\SR Q\	00	01	11	10
0	0	0	X	1
1	1	0	X	1

S

R'Q

S-R (set-reset) latch (置位-复位锁存器)



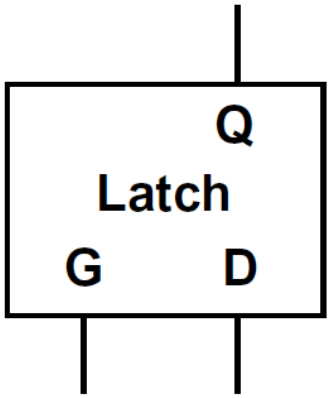
S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	–
1	1	1	–

$$Q^+ = S + R'Q$$

Q+ : the state after any input changes have propagated to Q

Gated D latch (门控D锁存器)

G	Q+
1	D
0	Q



G	D	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

\GD Q\	00	01	11	10
0	0	0	1	0
1	1	1	1	0

G'Q

GD

Q+ = GD + G'Q

Gated D latch (门控D锁存器)

\GD Q\	00	01	11	10
0	0	0	1	0
1	1	1	1	0

GD

G'Q

1-hazard

Q+ = GD + G'Q

\GD Q\	00	01	11	10
0	0	0	1	0
1	1	1	1	0

DQ

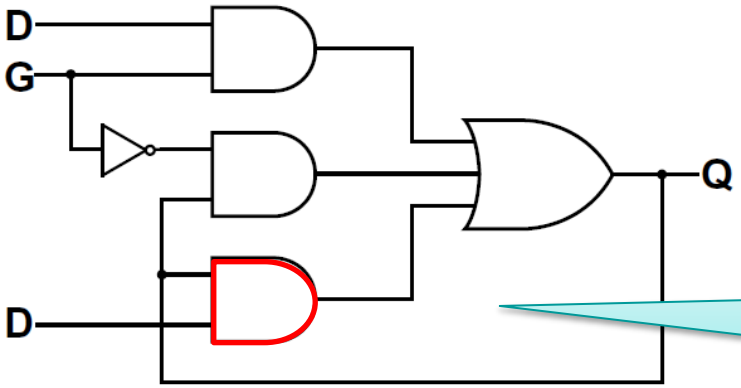
Hazard removed

Q+ = GD + G'Q + DQ

Gated D latch (门控D锁存器)

\GD Q\	00	01	11	10
0	0	0	1	0
1	1	1	1	0

$$Q+ = GD + G'Q + DQ$$



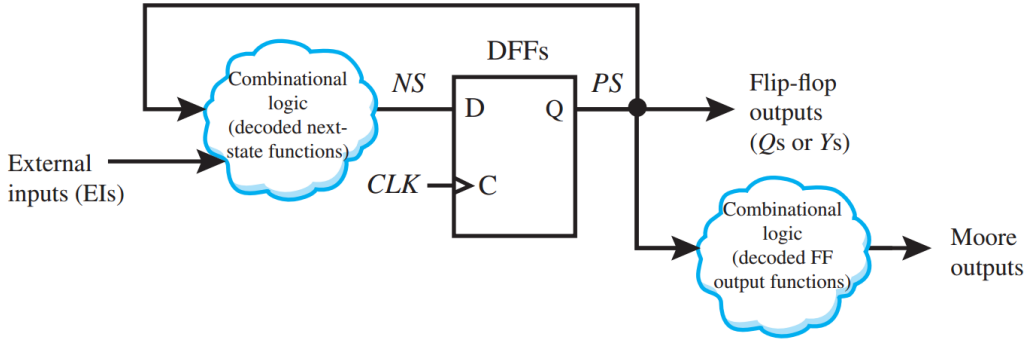
An extra AND gate is added to eliminate the 1-hazard

	Contents
1.1	Combinational Logic（组合逻辑）
1.2	Boolean Algebra and Algebra Simplification
1.3	Karnaugh Maps
1.4	Designing with NAND and NOR Gates
1.5	Hazards in Combinational Circuits
1.6	Flip-Flops and Latches
1.7	**Mealy Sequential Circuit Design**
1.8	Moore Sequential Circuit Design
1.9	Equivalent States and Reduction of State Tables
1.10	Sequential Circuit Timing
1.11	Tristate Logic and Busses

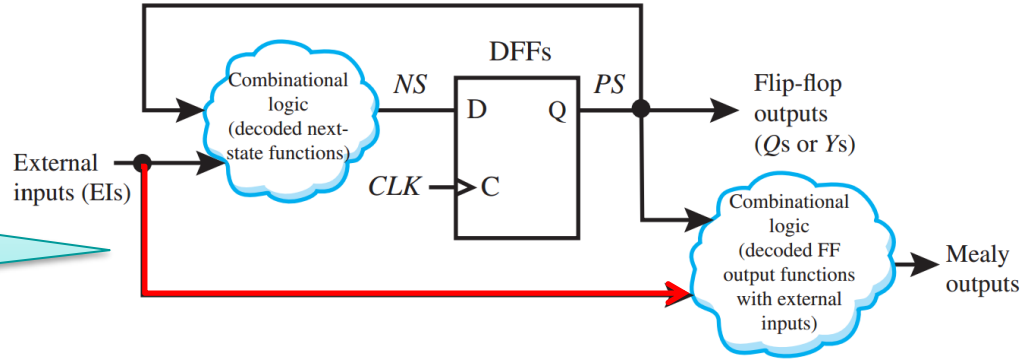
Sequential circuits

Circuits	The outputs depend on
Mealy circuit	➢ The present state ➢ The present inputs
Moore circuit	➢ Only the present **state**

Moore circuit can also have inputs

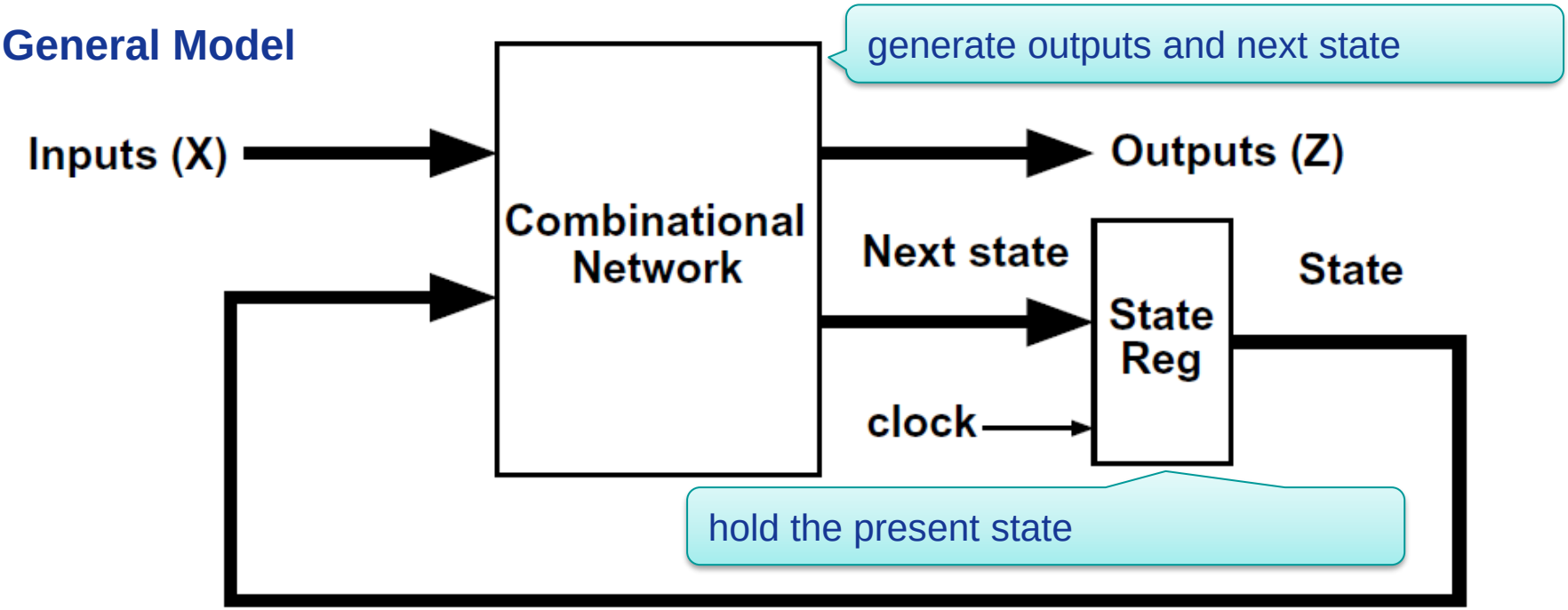


The present inputs affect the outputs of Mealy circuit



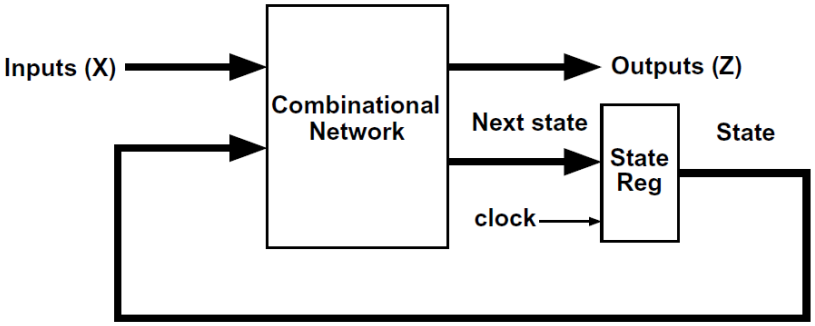
1.7 Mealy Sequential Circuit Design

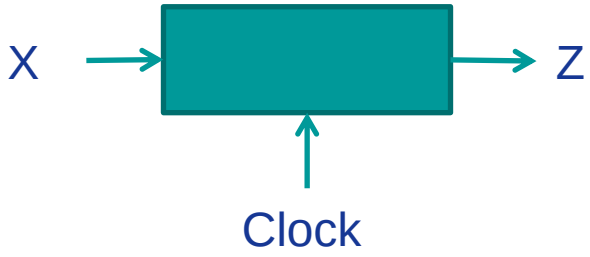
General Model



1. Inputs X change to a new value
2. After a delay, the Z outputs and next state appear at the output of the combinational circuit
3. The next state is clocked into state register and the state changes

1.7.1 Mealy Machine Design Example 1: Sequence Detector





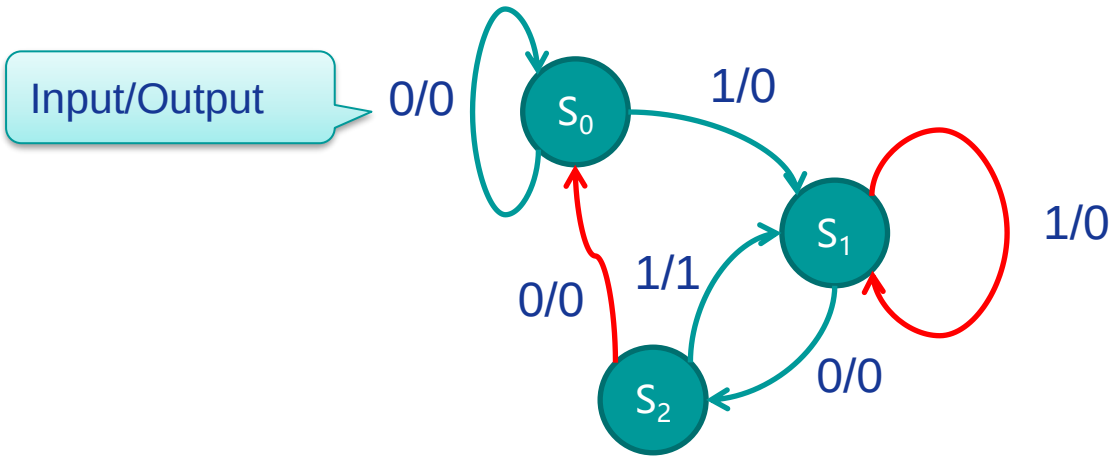
Input	**X = 0 0 1 1 0 1 1 0 0 1 0 1 0 1 0 0**
Output	Z = 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0

1.7.1 Mealy Machine Design Example 1: Sequence Detector

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 . . .

State graph



Present State	Next State		Present Output	
	X=0	X=1	X=0	X=1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S1	0	1

1.7.1 Mealy Machine Design Example 1: Sequence Detector

State Table for Sequence Detector

Present State	Next State		Present Output	
	X=0	X=1	X=0	X=1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S1	0	1

Two flip-flops **A** and **B** are used to represent all states (**encoded state assignment**)

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

1.7.1 Mealy Machine Design Example 1: Sequence Detector

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

K-Maps for Next States (A+B+) and Output (Z) of Sequence Detector

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

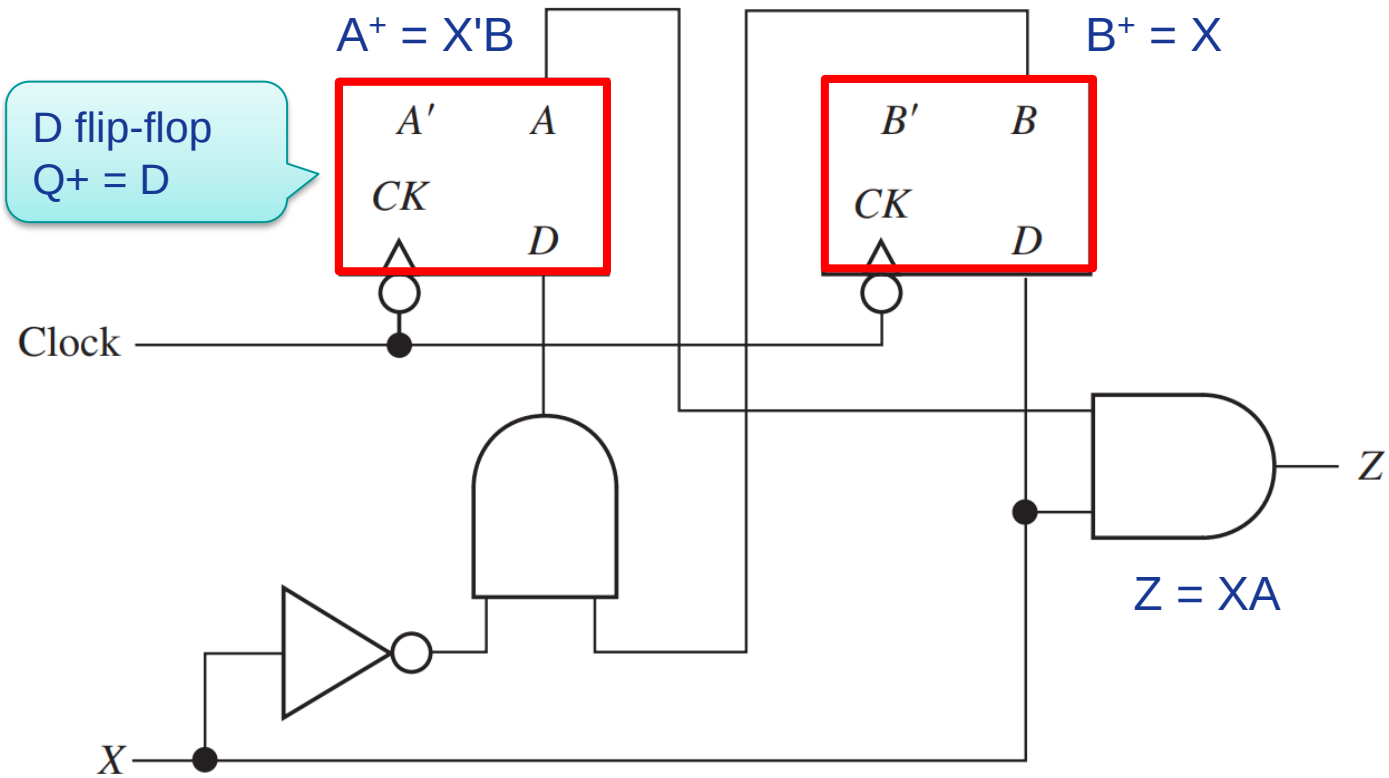
$B^+ = X$

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

$Z = XA$

1.7.1 Mealy Machine Design Example 1: Sequence Detector

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

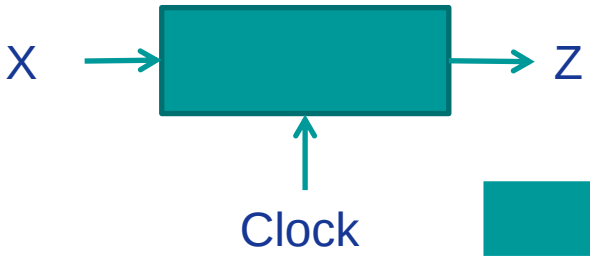


1.7.2 Mealy machine design example 2: BCD to excess-3 code converter

Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
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

