Asynchronous Sequential Logic

UNIT-5

Introduction

- **Synchronous Sequential Circuits**

The change of internal state occurs in response to the synchronized clock pulse.

Memory elements are clocked flip-flops.

- **Asynchronous Sequential Circuits**

The change of internal state occurs when there is a change in the input variables.

Memory elements are unclocked flip-flops or time-delay elements.

Introduction

- **Synchronous Sequential Circuits**

Timing problems are eliminated by triggering all flip-flops with pulse edge.

- **Asynchronous Sequential Circuits**

Care must be taken to ensure that each new state is stable even though a feedback path exists.

Higher speed , More economical

Introduction

- **Asynchronous Sequential Circuits**

When an input variable changes in value, the *y* secondary variables do not change instantaneously.

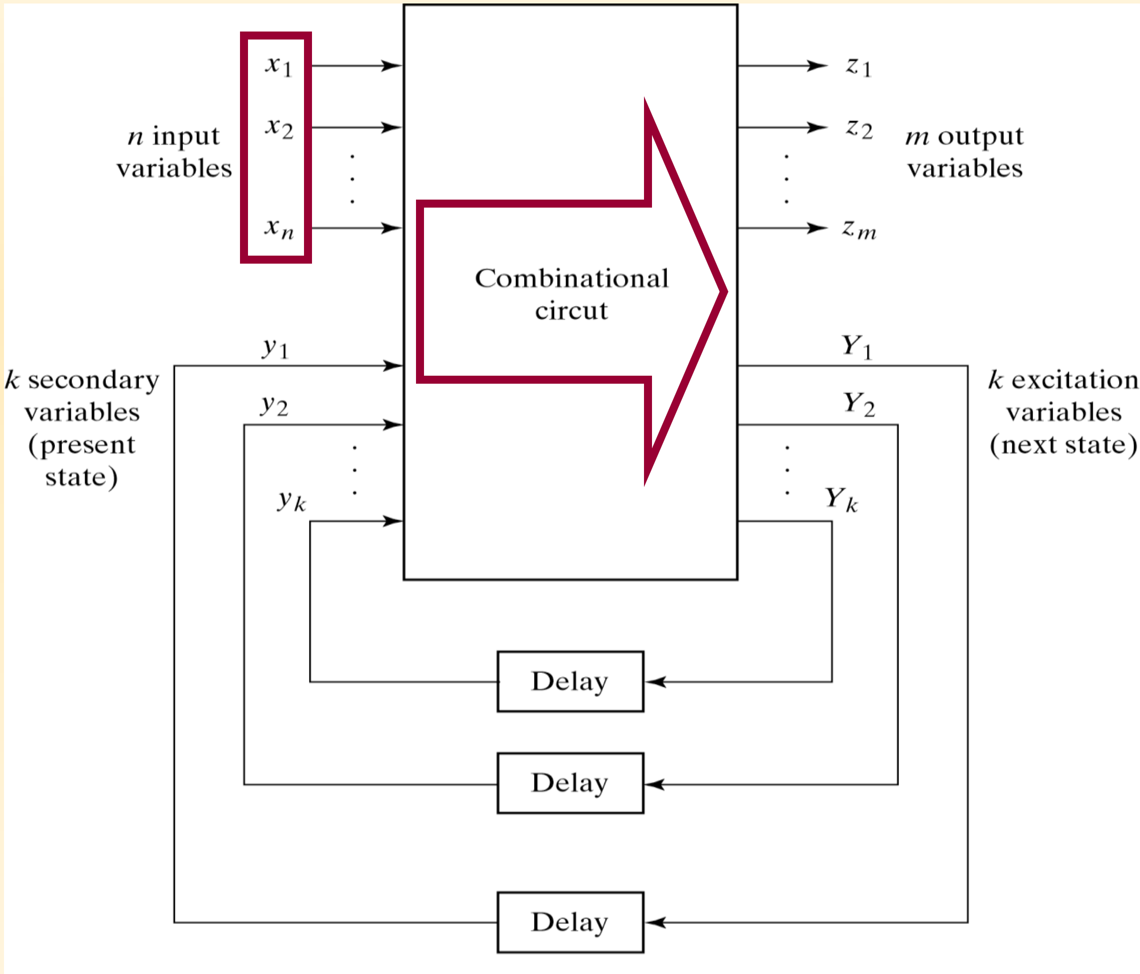


Fig. 9-1 Block Diagram of an Asynchronous Sequential Circuit

Introduction

- **Asynchronous Sequential Circuits**

In steady-state condition, the *y*'s and the *Y*'s are the same, but during transition they are not.