+0011

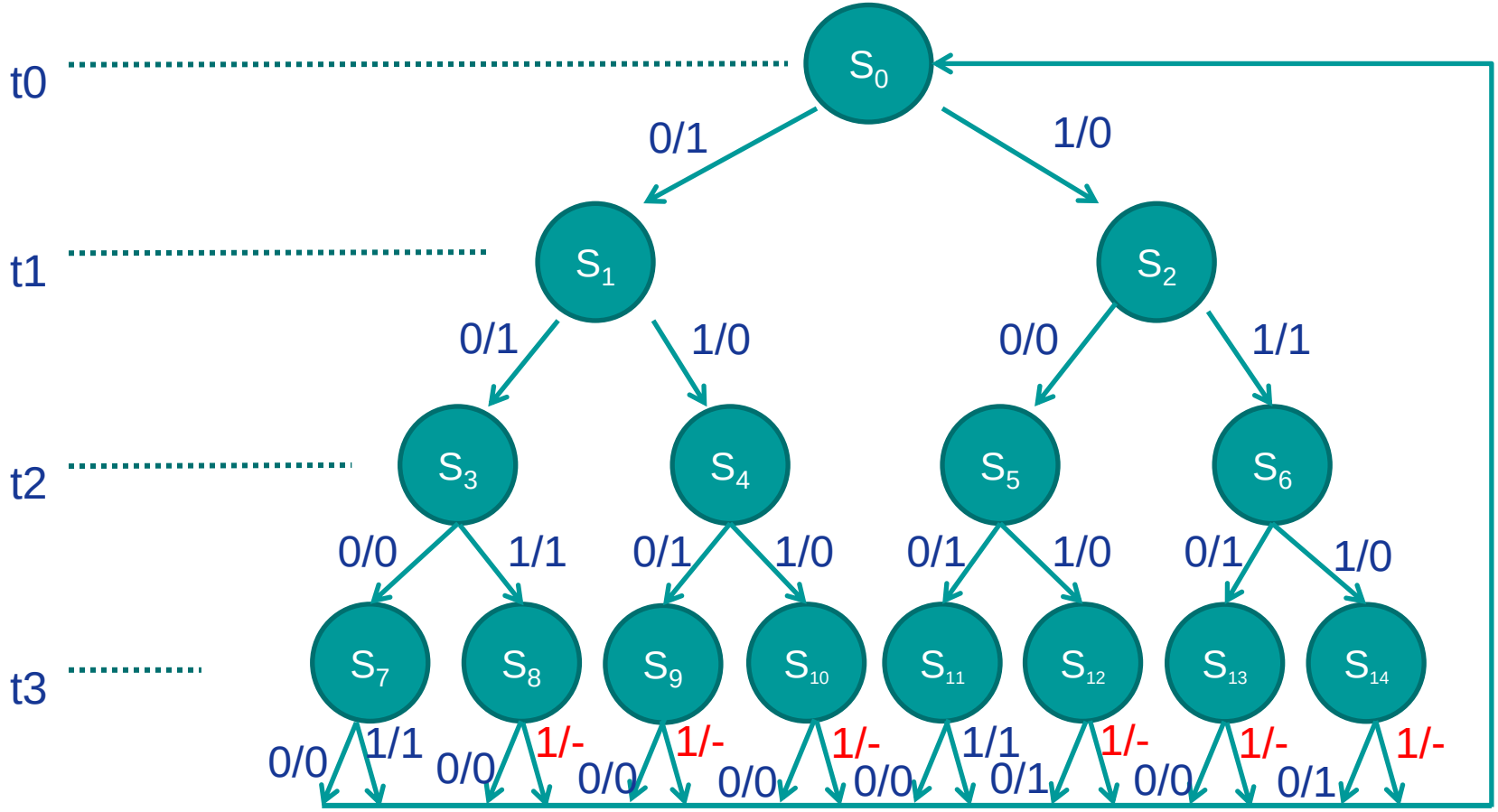
1.7.2 Mealy machine design example: BCD to excess-3 code converter



Least significant bit (LSB) arrives first

X Input (BCD)				Z Output (excess-3)			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

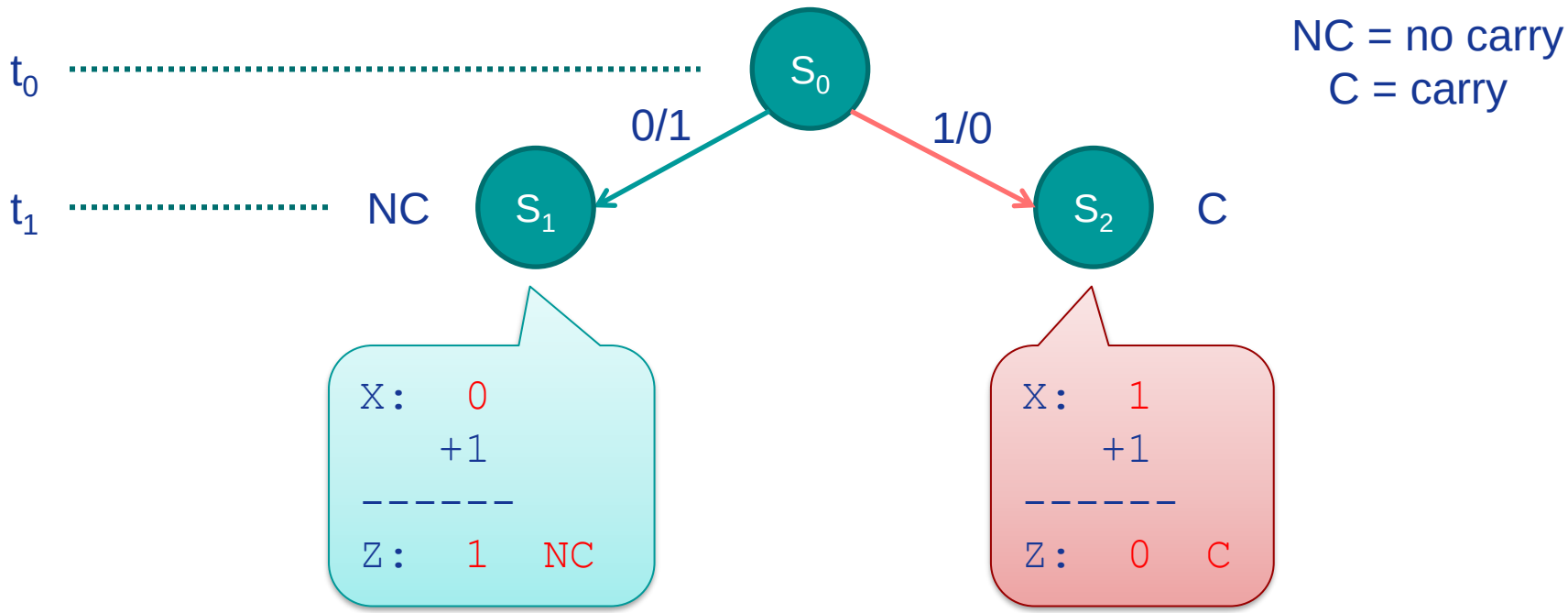
1.7.2 Mealy machine design example 2: BCD to excess-3 code converter



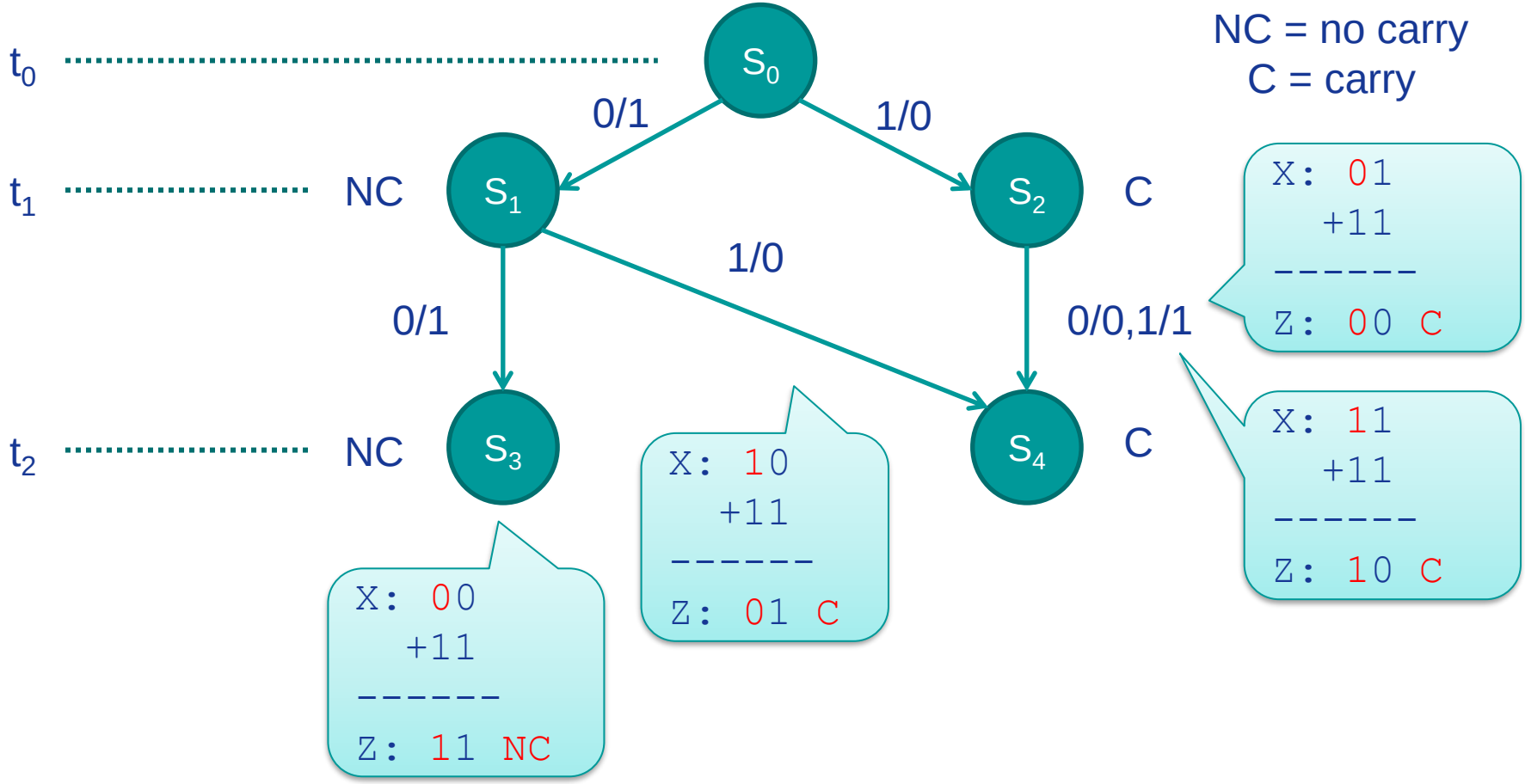
1+2+4+8=15 States!