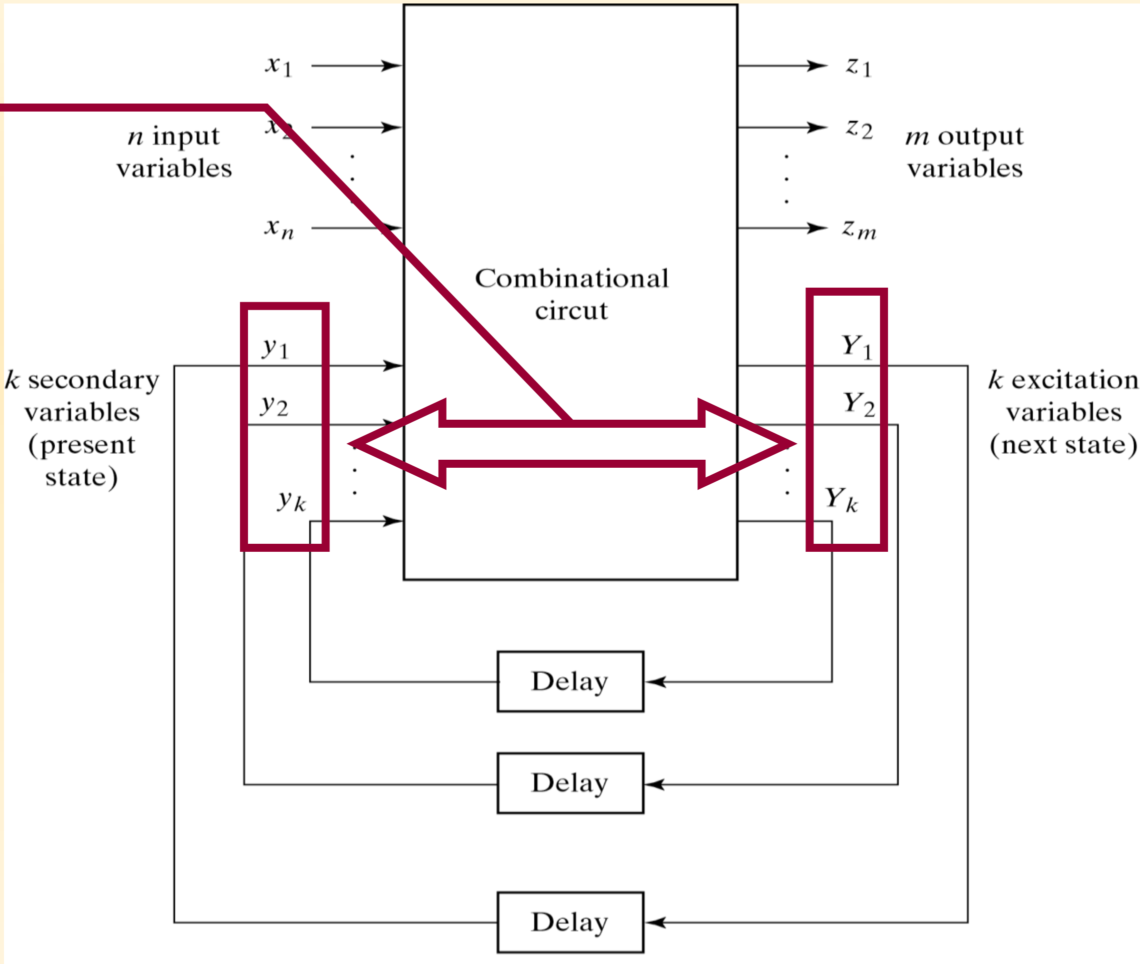


Fig. 9-1 Block Diagram of an Asynchronous Sequential Circuit

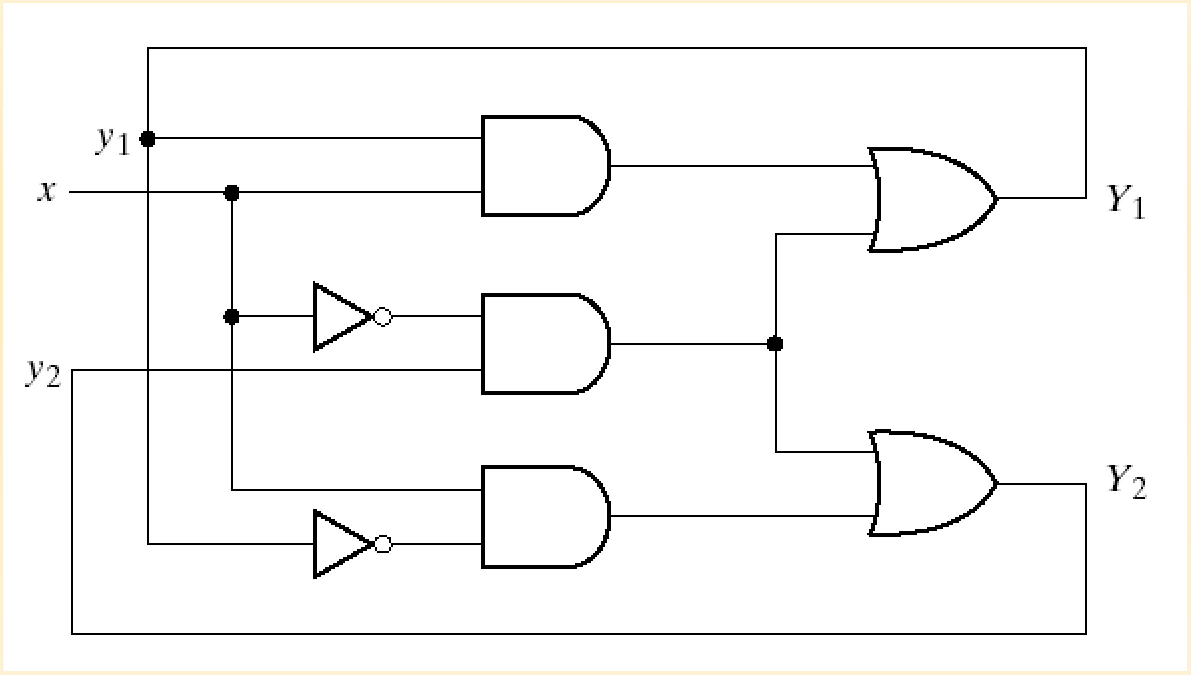
Introduction

- fundamental mode

fundamental mode :Only one input variable can change at any one time and the time between two input changes must be longer than the time it takes the circuit to reach a stable state.

Analysis Procedure

Transition Table



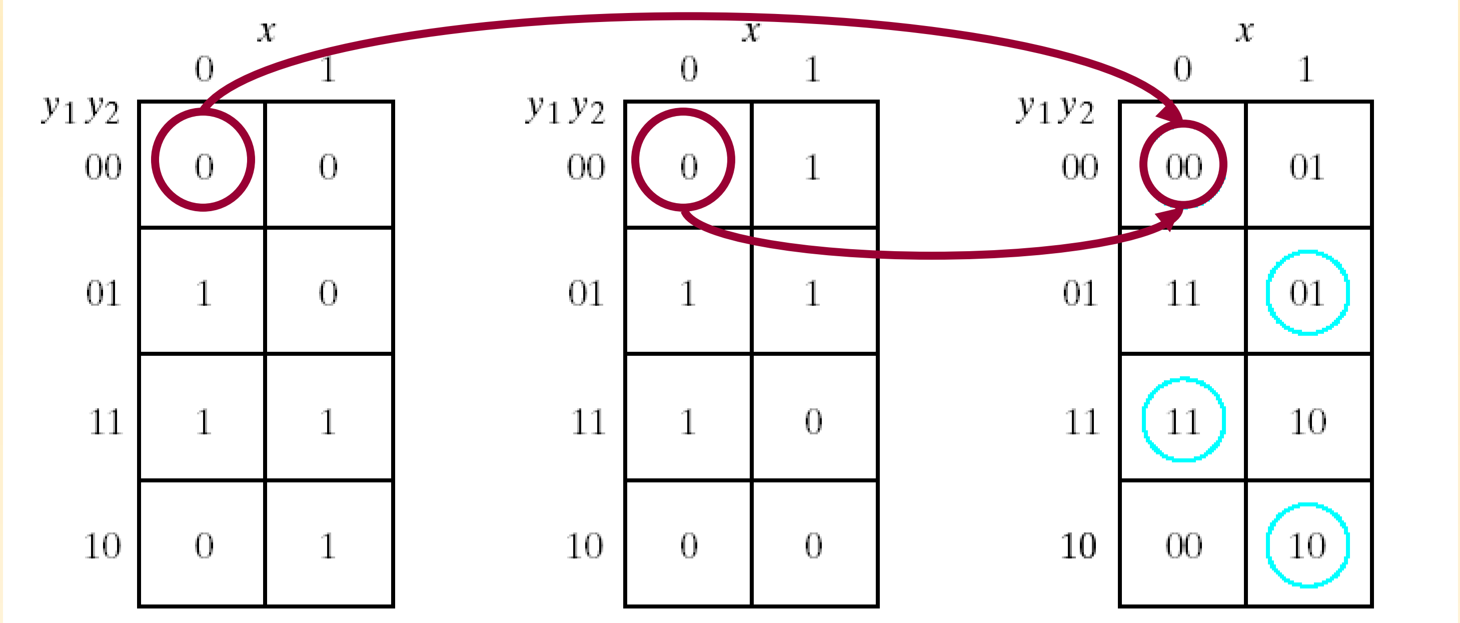
$$Y_1 = xy_1 + x'y_2$$

$$Y_2 = xy'_1 + x'y_2$$

Analysis Procedure

Transition Table

$$Y_1 = xy_1 + x'y_2 \qquad\qquad Y_2 = xy'_1 + x'y_2$$



y_1y_2	$x=0$	$x=1$
00	0	0
01	1	0
11	1	1
10	0	1

(a) Map for $Y_1 = xy_1 + x'y_2$

y_1y_2	$x=0$	$x=1$
00	0	1
01	1	1
11	1	0
10	0	0

(b) Map for $Y_2 = xy'_1 + x'y_2$

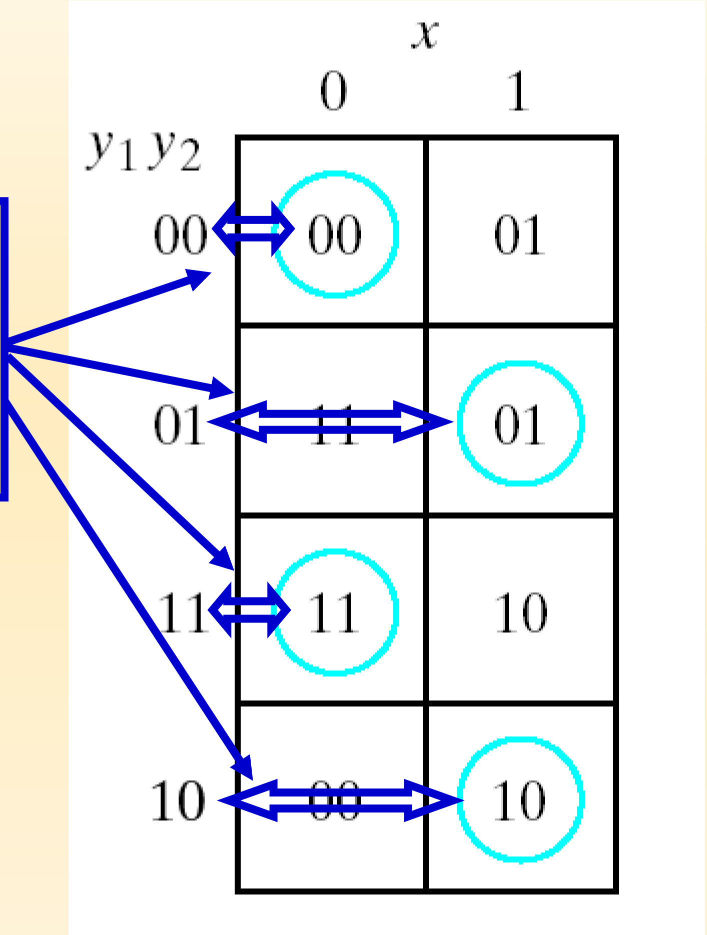
y_1y_2	$x=0$	$x=1$
00	00	01
01	11	01
11	11	10
10	00	10