1.7.2 Mealy machine design example 2: BCD to excess-3 code converter

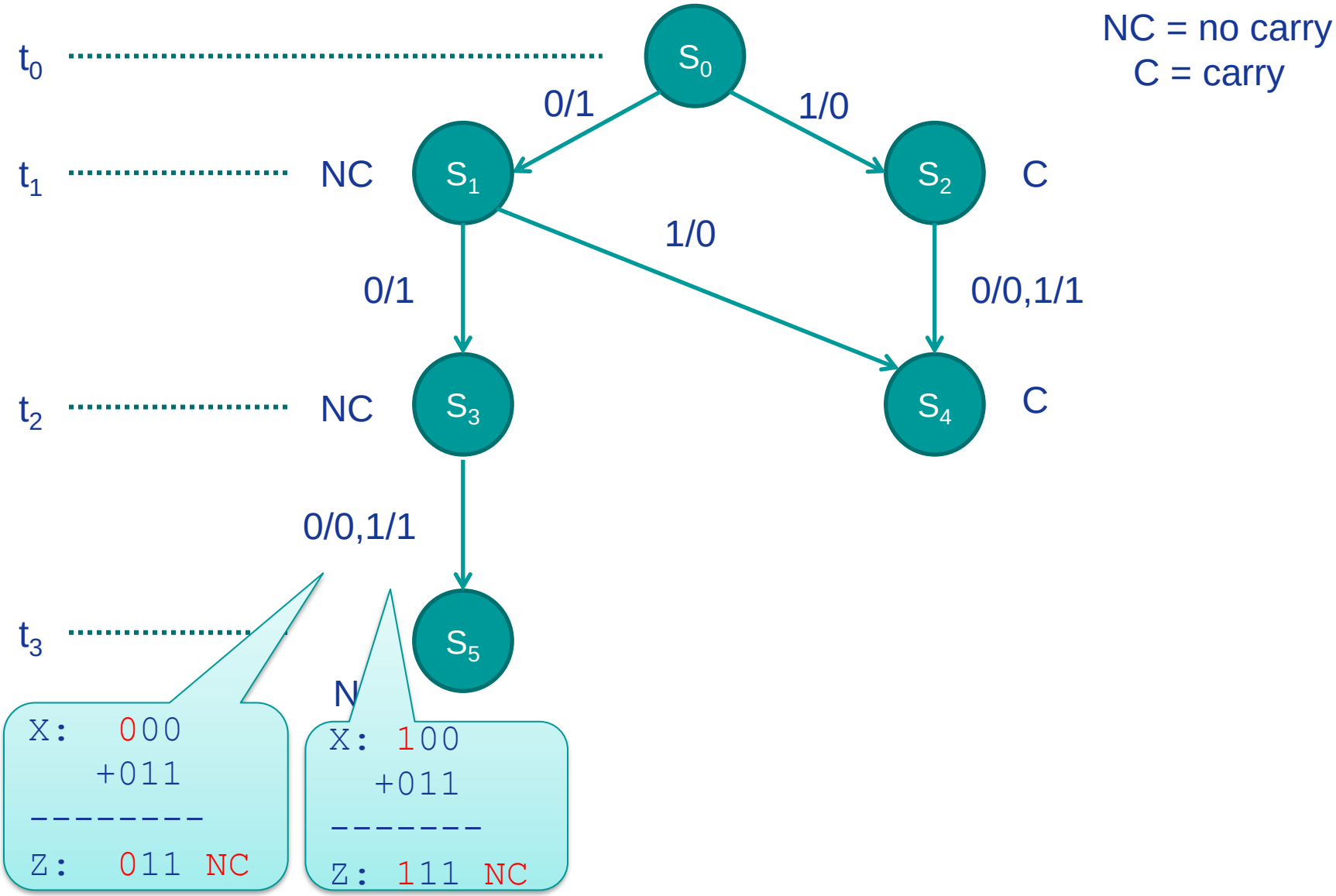
1.7.2 Mealy machine design example: BCD to excess-3 code converter



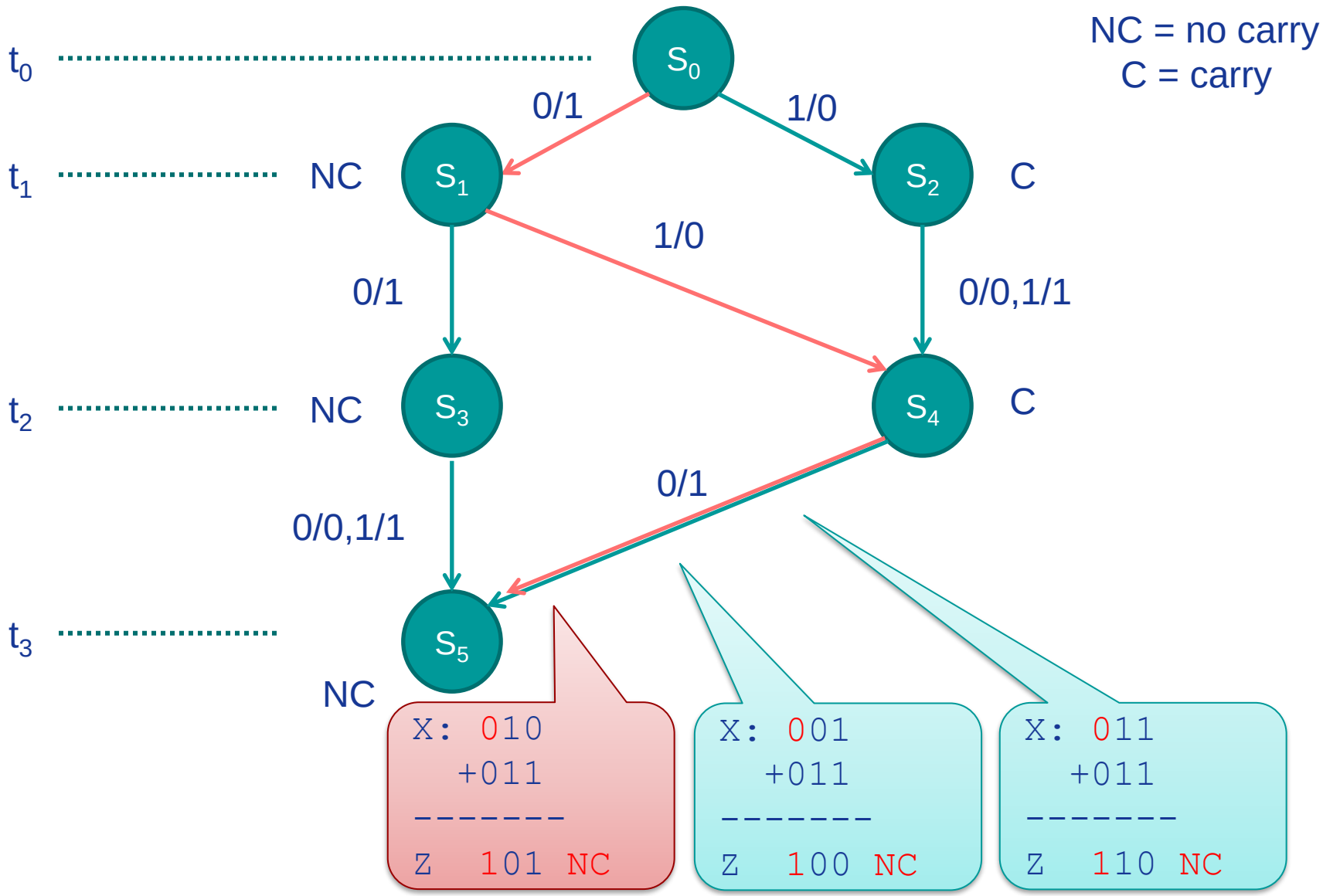
1.7.2 Mealy machine design example: BCD to excess-3 code converter



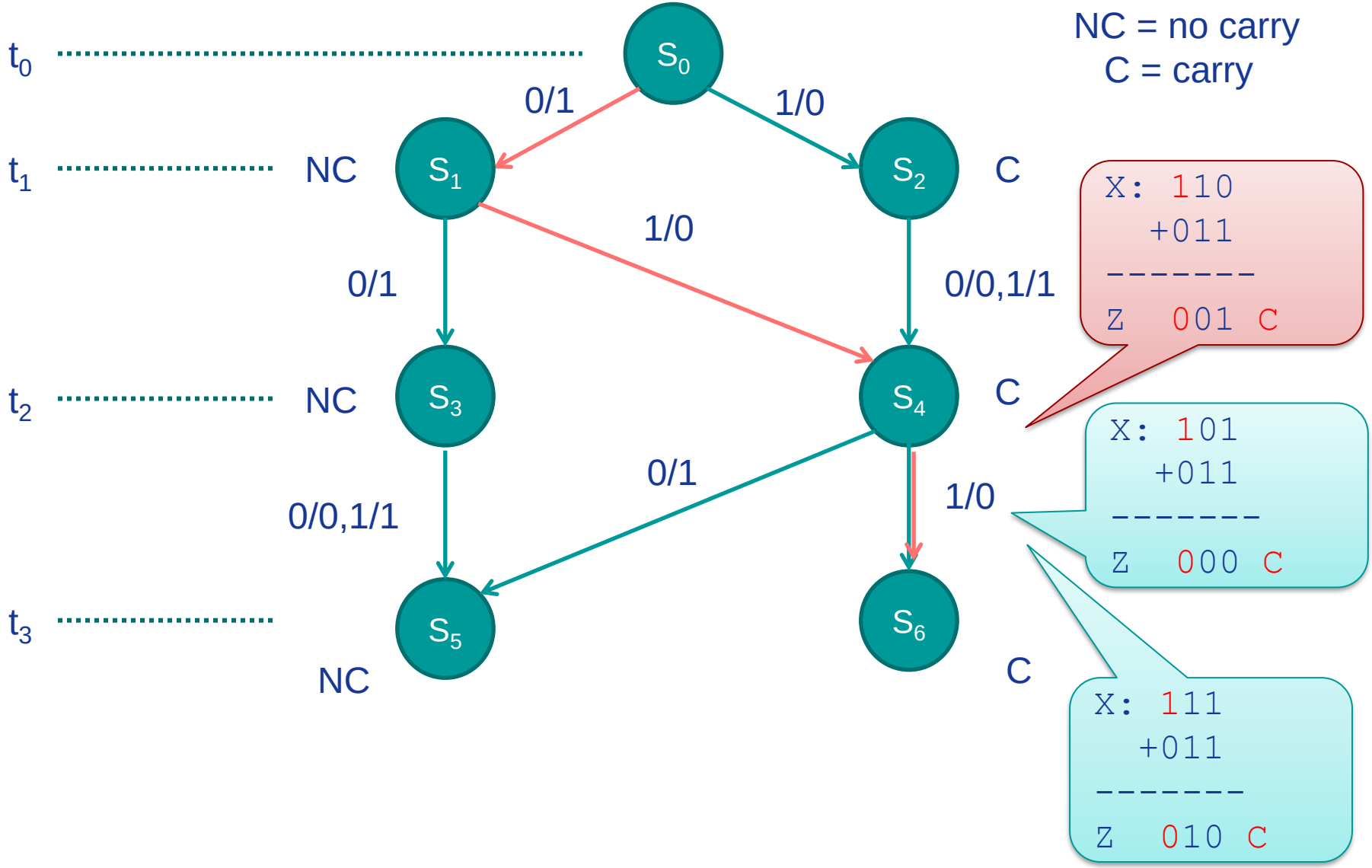
1.7.2 Mealy machine design example: BCD to excess-3 code converter



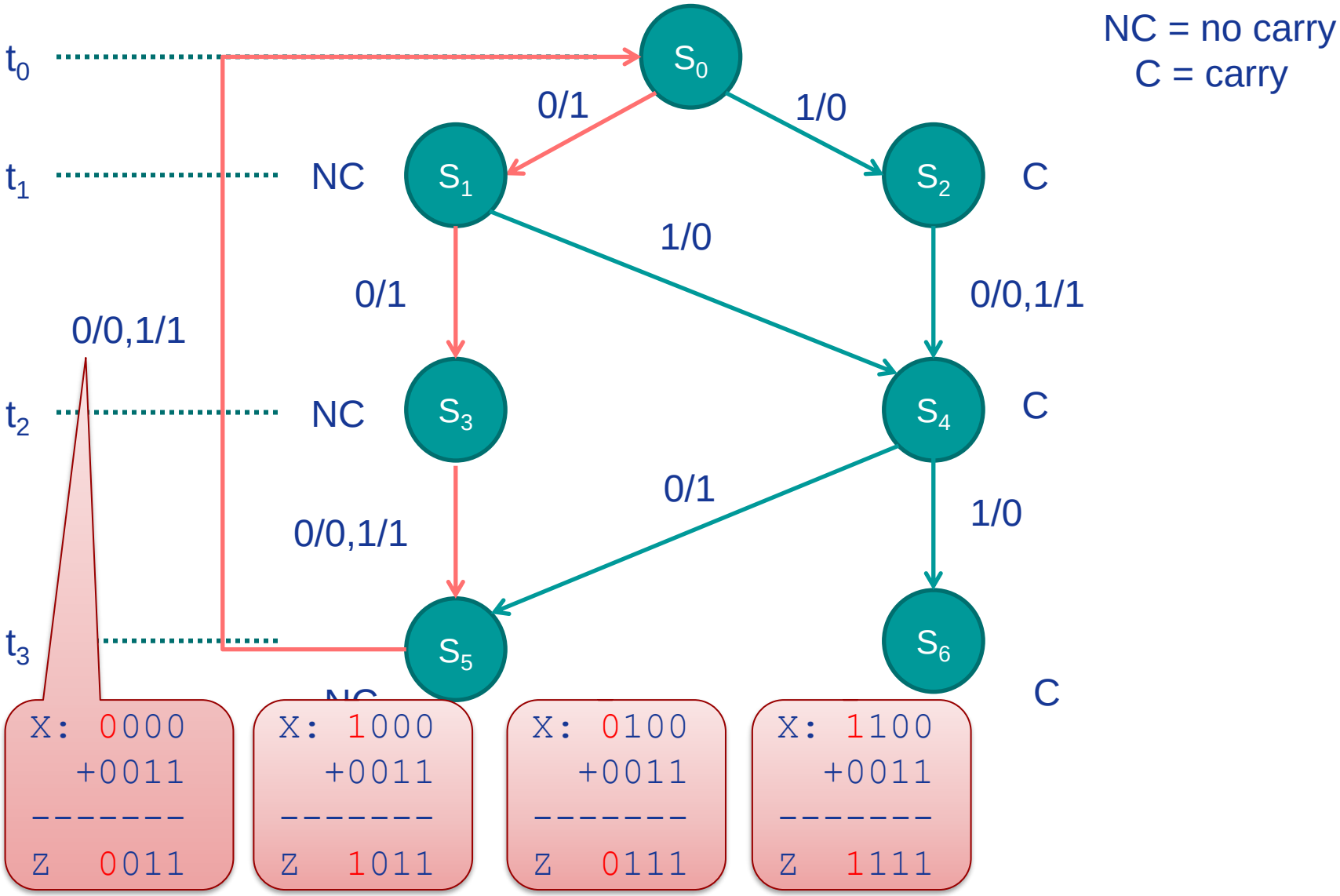
1.7.2 Mealy machine design example: BCD to excess-3 code converter



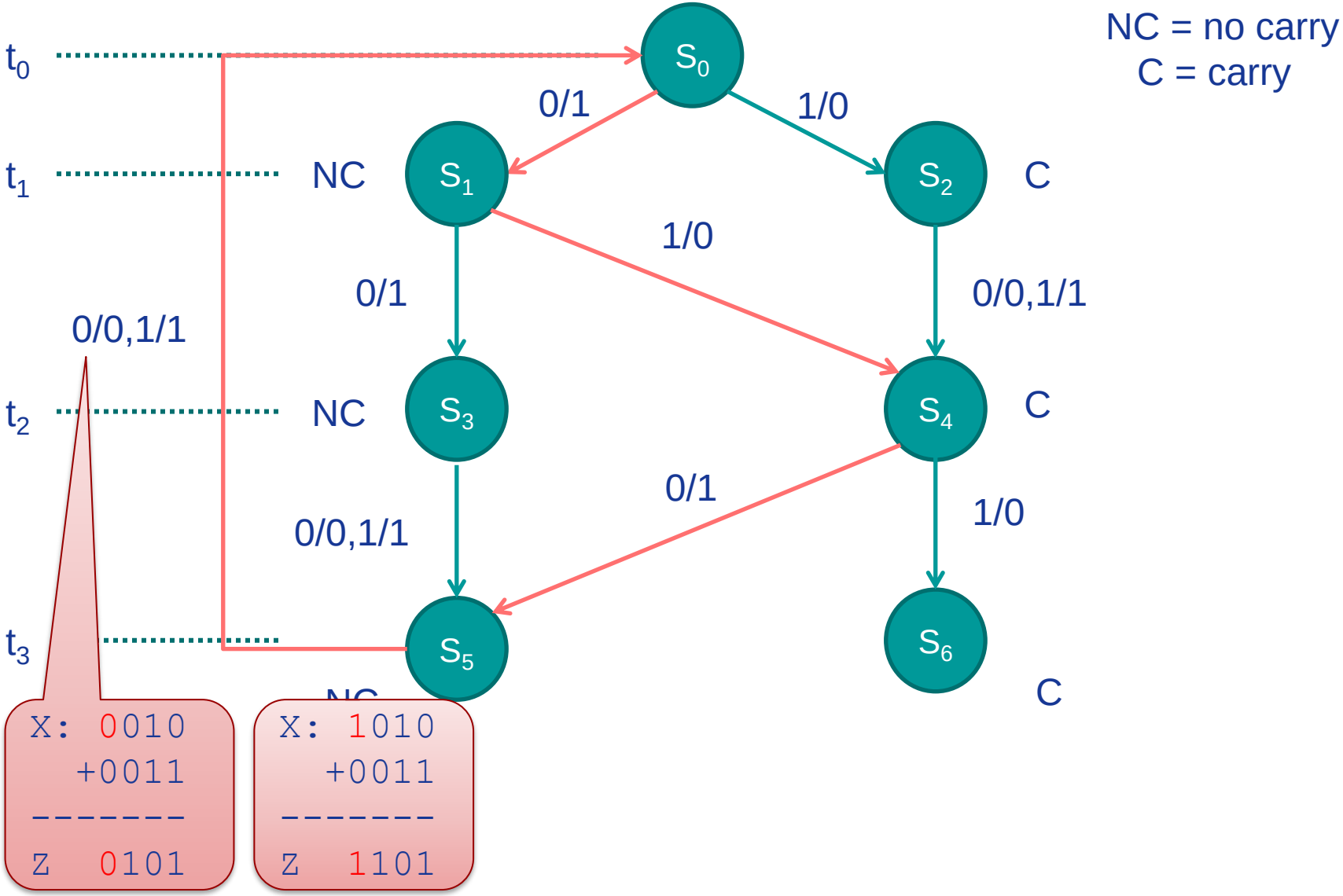
1.7.2 Mealy machine design example: BCD to excess-3 code converter



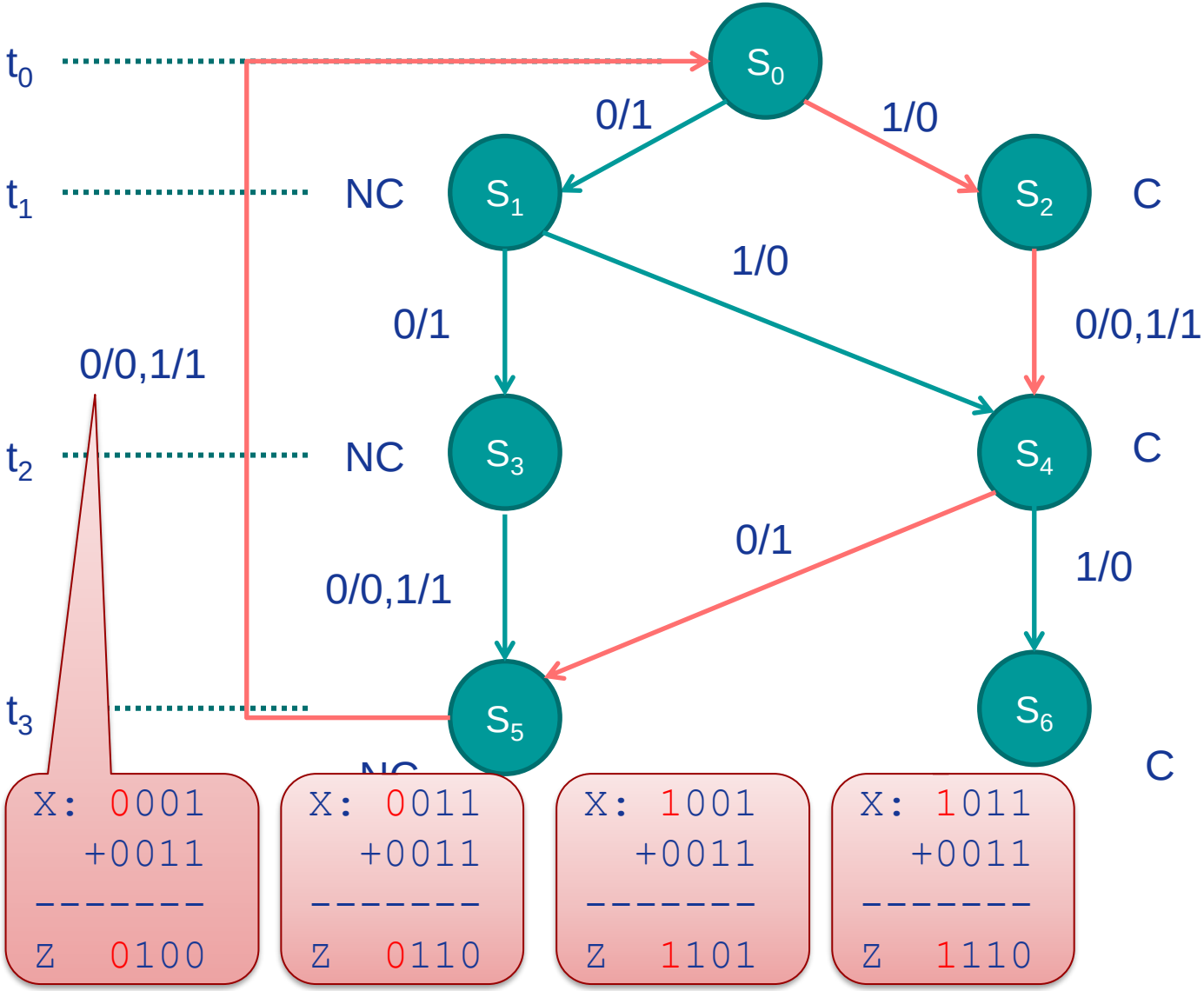
1.7.2 Mealy machine design example: BCD to excess-3 code converter



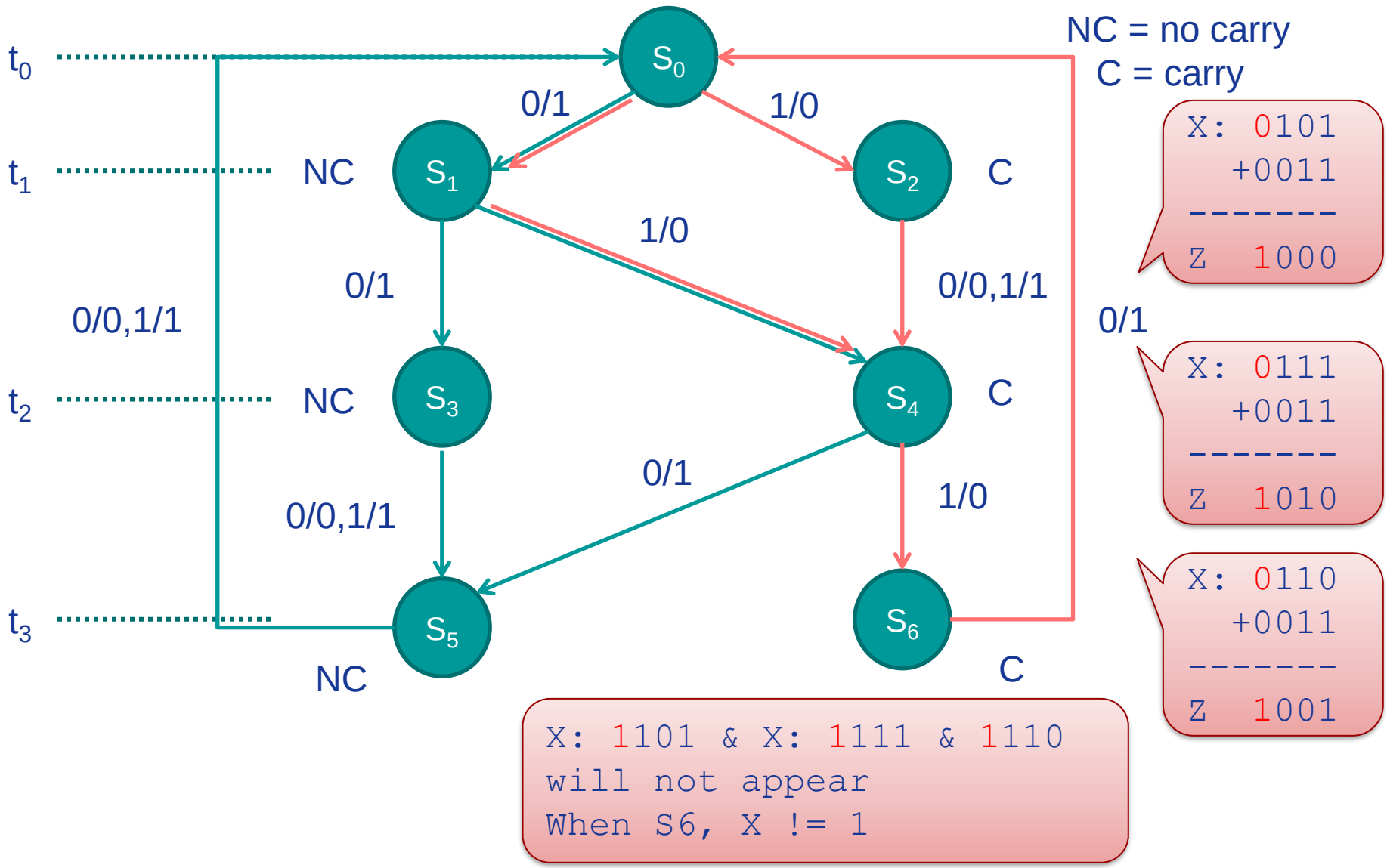
1.7.2 Mealy machine design example: BCD to excess-3 code converter



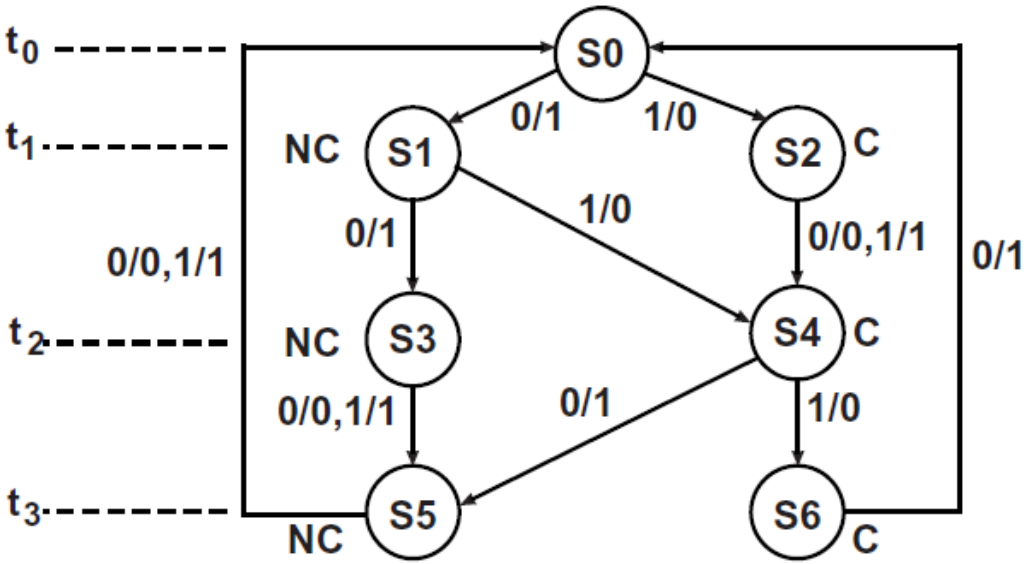
1.7.2 Mealy machine design example: BCD to excess-3 code converter



1.7.2 Mealy machine design example: BCD to excess-3 code converter



An example: BCD to excess-3 code converter



(a) Mealy state graph

PS	NS X = 0	NS X = 1	Z X = 0	Z X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

(b) State Table

7 states

X: 1101, X: 1111, X: 1110 will not appear
When S6, X != 1

One-hot state assignment: 7 flip-flops
Encoded state assignment: 3 flip-flops

An example: BCD to excess-3 code converter

	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

(b) State Table

	$Q_1Q_2Q_3$
S0	000
S1	001
S2	010
S3	011
S4	100
S5	101
S6	110

? Better state assignment

Optimal state assignment

Guidelines	
I. States which have the same next state (NS) for a given input should be given adjacent assignments	(1,2) (3,4) (5,6)

	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1		S4	1	0
S2		S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

In the X=1 column:

- S1 & S2 have NS S4

In the X=0 column:

- S3 & S4 have NS S5
- S5 & S6 have NS S0

Optimal state assignment

Guidelines	
I. States which have the same next state (NS) for a given input should be given adjacent assignments	(1,2) (3,4) (5,6)
II. States that are the next states of the same state should be given adjacent assignments	(1,2) (3,4) (5,6)

	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

- S1 & S2 are NS of S0
- S3 & S4 are NS of S1
- S5 & S6 are NS of S4

Optimal state assignment

Guidelines	
I. States which have the same next state (NS) for a given input should be given adjacent assignments	(1,2) (3,4) (5,6)
II. States that are the next states of the same state should be given adjacent assignments	(1,2) (3,4) (5,6)
III. States that have the same output for a given input should be given adjacent assignments	(0,1,4,6)

	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0			1	0
S1			1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4			1	0
S5	S0	S0	0	1
S6			1	–

Optimal state assignment

Guidelines	
I. States which have the same next state (NS) for a given input should be given adjacent assignments	(1,2) (3,4) (5,6)
II. States that are the next states of the same state should be given adjacent assignments	(1,2) (3,4) (5,6)
III. States that have the same output for a given input should be given adjacent assignments	(0,1,4,6) (2,3,5)

	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2			0	1
S3			0	1
S4	S5	S6	1	0
S5			0	1
S6	S0	–	1	–