(c) Transition table

Analysis Procedure

Transition Table

For a state to be stable, the value of Y must be the same as that of $y = y_1y_2$



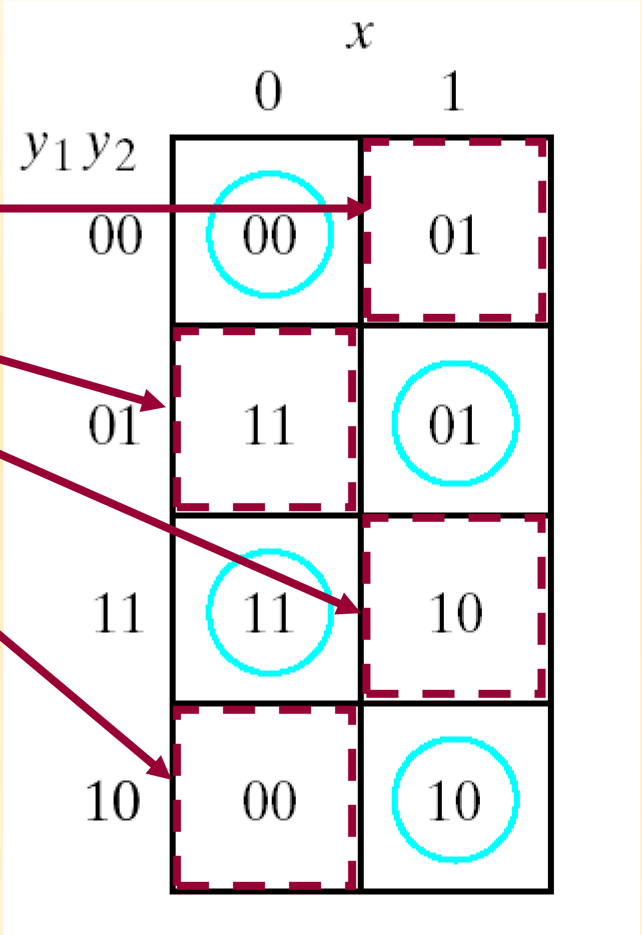
y_1y_2 \ x	0	1
00	00	01
01	11	01
11	11	10
10	00	10

(c) Transition table

Analysis Procedure

Transition Table

The Unstable states,
$Y \neq y$



$y_1 y_2$ \ x	0	1
00	00	01
01	11	01
11	11	10
10	00	10

(c) Transition table

Analysis Procedure

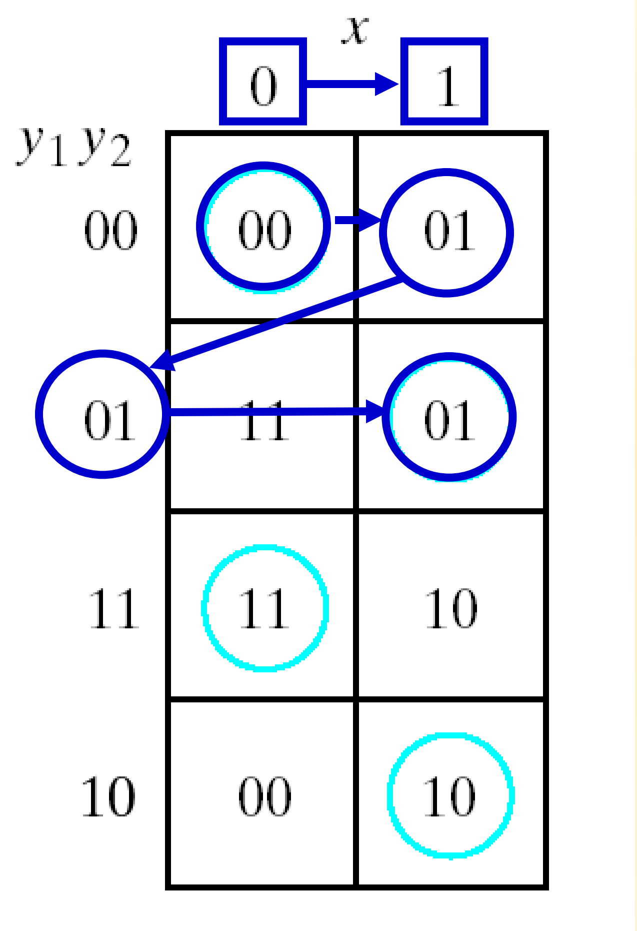
Transition Table

Consider the square for $x = 0$ and $y = 00$. It is stable.

x changes from 0 to 1.

The circuit changes the value of Y to 01. The state is unstable.

The feedback causes a change in y to 01. The circuit reaches stable.



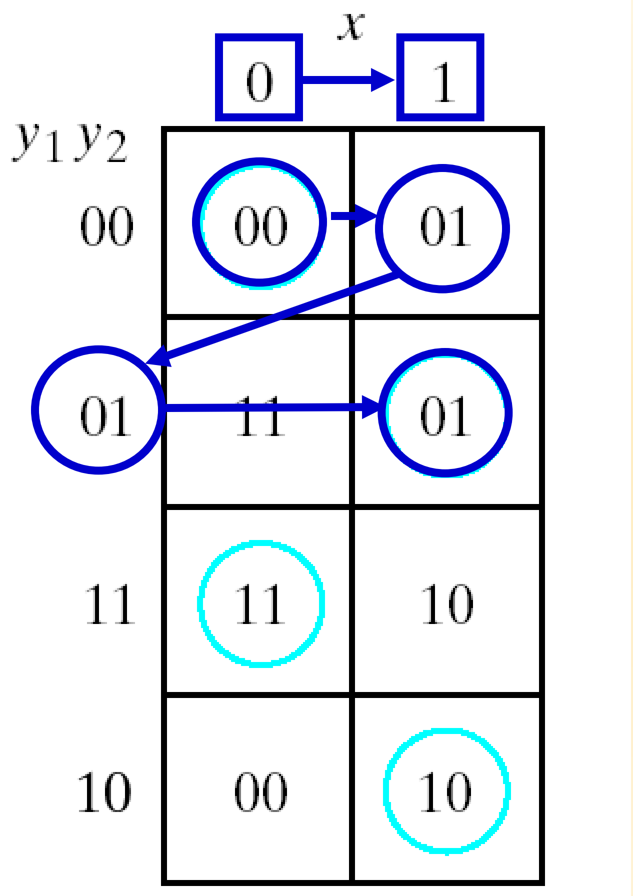
y_1y_2 \ x	0	1
00	00	01
01	11	01
11	11	10
10	00	10

(c) Transition table

Analysis Procedure

Transition Table

In general, if a change in the input takes the circuit to an unstable state, *y* will change until it reaches a stable state.