Optimal state assignment

Guidelines	
I. States which have the same next state (NS) for a given input should be given adjacent assignments	(1,2) (3,4) (5,6)
II. States that are the next states of the same state should be given adjacent assignments	(1,2) (3,4) (5,6)
III. States that have the same output for a given input should be given adjacent assignments	(0,1,4,6) (2,3,5)

	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

State Assignment Map

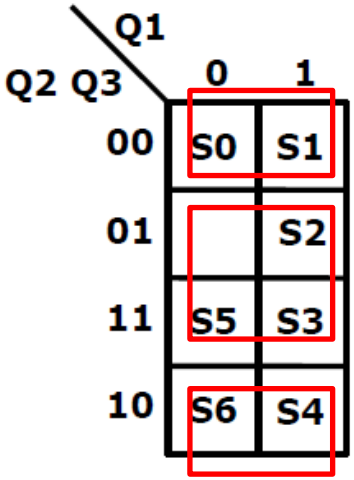
Q2 Q3 \ Q1	0	1
00	S0	S1
01		S2
11	S5	S3
10	S6	S4

Optimal state assignment

Guidelines	
I. States which have the same next state (NS) for a given input should be given adjacent assignments	(1,2) (3,4) (5,6)
II. States that are the next states of the same state should be given adjacent assignments	(1,2) (3,4) (5,6)
III. States that have the same output for a given input should be given adjacent assignments	(0,1,4,6) (2,3,5)

	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

State Assignment Map



Transition table

Q2 Q3 \ Q1	0	1
00	S0	S1
01		S2
11	S5	S3
10	S6	S4

	NS		Z	
PS	X=0	X=1	X=0	X=1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

	$Q1^+Q2^+Q3^+$		Z	
Q1Q2Q3	X=0	X=1	X=0	X=1
000	100	101	1	0
100	111	110	1	0
101	110	110	0	1
111	011	011	0	1
110	011	010	1	0
011	000	000	0	1
010	000	xxx	1	x
001	xxx	xxx	x	x

Karnaugh maps for code converter

Q1Q2Q3	$Q1^+Q2^+Q3^+$ X=0	$Q1^+Q2^+Q3^+$ X=1	Z X=0	Z X=1
000	100	101	1	0
100	111	110	1	0
101	110	110	0	1
111	011	011	0	1
110	011	010	1	0
011	000	000	0	1
010	000	xxx	1	x
001	xxx	xxx	x	x

$Q1^+ = ?$

Q_2Q_3 \ XQ_1	00	01	11	10
00	1	1	1	1
01	X	1	1	X
11	0	0	0	0
10	0	0	0	X

$D_1 = Q_1^+ = Q_2'$

Karnaugh maps for code converter

Q_2Q_3 \ XQ_1	00	01	11	10
00	1	1	1	1
01	X	1	1	X
11	0	0	0	0
10	0	0	0	X

$D_1 = Q_1^+ = Q_2'$

Q_2Q_3 \ XQ_1	00	01	11	10
00	0	1	1	0
01	X	1	1	X
11	0	1	1	0
10	0	1	1	X

$D_2 = Q_2^+ = Q_1$

Q_2Q_3 \ XQ_1	00	01	11	10
00	0	1	0	1
01	X	0	0	X
11	0	1	1	0
10	0	1	0	X

$D_3 = Q_3^+ = Q_1Q_2Q_3 + X'Q_1Q_3' + XQ_1'Q_2'$

Q_2Q_3 \ XQ_1	00	01	11	10
00	1	1	0	0
01	X	0	1	X
11	0	0	1	1
10	1	1	0	X

$Z = X'Q_3' + XQ_3$

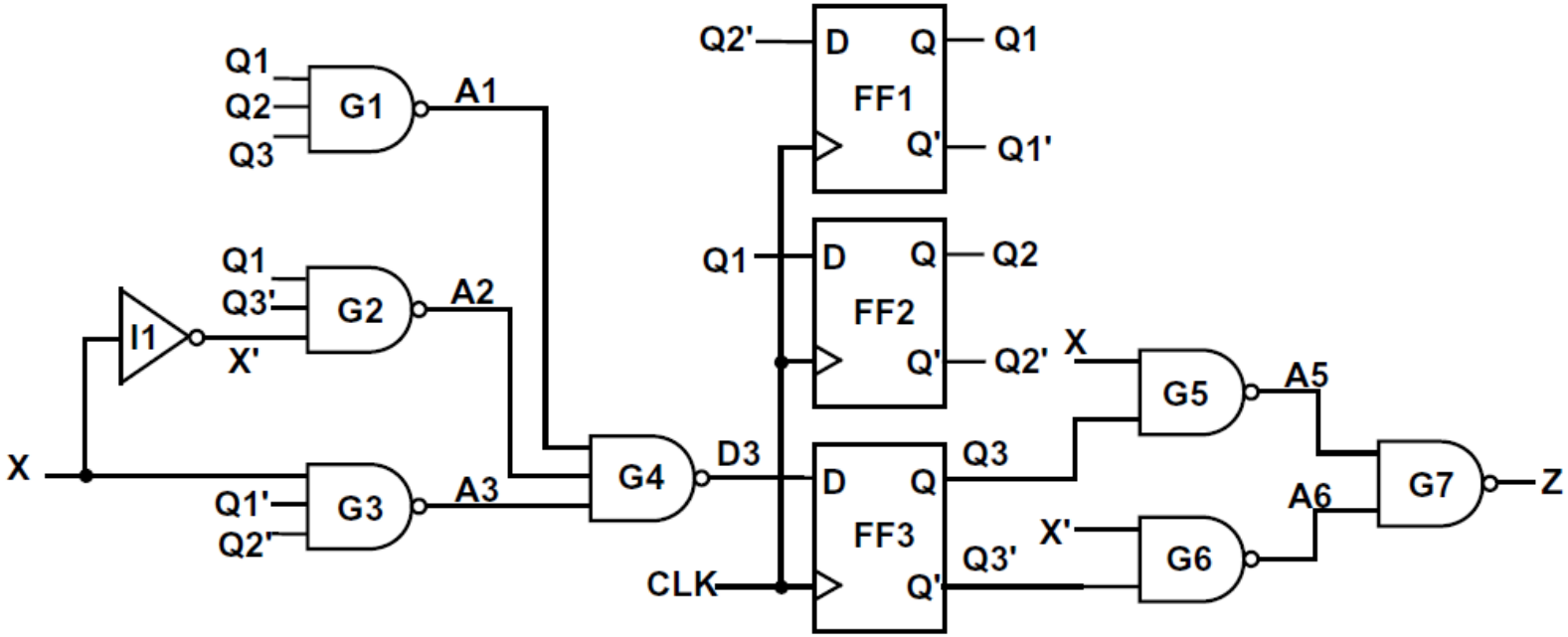
Realization of code converter

$D_1 = Q_1^+ = Q_2'$ $\qquad$ $D_2 = Q_2^+ = Q_1$

$D_3 = Q_3^+ = Q_1Q_2Q_3 + X'Q_1Q_3' + XQ_1'Q_2'$

$Z = X'Q_3' + XQ_3$

3 D flip-flops are used



Without state assignment optimization

S0	000
S1	001
S2	010
S3	011
S4	100
S5	101
S6	110

Without state assignment optimization

	NS		Z	
PS	X = 0	X = 1	X = 0	X = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	–	1	–

Q1Q2 Q3	$Q1^+Q2^+Q3^+$		Z	
	X = 0	X = 1	X = 0	X = 1
000	001	010	1	0
001	011	100	1	0
010	100	100	0	1
011	101	101	0	1
100	101	110	1	0
101	000	000	0	1
110	000	--	1	--

Without state assignment optimization

Q1Q2Q3	$Q1^+Q2^+Q3^+$		Z	
	X=0	X=1	X=0	X=1
000	001	010	1	0
001	011	100	1	0
010	100	100	0	1
011	101	101	0	1
100	101	110	1	0
101	000	000	0	1
110	000	--	1	--

$Q1^+ = ?$

Q1Q2'Q3'

XQ1'Q3'

\XQ1 Q2Q3\	00	01	11	10
00	0	1	1	0
01	0	0	0	1
11	1	X	X	1
10	1	0	X	1

Q1'Q2

State assignment optimization (Comp.)

$Q1^+ = Q1'Q2+Q1Q2'Q3'+XQ1'Q3$

\XQ1 Q2Q3\	00	01	11	10
00	0	1	1	0
01	0	0	0	1
11	1	X	X	1
10	1	0	X	1

Q1Q2'Q3'

XQ1'Q3'

Q1'Q2

Unoptimized

Optimized

$Q1^+ = Q2'$

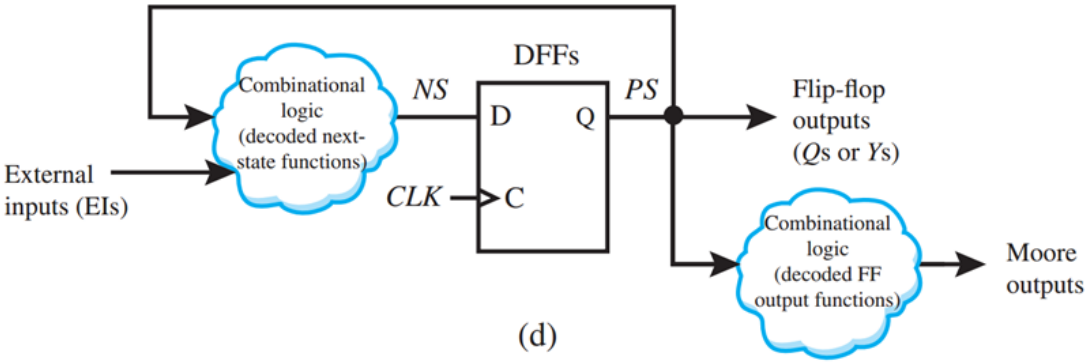
Q_2Q_3 \ XQ_1	00	01	11	10
00	1	1	1	1
01	X	1	1	X
11	0	0	0	0
10	0	0	0	X

$D_1 = Q_1^+ = Q_2'$

Using guidelines I~III tends to clump 1's together on the Karnaugh maps for the next state and output functions

	Contents
1.1	Combinational Logic（组合逻辑）
1.2	Boolean Algebra and Algebra Simplification
1.3	Karnaugh Maps
1.4	Designing with NAND and NOR Gates
1.5	Hazards in Combinational Circuits
1.6	Flip-Flops and Latches
1.7	Mealy Sequential Circuit Design
1.8	**Moore Sequential Circuit Design**
1.9	Equivalent States and Reduction of State Tables
1.10	Sequential Circuit Timing
1.11	Tristate Logic and Busses

Moore sequential machine



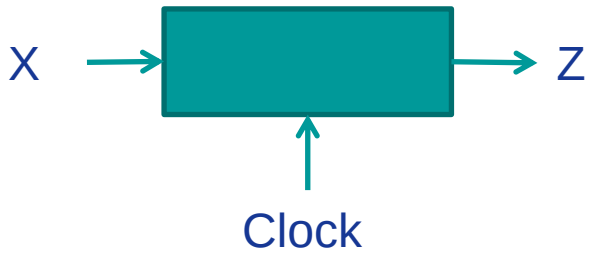
The output depends only on the present state

Moore machines are typically easier to design and debug

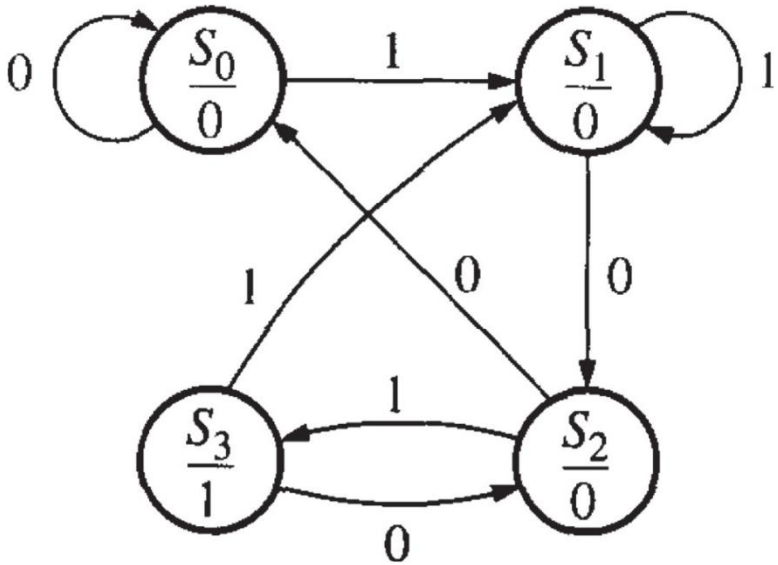
But often contain more states

Moore SM

1.8.1 Moore Machine Design Example 1: Sequence Detector



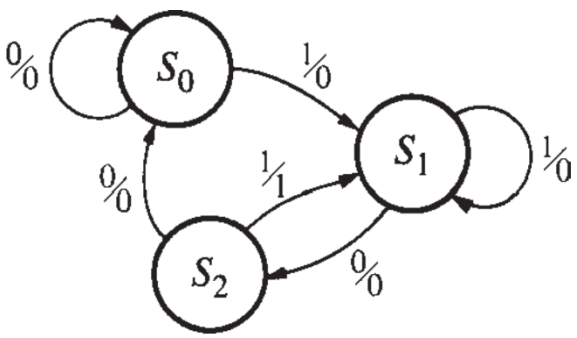
Input	X = 0 0 1 1 0 1 1 0 0 1 0 1 0 1 0 0
Output	Z = 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0



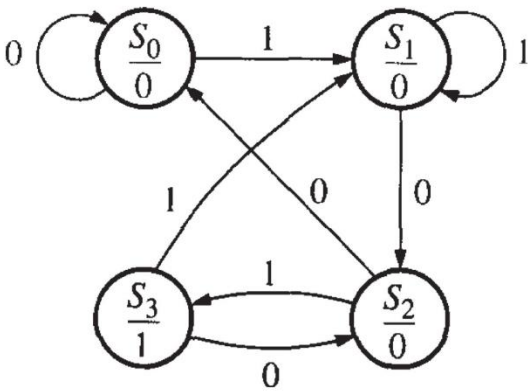
Moore state graph

1.8.1 Moore Machine Design Example 1: Sequence Detector

Mealy state graph



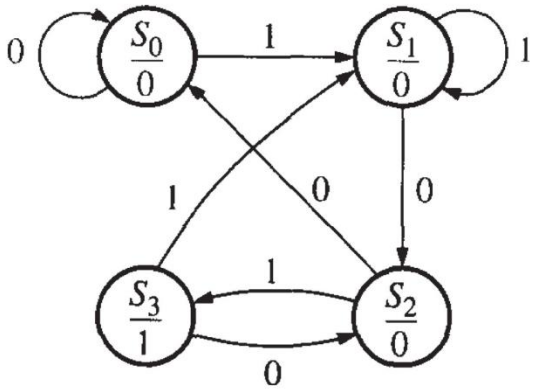
Moore state graph



Present State	Next State		Present Output	
	X=0	X=1	X=0	X=1
S0	S0	S1	0	0
S1	S2	S1	0	0
S2	S0	S1	0	1