	x	
y_1y_2	0	1
00	00	01
01	11	01
11	11	10
10	00	10

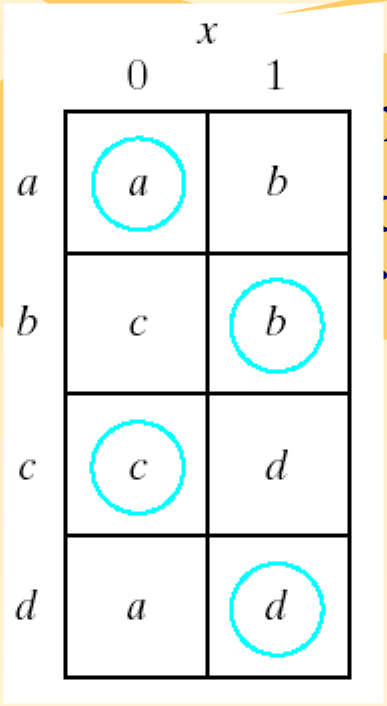
(c) Transition table

Analysis Procedure

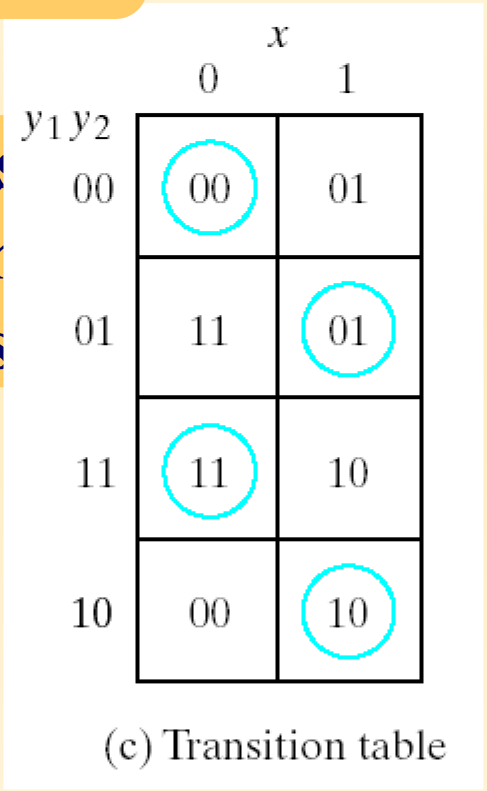
Flow Table

Flow Table

table whose
by letter sym
binary values



	x = 0	1
a	a	b
b	c	b
c	c	d
d	a	d



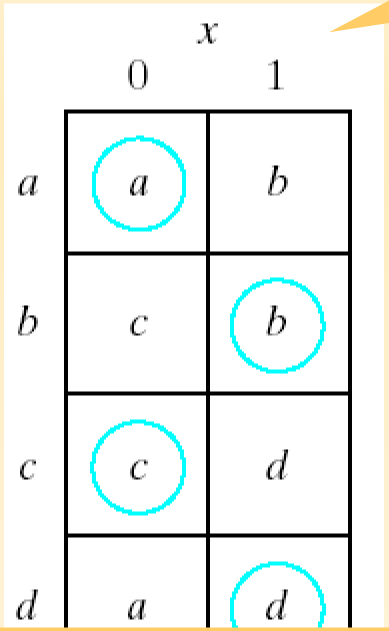
y_1y_2	x = 0	1
00	00	01
01	11	01
11	11	10
10	00	10

(c) Transition table

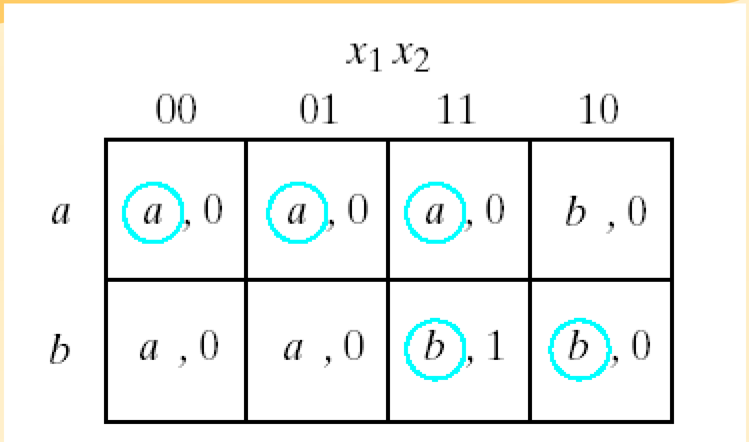
Analysis Procedure

Flow Table

It is called primitive flow table because it has only one stable state in each row.



	x = 0	1
a	a	b
b	c	b
c	c	d
d	a	d



	x_1x_2 = 00	01	11	10
a	a, 0	a, 0	a, 0	b, 0
b	a, 0	a, 0	b, 1	b, 0

(b) Two states with two inputs and one output

It is a flow table with more than one stable state in the same row.

Analysis Procedure

Flow Table

y_1y_2 \ x	0	1
00	00	01
01	11	01
11	11	10
10	00	10

(c) Transition table

y \ x_1x_2	00	01	11	10
0	0	0	0	1
1	0	0	1	1

(a) Transition table

$Y = x_1x'_2 + x_1y$

y \ x_1x_2	00	01	11	10
0	0	0	0	0
1	0	0	1	0

(b) Map for output

$z = x_1x_2y$

Analysis Procedure

y \ x_1x_2	00	01	11	10
0	0	0	0	1
1	0	0	1	1

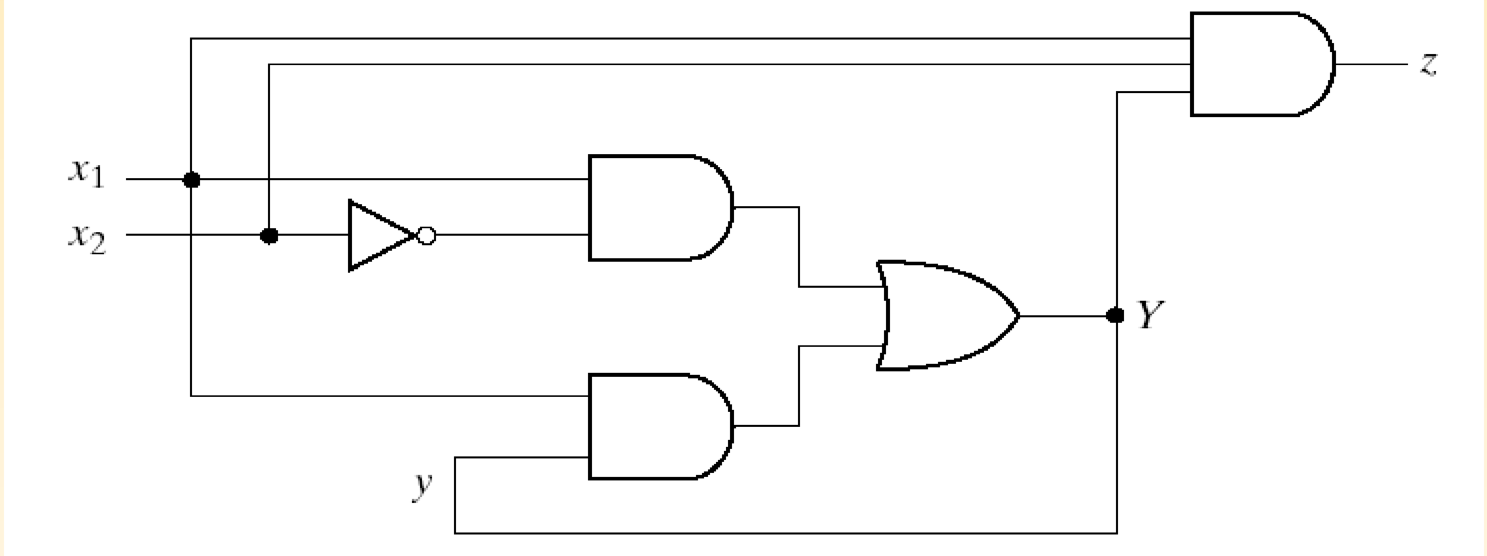
(a) Transition table

$Y = x_1x'_2 + x_1y$

y \ x_1x_2	00	01	11	10
0	0	0	0	0
1	0	0	1	0

(b) Map for output

$z = x_1x_2y$



Analysis Procedure

Race Conditions

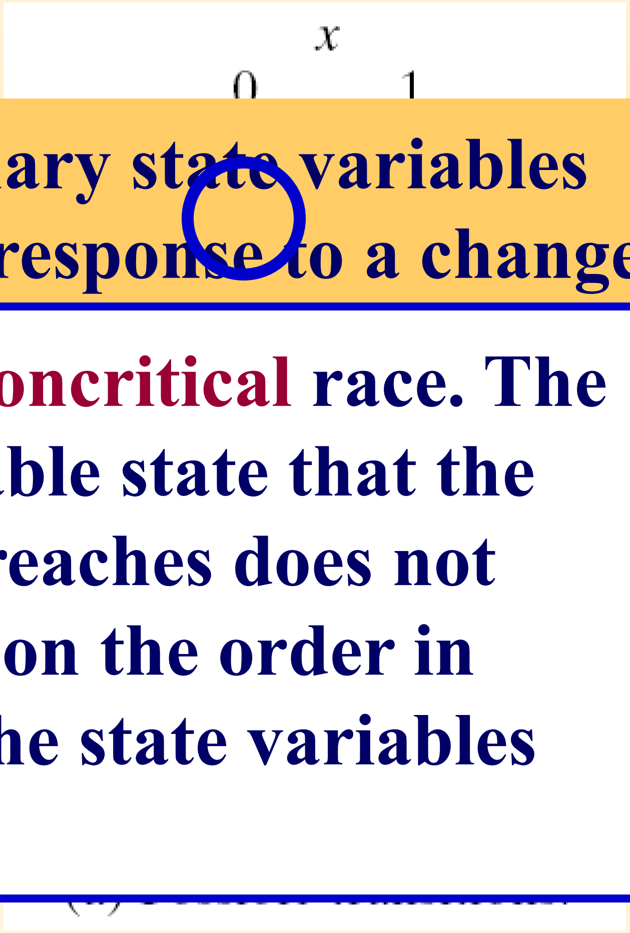
Noncritical Race:

State variables change from 00 to 11. The possible
could be

Two or more binary state variables change value in response to a change in an inp

00	11	
00	01	11
00	10	11

It is a noncritical race. The final stable state that the circuit reaches does not depend on the order in which the state variables change.



Analysis Procedure

Race Conditions

Critical Race:

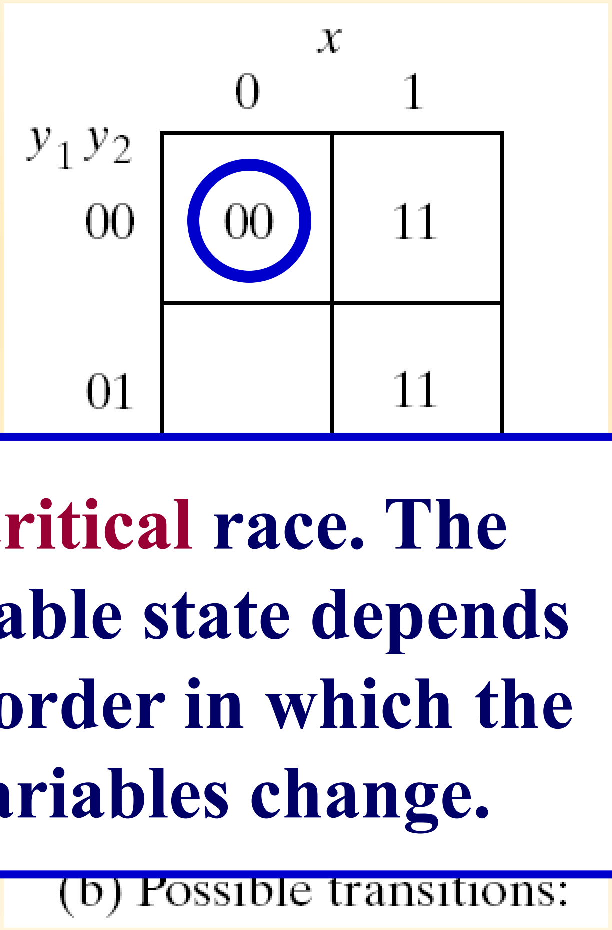
State variables change from 00 to 11. The possible transition could be

00 11

00 01 11

00 10

It is a critical race. The final stable state depends on the order in which the state variables change.



(b) Possible transitions:

Analysis Procedure

Race Conditions

Cycle

It st 00

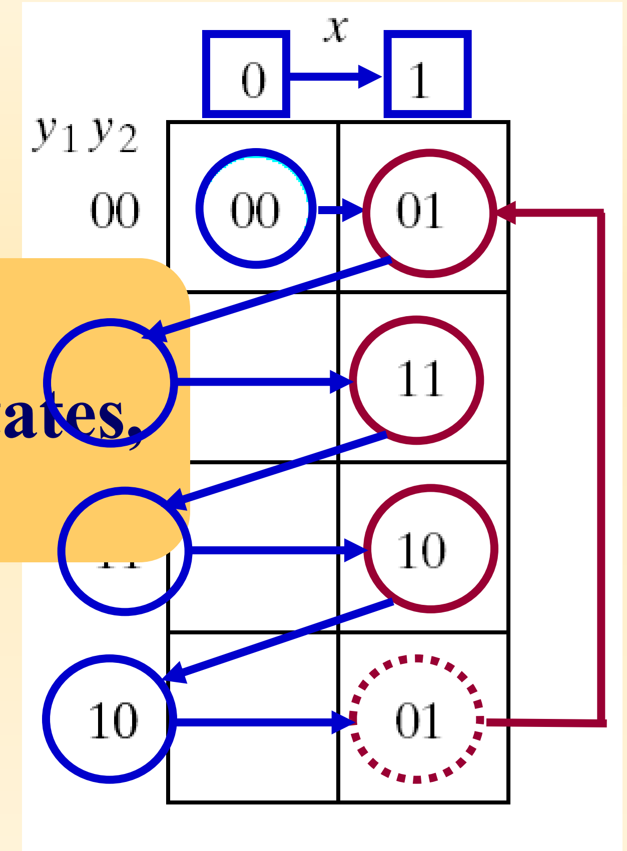
the

0 t

When a circuit goes through a unique sequence of unstable states, it is said to have a *cycle.*

The sequence is as follows,

$$\begin{tikzcd} 00 & 01 & 11 \arrow[ll, bend left] \end{tikzcd}$$



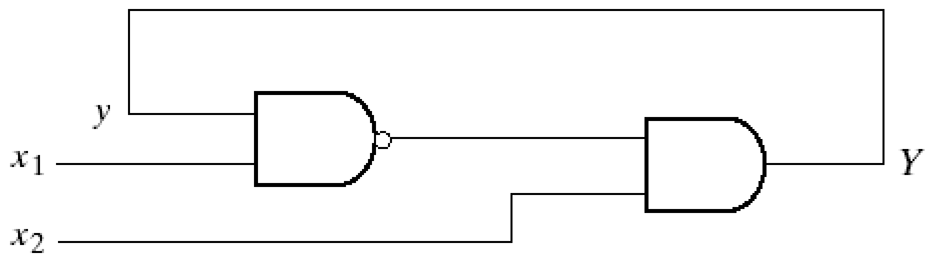
(c) Unstable

Analysis Procedure

Stability Consideration

Column 11 has no stable state. With input $x_1\, x_2 = 11$, Y and y are never the same.

This will cause instability.



(a) Logic diagram

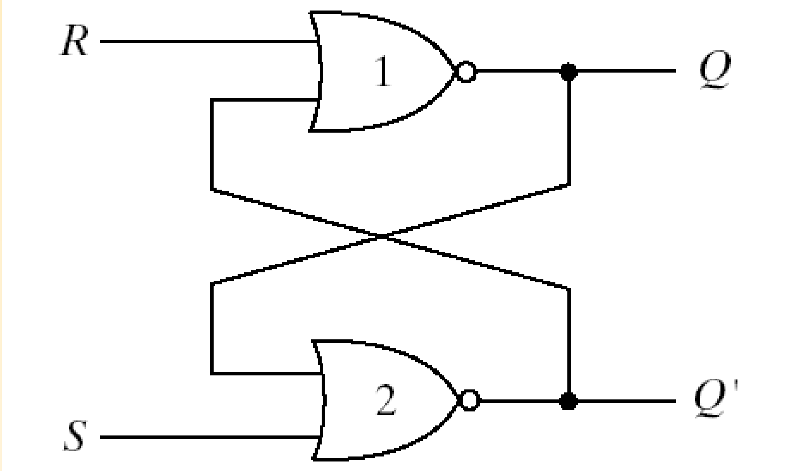
y \ $x_1 x_2$	00	01	11	10
0	0	1	1	0
1	0	1	0	0