Present State	Next State		Present Output
	X=0	X=1	
S0	S0	S1	0
S1	S2	S1	0
S2	S0	S3	0
S3	S2	S1	1

1.8.1 Moore Machine Design Example 1: Sequence Detector

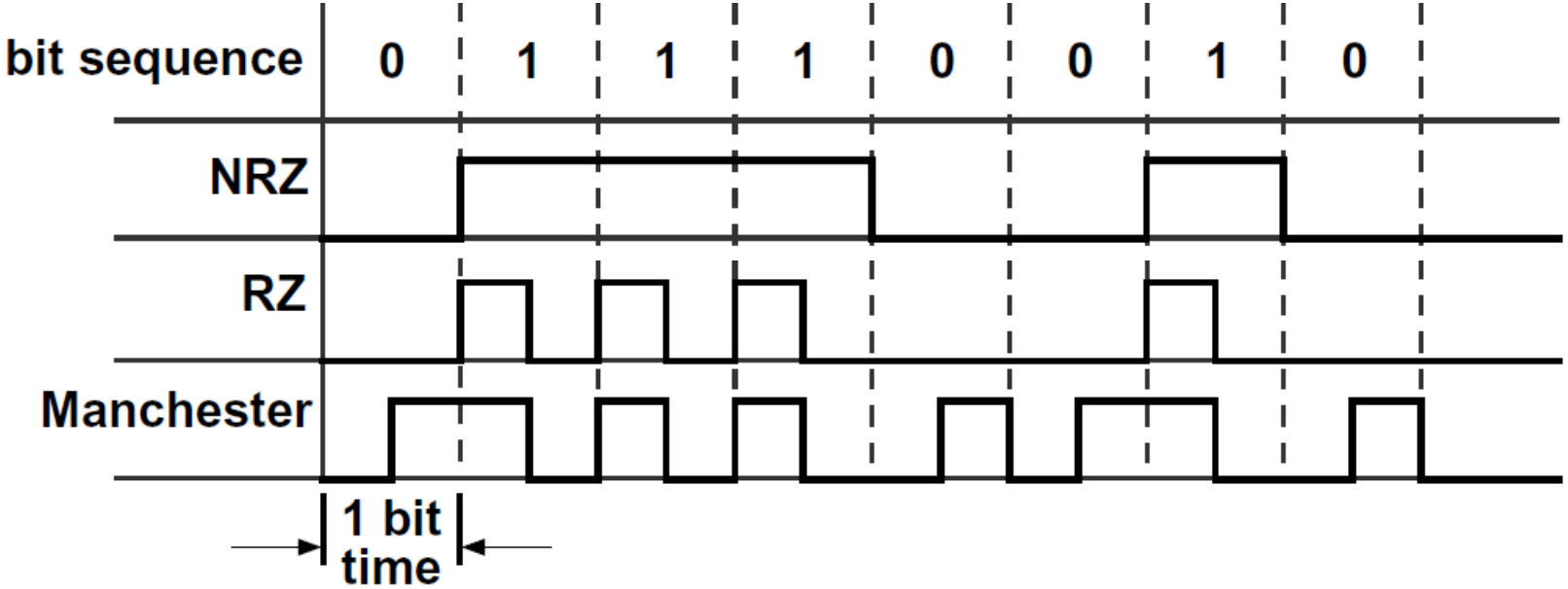


Moore state graph

Present State	Next State		Present Output (Z)
	X=0	X=1	
S0	S0	S1	0
S1	S2	S1	0
S2	S0	S3	0
S3	S2	S1	1

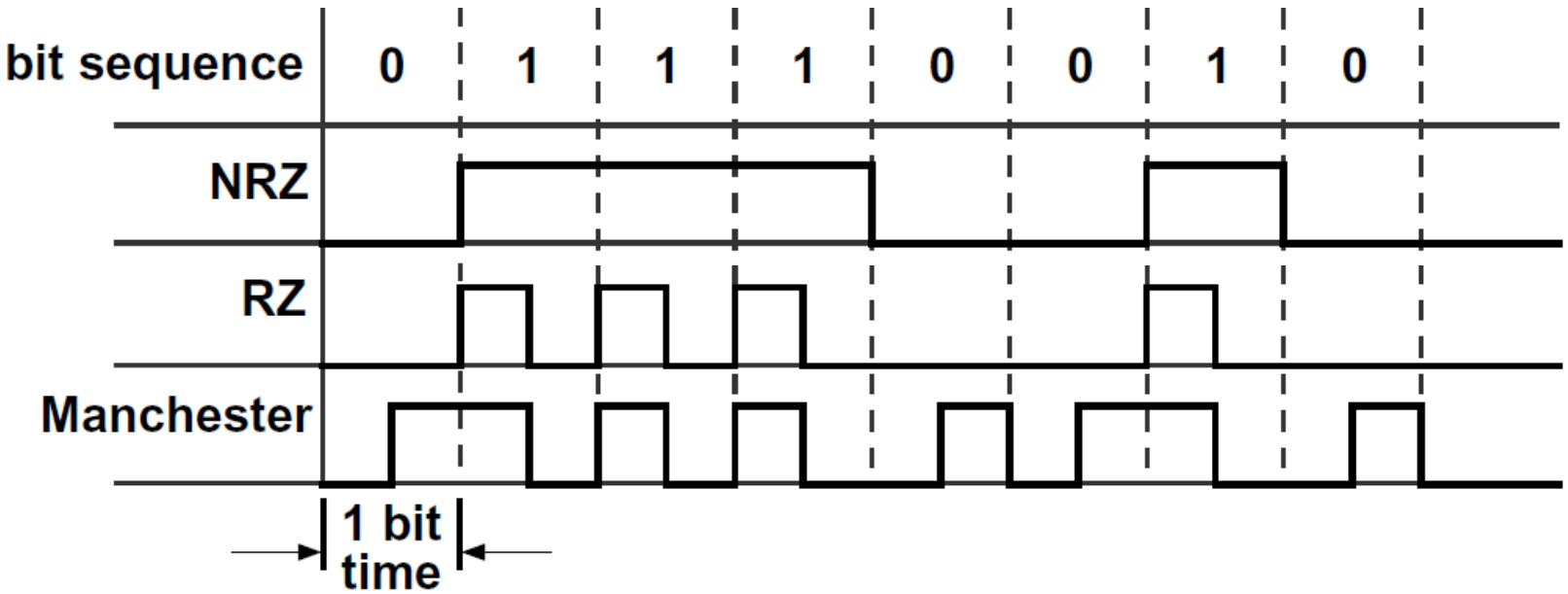
AB	A+B+		Z
	X=0	X=1	
00	00	01	0
01	11	01	0
11	00	10	0
10	11	01	1

1.8.2 An example: NRZ(非归零码) to Manchester code converter



NRZ (nonreturn-to-zero) code: each bit is transmitted for one bit time without any change

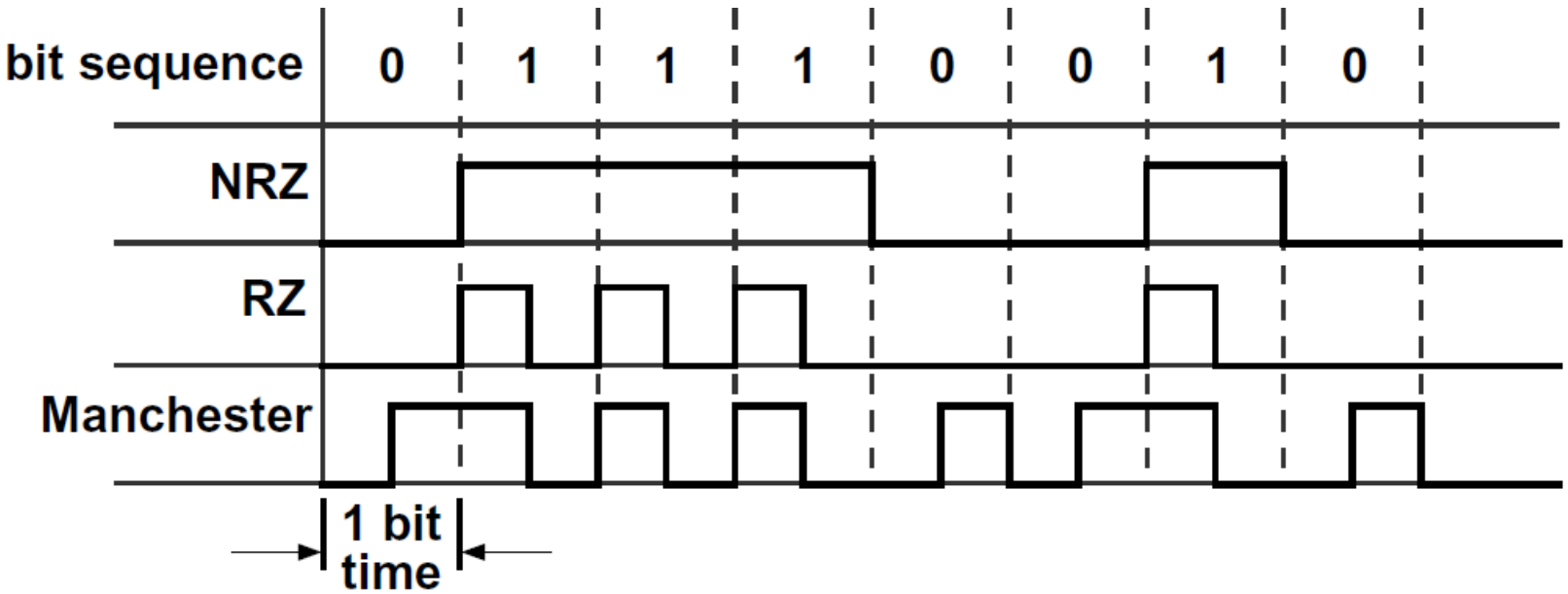
1.8.2 An example: NRZ(非归零码) to Manchester code converter



RZ (return-to-zero) code:

- bit **0**: transmitted as 0 for one full bit time
- bit **1**: transmitted as 1 for the first half of the bit time and then the signal returns to 0 for the second half

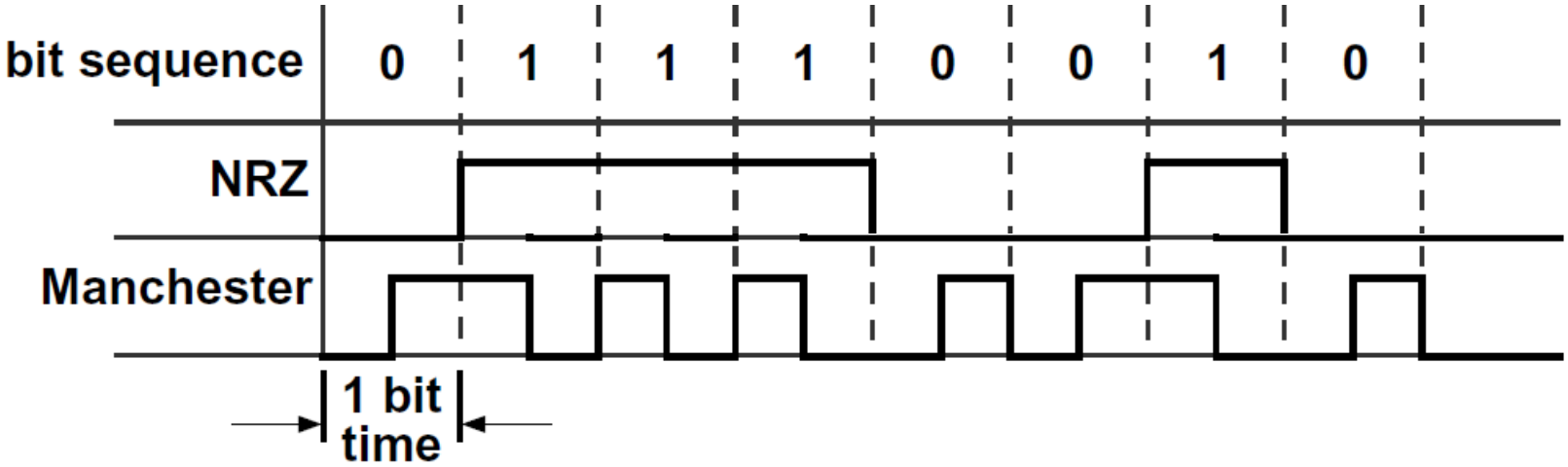
1.8.2 An example: NRZ(非归零码) to Manchester code converter

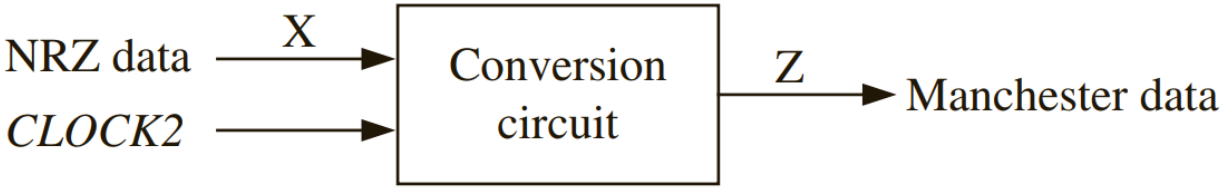


Manchester code:

- bit **0**: transmitted as 0 for the first half of the bit time and a 1 for the second half
- bit **1**: transmitted as a 1 for the first half a 0 for the second half

Moore circuits for code converter





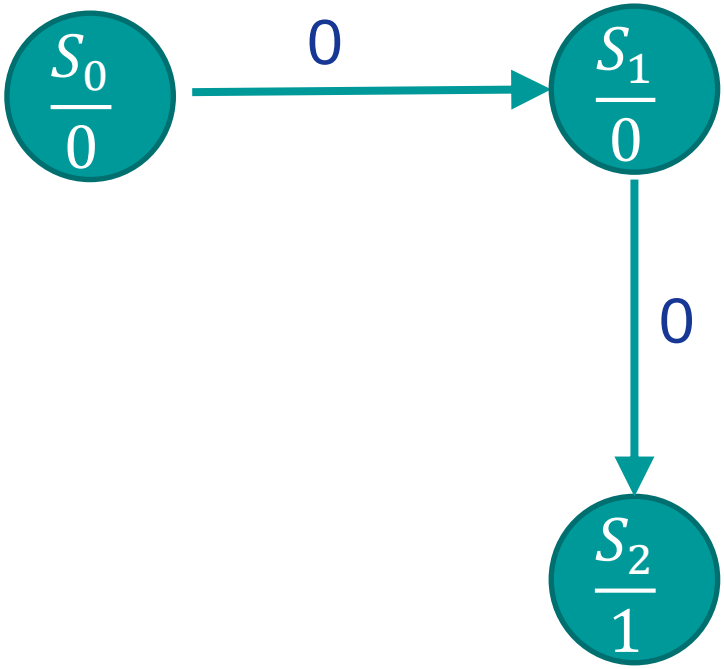
(a) Conversion circuit

CLK2 is twice the frequency of the basic bit clock

If the NRZ bit is 0, it will be 0 for two CLK2 period, and if it is 1, it will be 1 for two CLK2 period

Moore circuits for code converter

- Starting in S0, the only two possible input sequences are 00 and 11



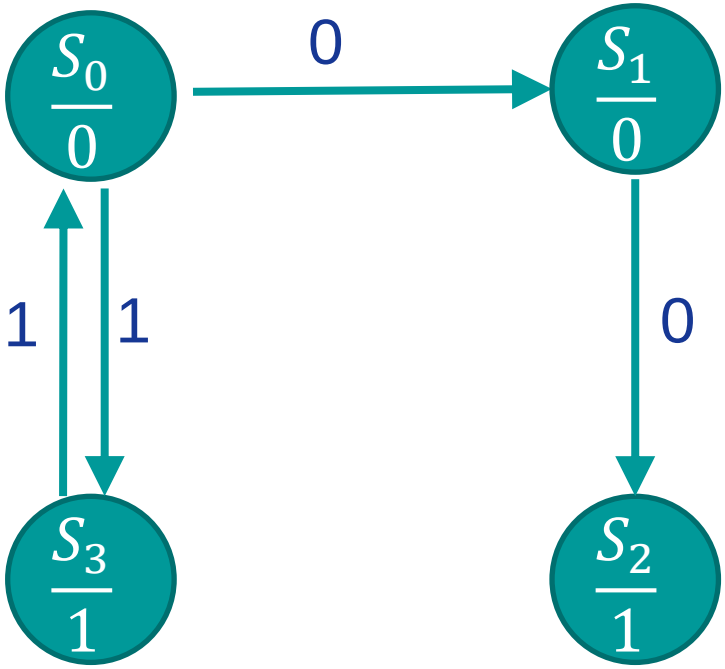
- If the input is 00
- When the first 0 is received, go to S1 and output a 0

- When the second 0 is received, go to S2 and output a 1

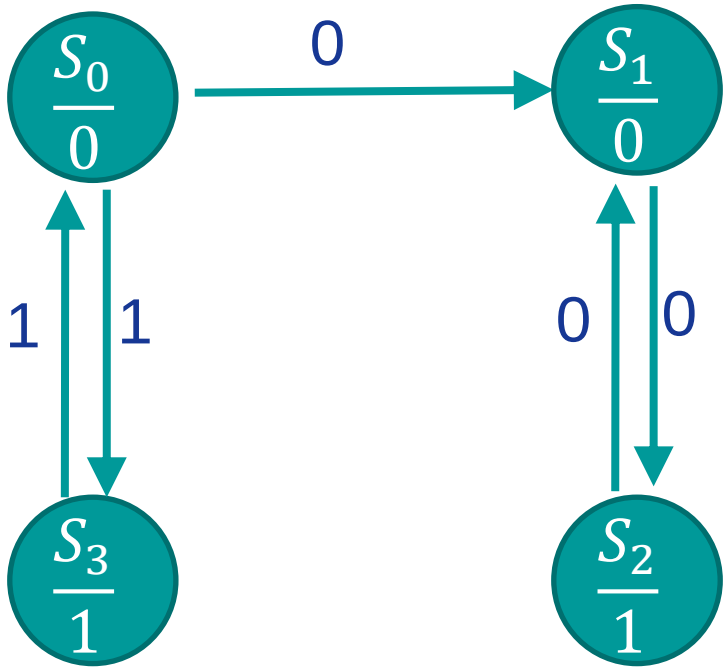
Moore circuits for code converter

- When the second 1 is received, go back to S0 and output a 0

- Starting in S0, if the input is 11
- When the first 1 is received, go to S3 and output a 1



Moore circuits for code converter



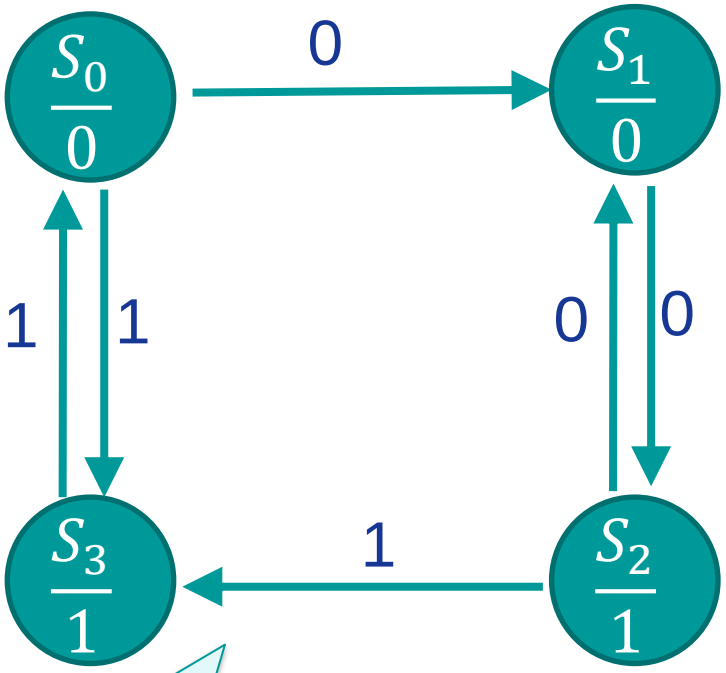
- When in S2, if the input is 00
- When the first 0 is received, go back to S1 to output a 0

- When the second 0 is received, go to S2 to output a 1

- When in S2:

Moore circuits for code converter

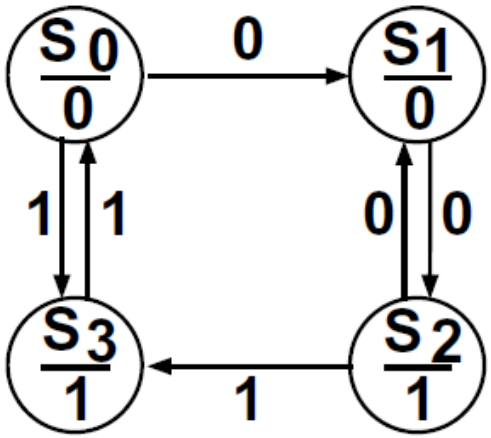
- When the second 1 is received, go to S0 to output a 0



- When in S2, if the input is 11
- When the first 1 is received, go to S3 to output a 1

- When in S2:

Moore circuits for code converter



(b) State Graph

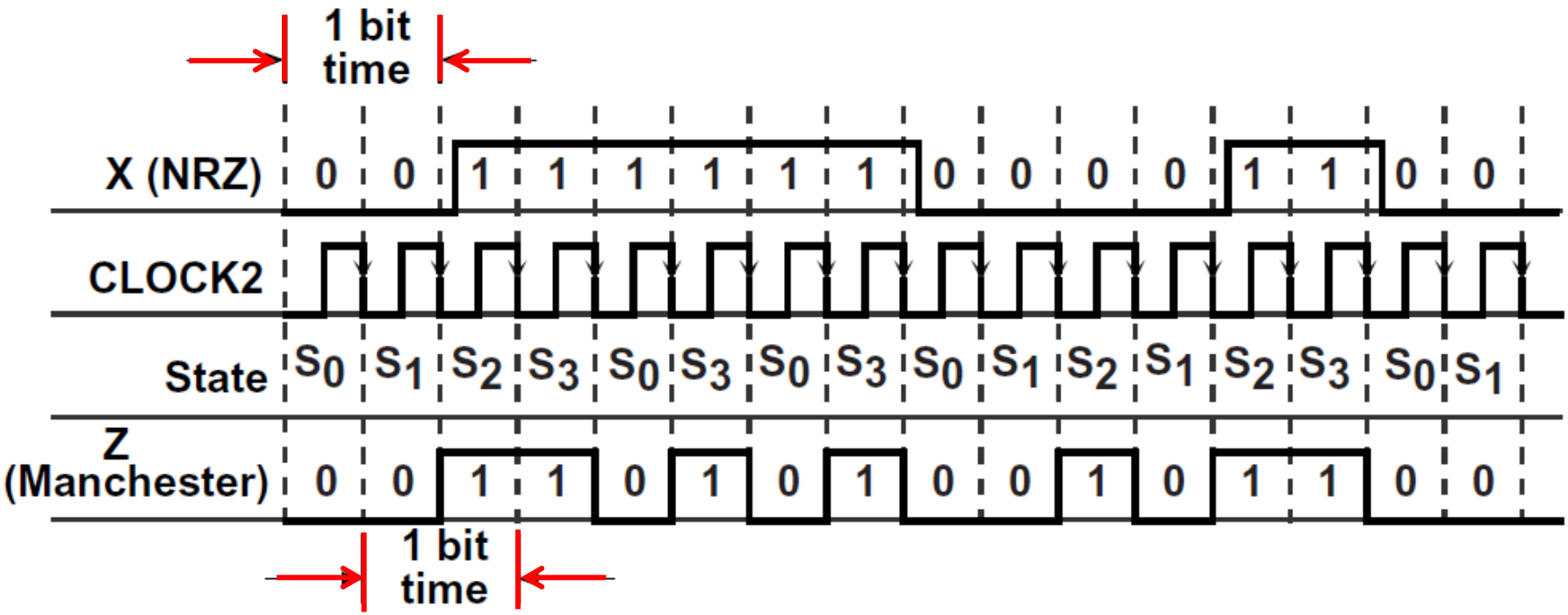
Present State	Next State X = 0	Next State X = 1	Present Output (Z)
S_0	S_1	S_3	0
S_1	S_2	–	0
S_2	S_1	S_3	1
S_3	–	S_0	1

(c) State table

Two don't cares correspond to input sequences that cannot occur

Timing for Moore circuits

- Note that the Manchester output is shifted one clock time w.r.t. the NRZ input
- This shift occurs because a Moore circuit cannot respond to an input until the active edge of the clock occurs

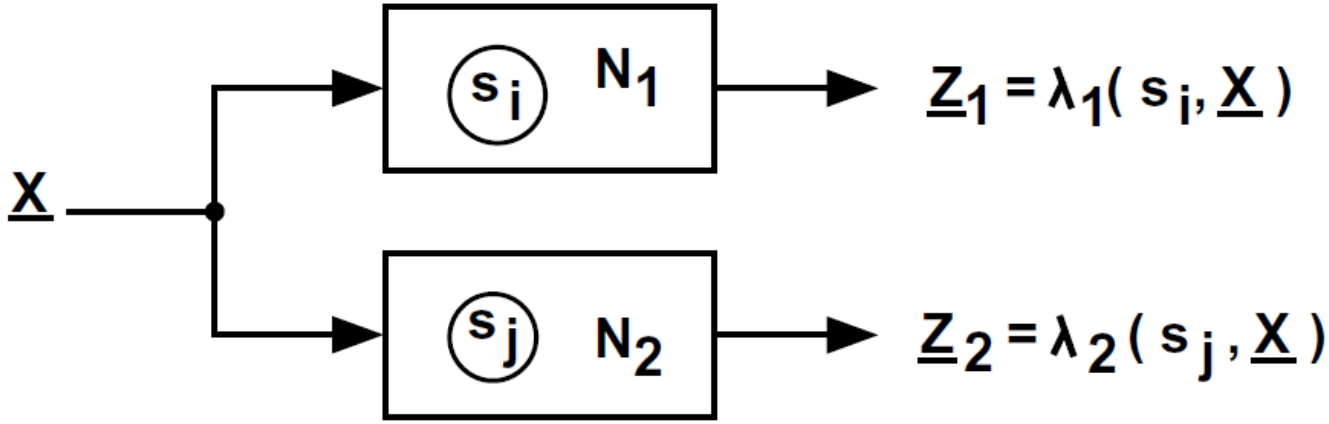


- This is in contrast to a Mealy circuit, for which the output can change after the input changes and before the next clock

	Contents
1.1	Combinational Logic
1.2	Boolean Algebra and Algebra Simplification
1.3	Karnaugh Maps
1.4	Designing with NAND and NOR Gates
1.5	Hazards in Combinational Circuits
1.6	Flip-Flops and Latches
1.7	Mealy Sequential Circuit Design
1.8	Moore Sequential Circuit Design
1.9	**Equivalent States and Reduction of State Tables**
1.10	Sequential Circuit Timing
1.11	Tristate Logic and Busses

Equivalent states (等价状态)

- The number of states in a sequential circuit has a significant impact on its physical implementation
- It is therefore desirable to know when two or more states play identical roles

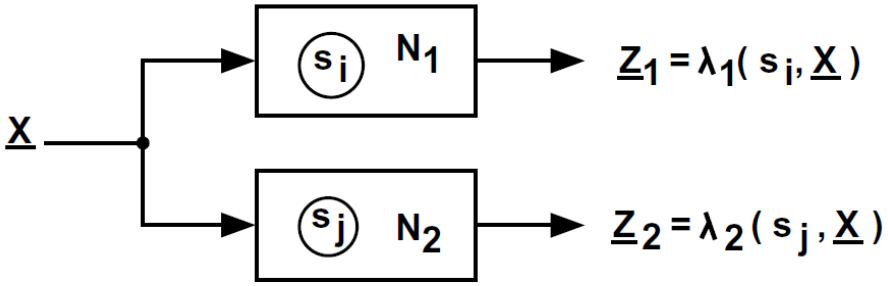


$s_i \equiv s_j$ iff $\underline{Z}_1 = \underline{Z}_2$
for every input sequence $\underline{X}$

Two states are said to be **equivalent** if we cannot tell them apart by observing input and output sequences

This is impractical to test as it requires input sequences of infinite length

Equivalent states



$s_i \equiv s_j$ iff $\underline{Z}_1 = \underline{Z}_2$
for every input sequence $\underline{X}$

State equivalence theorem
$s_i \equiv s_j$ if and only if for every single input X, the outputs are the same and the next states are equivalent.