(b) Transition table

Circuits with Latches

SR Latch

The circuit diagram and truth table of the *SR* latch are shown as follows,



(a) Crossed-coupled circuit

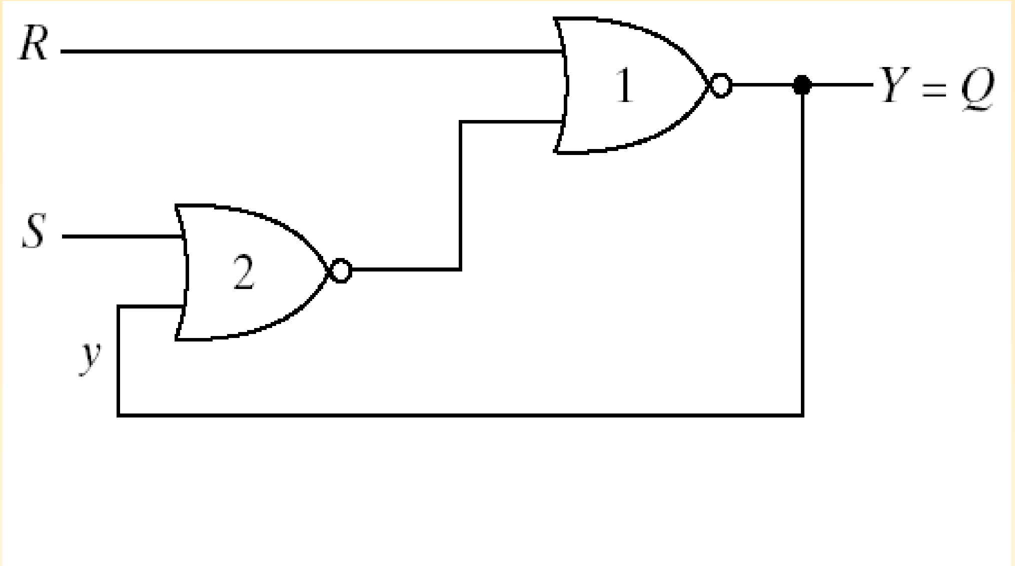
S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(After SR = 10)
0	1	0	1	
0	0	0	1	(After SR = 01)
1	1	0	0	

(b) Truth table

Circuits with Latches

SR Latch

The circuit diagram of the *SR* latch can be redrawn as follows,



(c) Circuit showing feedback

y	00	01	11	10
0	0	0	0	1
1	1	0	0	1

$Y = SR' + R'y$

$Y = S + R'y$ when $SR = 0$

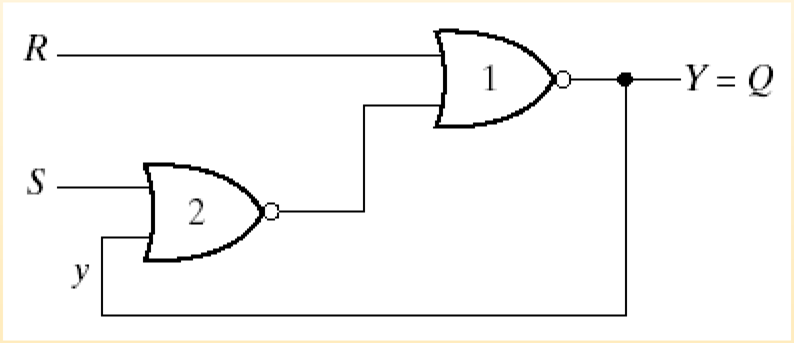
(d) Transition table

Circuits with Latches

SR Latch

$$Y = [(S + y)' + R]'$$

$$= (S + y)R' = SR' + R'y$$



$$\left.\begin{array}{l} SR' + SR = S\,(\,R' + R\,) = S \\ SR = 0 \end{array}\right\} \quad SR' = S$$