State equivalence theorem	Definition of equivalence
Look at both output and next state, but need to consider only single inputs rather than input sequence	Consider all input sequences, but we do not need any information about the internal state of the system

State table reduction

Present State	Next State		Present Output	
	X=0	X=1	X=0	X=1
a	c	f	0	0
b	d	e	0	0
c	h	g	0	0
d	b	g	0	0
e	e	b	0	1
f	f	a	0	1
g	c	g	0	1
h	c	f	0	0

$a \equiv h$

State table reduction

Present State	Next State		Present Output	
	X=0	X=1	X=0	X=1
a	c	f	0	0
b	d	e	0	0
c	~~h~~a	g	0	0
d	b	g	0	0
e	e	b	0	1
f	f	a	0	1
g	c	g	0	1
~~h~~	~~c~~	~~f~~	~~0~~	~~0~~

State table reduction

Present State	Next State		Present Output	
	X=0	X=1	X=0	X=1
a	c	f	0	0
b	d	e	0	0
c	a	g	0	0
d	b	g	0	0
e	e	b	0	1
f	f	a	0	1
g	c	g	0	1

? a ≡ b

State table reduction

a ≡ b iff c ≡ d and e ≡ f

Present State	Next State		Present Output	
	X=0	X=1	X=0	X=1
a	c	f	0	0
b	d	e	0	0
c	a	g	0	0
d	b	g	0	0
e	e	b	0	1
f	f	a	0	1
g	c	g	0	1

State table reduction

Present State	Next State X = 0	Next State X = 1	Present Output X = 0	Present Output X = 1
a	c	f	0	0
b	d	e	0	0
c	~~h~~ a	g	0	0
d	b	g	0	0
e	e	b	0	1
f	f	a	0	1
g	c	g	0	1
~~h~~	~~c~~	~~f~~	~~0~~	~~0~~

Implication chart (蕴涵表)

	a	b	c	d	e	f
b	c-d e-f					
c	f-g	a-d e-g				
d	b-c f-g	e-g	a-b			
e						
f					a-b	
g					c-e b-g	e-f a-g

State table reduction

Present State	Next State X = 0	Next State X = 1	Present Output X = 0	Present Output X = 1
a	c	f	0	0
b	d	e	0	0
c	~~h~~ a	g	0	0
d	b	g	0	0
e	e	b	0	1
f	f	a	0	1
g	c	g	0	1
~~h~~	~~c~~	~~f~~	~~0~~	~~0~~

Implication chart (蕴涵表)

	a	b	c	d	e	f
b	c-d e-f					
c	f-g	a-d e-g				
d	b-c f-g	e-g	a-b			
e						
f					a-b	
g					c-e b-g	e-f a-g

State table reduction

	a	b	c	d	e	f
b		c-d e-f				
c		f-g	a-d e-g			
d		b-c f-g	e-g	a-b		
e						
f					a-b	
g					c-e b-g	e-f a-g

	a	b	c	d	e	f
b		c-d e-f				
c		f-g	a-d e-g			
d		b-c f-g	e-g	a-b		
e						
f					a-b	
g					c-e b-g	e-f a-g

State table reduction

	a	b	c	d	e	f
b	c-d e-f					
c	f-g	a-d e-g				
d	b-c f-g	e-g	a-b			
e						
f					a-b	
g					c-e b-g	e-f a-g

	a	b	c	d	e	f
b	c-d e-f					
c	f-g	a-d e-g				
d	b-c f-g	e-g	a-b			
e						
f					a-b	
g					c-e b-g	e-f a-g

State table reduction

	a	b	c	d	e	f
b		c-d e-f				
c		f-g	a-d e-g			
d		b-c f-g	e-g	a-b		
e						
f					a-b	
g					c-e b-g	e-f a-g

	a	b	c	d	e	f
b		c-d e-f				
c		~~f-g~~	~~a-d~~ ~~e-g~~			
d		~~b-c~~ ~~f-g~~	~~e-g~~	a-b		
e						
f					a-b	
g					~~c-e~~ ~~b-g~~	~~e-f~~ ~~a-g~~

State table reduction

	a	b	c	d	e	f
b	c-d e-f					
c	f-g	a-d e-g				
d	b-c f-g	e-g	a-b			
e						
f					a-b	
g					c-e b-g	e-f a-g

	a	b	c	d	e	f
b	c-d e-f					
c	~~f-g~~	~~a-d~~ ~~e-g~~				
d	~~b-c~~ ~~f-g~~	~~e-g~~	a-b			
e						
f					a-b	
g					~~c-e~~ ~~b-g~~	~~e-f~~ ~~a-g~~

a≡b ⇔ c≡d & e≡f

c≡d ⇔ a≡b

e≡f ⇔ a≡b

a ≡ b, c ≡ d, e ≡ f

State table reduction

Present State	Next State X = 0	Next State X = 1	Present Output X = 0	Present Output X = 1
a	c	f	0	0
b	d	e	0	0
c	~~h~~ a	g	0	0
d	b	g	0	0
e	e	b	0	1
f	f	a	0	1
g	c	g	0	1
~~h~~	~~c~~	~~f~~	~~0~~	~~0~~

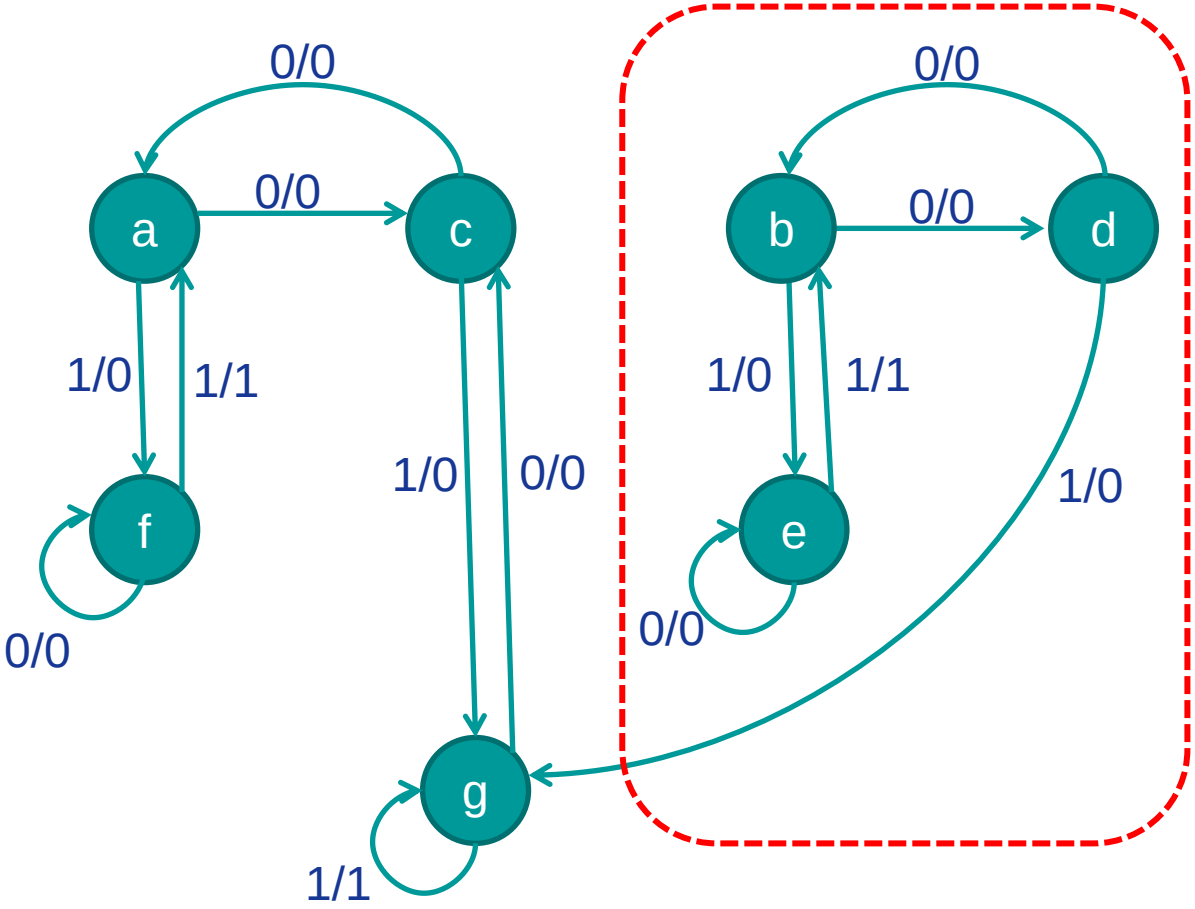
$a \equiv b, \quad c \equiv d, \quad e \equiv f$

Present State	X = 0	1	X = 0	1
a	c	e	0	0
c	a	g	0	0
e	e	a	0	1
g	c	g	0	1

Final Reduced Table

State table reduction

PS	NS		PO	
	X=0	X=1	X=0	X=1
a	c	f	0	0
b	d	e	0	0
c	a	g	0	0
d	b	g	0	0
e	e	b	0	1
f	f	a	0	1
g	c	g	0	1



	Contents
1.1	Combinational Logic
1.2	Boolean Algebra and Algebra Simplification
1.3	Karnaugh Maps
1.4	Designing with NAND and NOR Gates
1.5	Hazards in Combinational Circuits
1.6	Flip-Flops and Latches
1.7	Mealy Sequential Circuit Design
1.8	Moore Sequential Circuit Design
1.9	Equivalent States and Reduction of State Tables
1.10	**Sequential Circuit Timing**
1.11	Tristate Logic and Busses

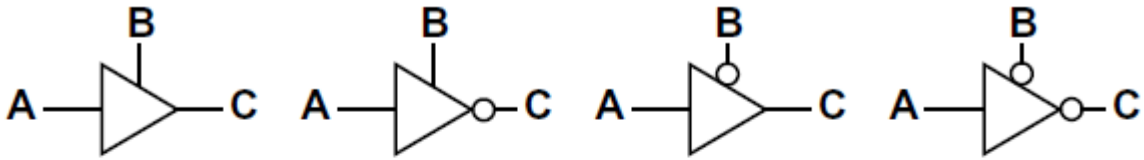
	Contents
1.1	Combinational Logic
1.2	Boolean Algebra and Algebra Simplification
1.3	Karnaugh Maps
1.4	Designing with NAND and NOR Gates
1.5	Hazards in Combinational Circuits
1.6	Flip-Flops and Latches
1.7	Mealy Sequential Circuit Design
1.8	Moore Sequential Circuit Design
1.9	Equivalent States and Reduction of State Tables
1.10	Sequential Circuit Timing
1.11	**Tristate Logic and Busses**

Tristate(三态) logic and busses

If we connect outputs of two gates or flip-flops together, the circuit will not operate properly

When we need to connect to multiple gate outputs to the same wire, one way to do that is by using **tristate buffers**

Tristate buffers are gates with a high impedance state (Hi-Z) in addition to ‘0’ and ‘1’



B	A	C
0	0	Hi-Z
0	1	Hi-Z
1	0	0
1	1	1

(a)

B	A	C
0	0	Hi-Z
0	1	Hi-Z
1	0	1
1	1	0

(b)

B	A	C
0	0	0
0	1	1
1	0	Hi-Z
1	1	Hi-Z

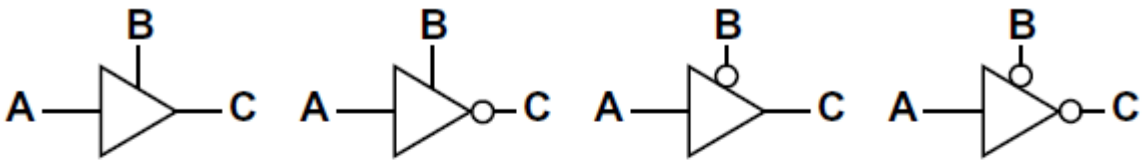
(c)

B	A	C
0	0	1
0	1	0
1	0	Hi-Z
1	1	Hi-Z

(d)

Tristate(三态) logic and busses

- B: control input used to enable or disable the buffer output
- Tristate buffer outputs can be connected, provided that only one output is enabled at a time



B	A	C
0	0	Hi-Z
0	1	Hi-Z
1	0	0
1	1	1

(a)

B	A	C
0	0	Hi-Z
0	1	Hi-Z
1	0	1
1	1	0

(b)

B	A	C
0	0	0
0	1	1
1	0	Hi-Z
1	1	Hi-Z

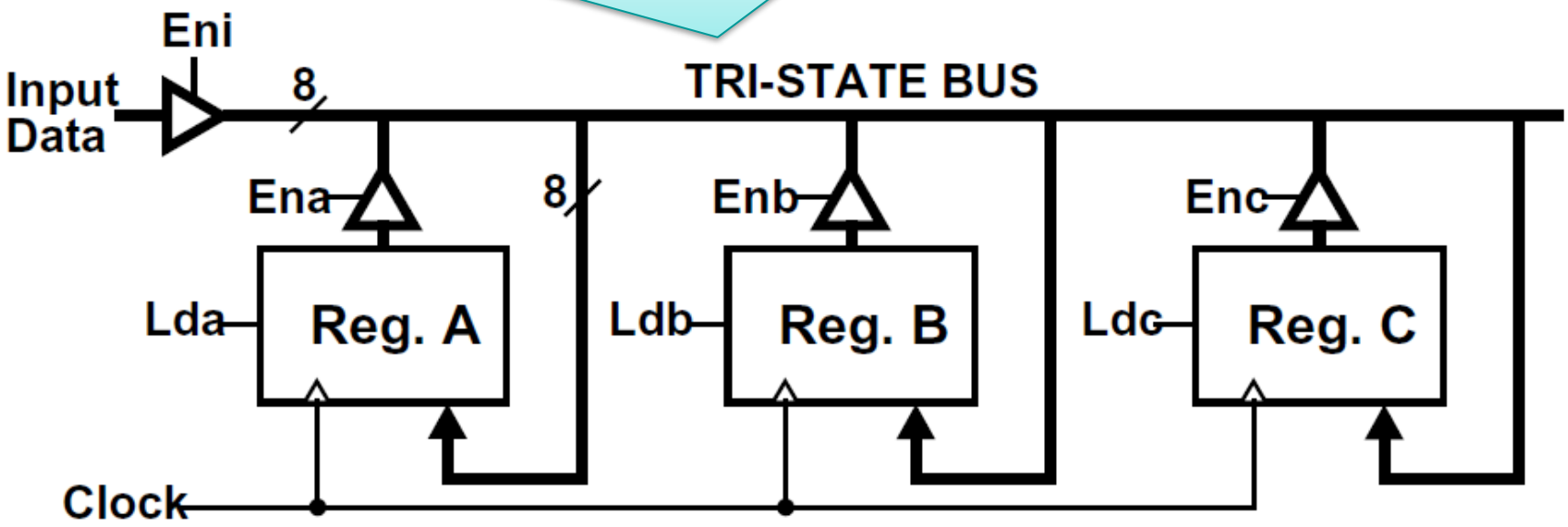
(c)

B	A	C
0	0	1
0	1	0
1	0	Hi-Z
1	1	Hi-Z

(d)

Tristate logic and busses

Only one group of buffers is enabled at a time



Data is loaded into a register only when its load input is 1 and the register is clocked