$$\Rightarrow \quad Y = SR' + R'y = S + R'y \quad \text{when } SR=0$$

Circuits with Latches

Analysis Example

$S_1 = x_1 y_2$

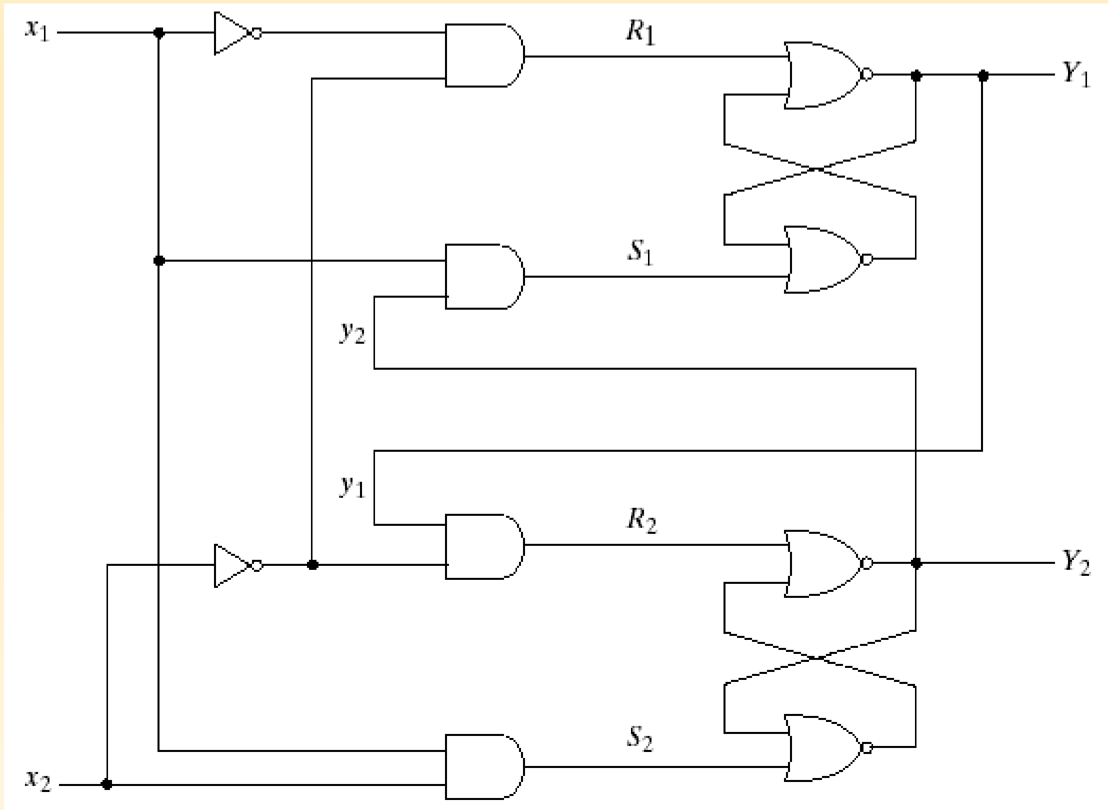
$S_2 = x_1 x_2$

$R_1 = x'_1 x'_2$

$R_2 = x'_2 y_1$

$S_1 R_1 = x_1 y_2 x'_1 x'_2 = 0$

$S_2 R_2 = x_1 x_2 x'_2 y_1 = 0$



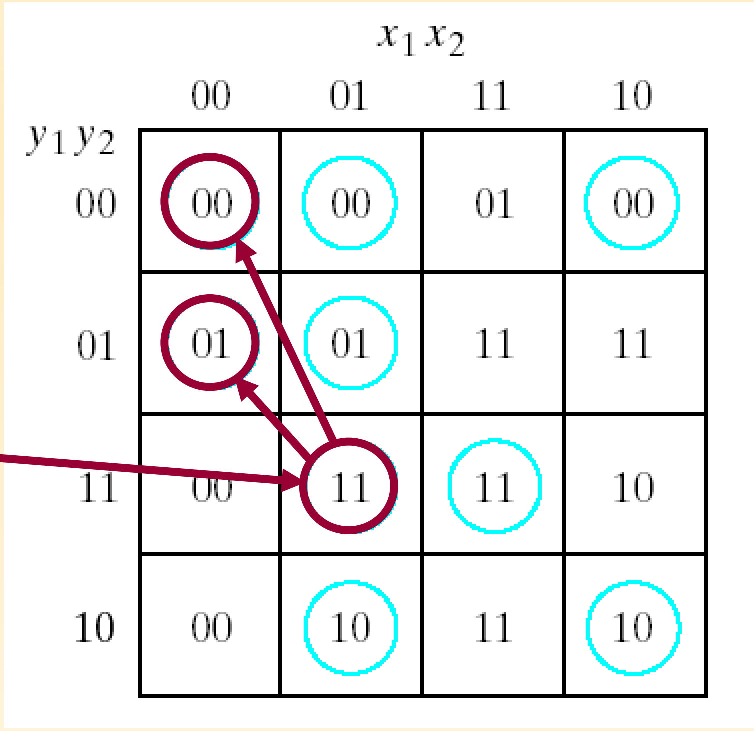
Circuits with Latches

Analysis Example

$$Y_1 = S_1 + R'_1 y_1 = x_1 y_2 + (x_1 + x_2) y_1$$

$$Y_2 = S_2 + R'_2 y_2 = x_1 x_2 + (x_2 + y'_1) y_2$$

There is a critical race condition



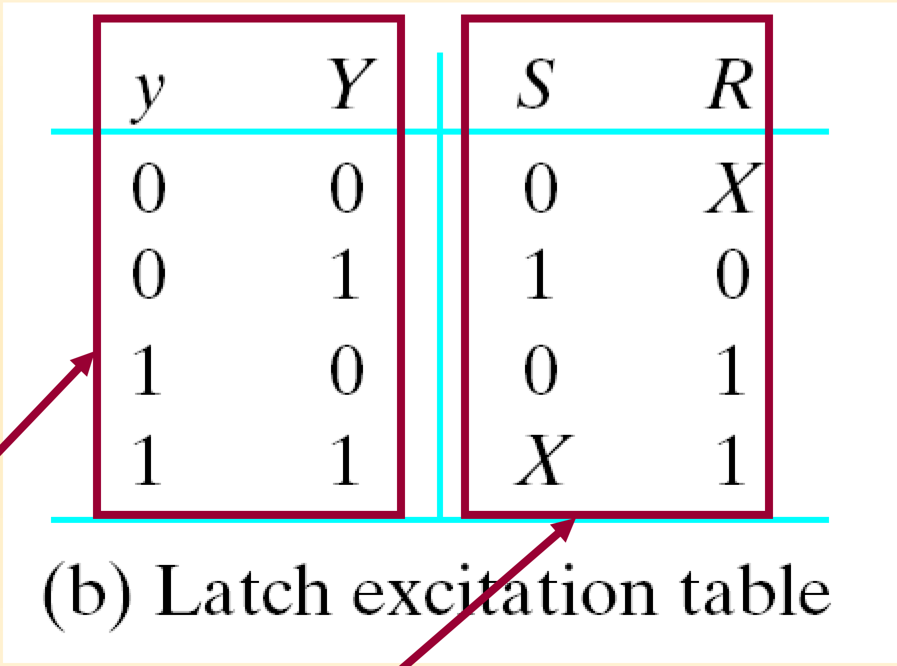
$y_1 y_2$ \ $x_1 x_2$	00	01	11	10
00	00	00	01	00
01	01	01	11	11
11	00	11	11	10
10	00	10	11	10

Circuits with Latches

Latch Excitation Table

A table that lists the required inputs *S* and R for each of the possible transitions from *y* to *Y*

The first two columns list the four possible transitions from *y* to *Y*.



y	Y	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	1

(b) Latch excitation table

The next two columns specify the required input values that will result in the specified transition.

Circuits with Latches

Implementation Example

y \ x_1x_2	00	01	11	10
0	0	0	0	1
1	0	0	1	1

(a) Transition table
$Y = x_1x'_2 + x_1y$

y	Y	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

(b) Latch excitation table

y \ x_1x_2	00	01	11	10
0	0	0	0	1
1	0	0	X	X

(c) Map for $S = x_1x'_2$

y \ x_1x_2	00	01	11	10
0	X	X	X	0
1	1	1	0	0

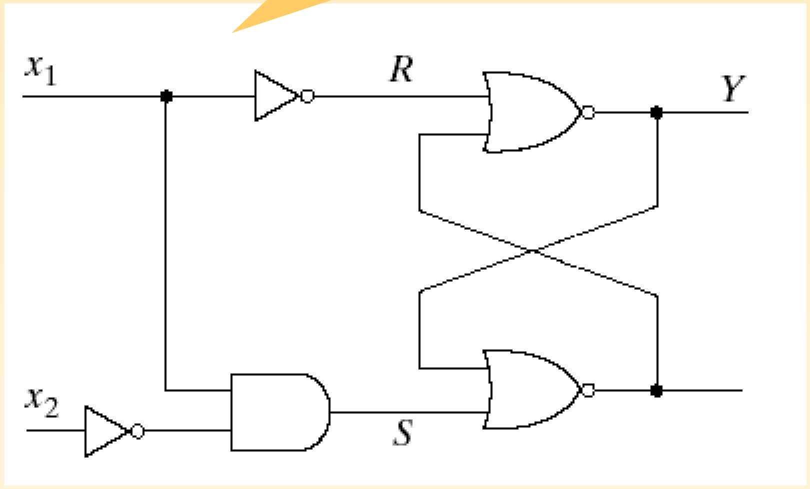
(d) Map for $R = x'_1$

Circuits with Latches

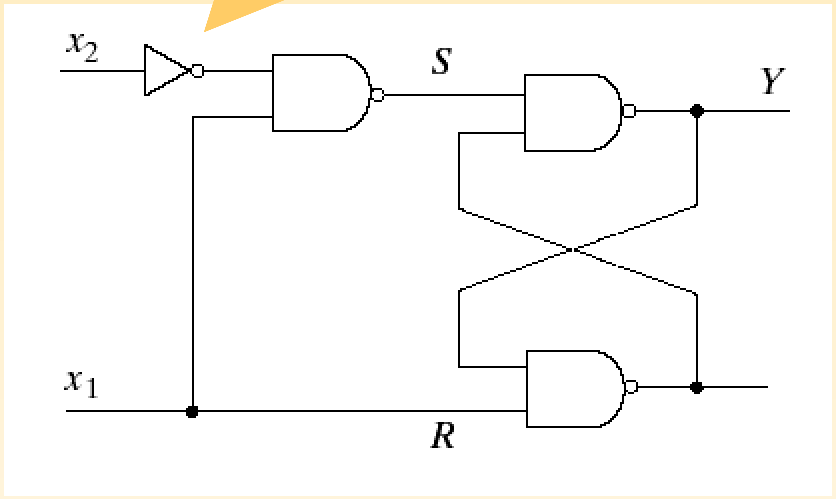
Implementation Example

$S = x_1 x'_2 \qquad R = x'_1$

Circuit with NOR latch



Circuit with NAND latch



Design Procedure

Design Example

Design a gated latch circuit with t

Gated-Latch
Total States

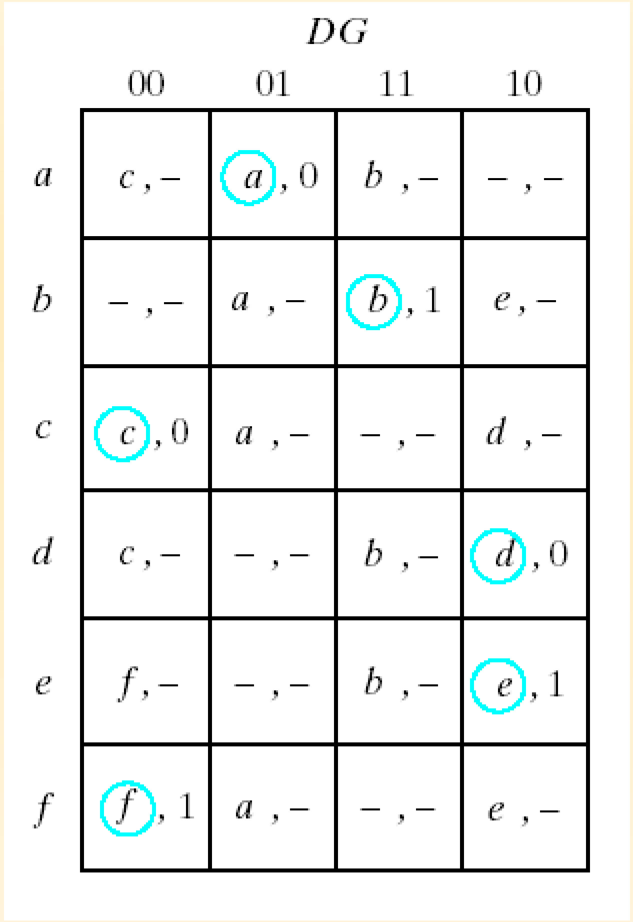
te) and *D* (data), and one output *Q*.

State	Inputs		Output	comments
	D	G	Q	
a	0	1	0	$D = Q$ because $G = 1$
b	1	1	1	$D = Q$ because $G = 1$
c	0	0	0	After state a or d
d	1	0	0	After state c
e	1	0	1	After state b or f
f	0	0	1	After state e

Design Procedure

Design Example

State	Inputs D	Inputs G	Output Q
a	0	1	0
b	1	1	1
c	0	0	0
d	1	0	0
e	1	0	1
f	0	0	1



	DG 00	01	11	10
a	c, –	$\boxed{a}$, 0	b, –	–, –
b	–, –	a, –	$\boxed{b}$, 1	e, –
c	$\boxed{c}$, 0	a, –	–, –	d, –
d	c, –	–, –	b, –	$\boxed{d}$, 0
e	f, –	–, –	b, –	$\boxed{e}$, 1
f	$\boxed{f}$, 1	a, –	–, –	e, –

D

Two of more rows in the primitive flow table can be merged into one row if there are non-conflicting states and outputs in each of the columns.

Reduction
Primitive Flo able

	DG 00	01	11	10
a	c , –	$\textcircled{a}$, 0	b , –	– , –
b	– , –	a , –	$\textcircled{b}$, 1	e , –
c	$\textcircled{c}$, 0	a , –	– , –	d , –
d	c , –	– , –	b , –	$\textcircled{d}$, 0
e	f , –	– , –	b , –	$\textcircled{e}$, 1
f	$\textcircled{f}$, 1	a , –	– , –	e , –

$\Rightarrow$

	DG 00	01	11	10
a	c , –	$\textcircled{a}$, 0	b , –	– , –
c	$\textcircled{c}$, 0	a , –	– , –	d , –
d	c , –	– , –	b , –	$\textcircled{d}$, 0

	DG 00	01	11	10
b	– , –	a , –	$\textcircled{b}$, 1	e , –
e	f , –	– , –	b , –	$\textcircled{e}$, 1
f	$\textcircled{f}$, 1	a , –	– , –	e , –

Design Procedure

Reduction of the Primitive Flow Table

	DG 00	01	11	10
a	*c* , –	ⓐ, 0	*b* , –	– , –
c	ⓒ ,0	*a* , –	– , –	*d* , –
d	*c* , –	– , –	*b* , –	ⓓ, 0

	DG 00	01	11	10
b	– , –	*a* , –	ⓑ, 1	*e* , –
e	*f* , –	– , –	*b* , –	ⓔ, 1
f	ⓕ, 1	*a* , –	– , –	*e* , –

	DG 00	01	11	10
a, c, d	ⓒ, 0	ⓐ, 0	*b* , –	ⓓ, 0
b, e, f	ⓕ, 1	*a* , –	ⓑ, 1	ⓔ, 1

	DG 00	01	11	10
a	ⓐ, 0	ⓐ, 0	*b* , –	ⓐ, 0
b	ⓑ, 1	*a* , –	ⓑ, 1	ⓑ, 1

Design Procedure

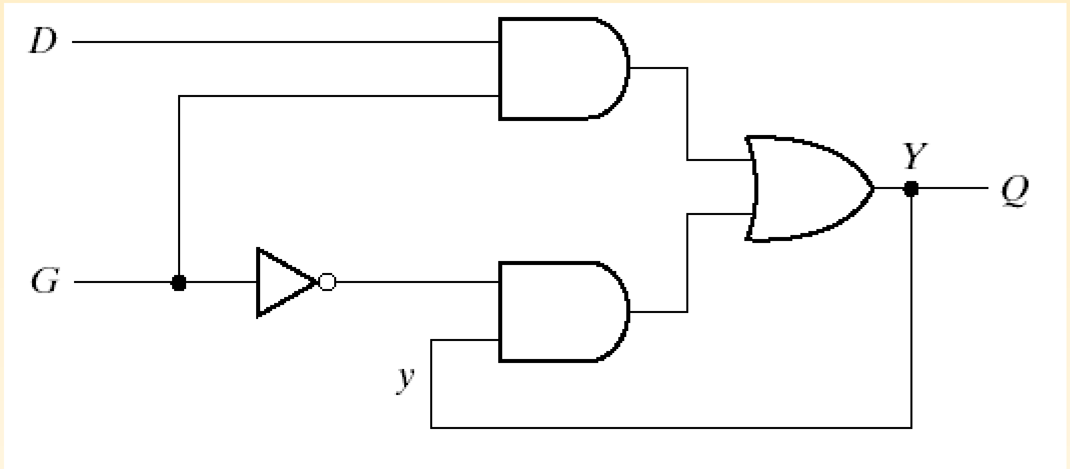
Transition Table and Logic Diagram

	DG			
y	00	01	11	10
0	0	0	1	0
1	1	0	1	1

(a) $Y = DG + G'y$

	DG			
y	00	01	11	10
0	0	0	1	0
1	1	0	1	1

(b) $Q = Y$



Design Procedure

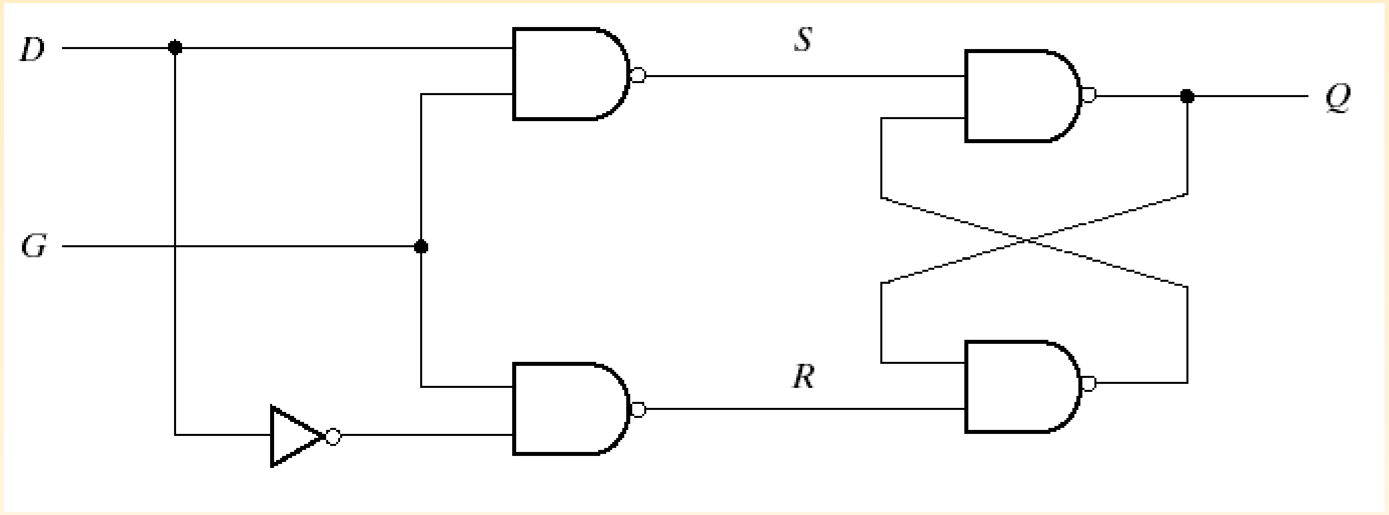
Circuit With *SR* Latch

y \ DG	00	01	11	10
0	0	0	1	0
1	X	0	X	X

(a) $S = DG$

y \ DG	00	01	11	10
0	X	X	0	X
1	0	1	0	0

$R = D'G$



Design Procedure

Assigning Output to Unstable States

1. Assign a 0 to an output variable associated with an unstable state that is a transient state between two stable states that have a 0 in the corresponding output variable.
2. Assign a 1 to an output variable associated with an unstable state that is a transient state between two stable states that have a 1 in the corresponding output variable.
3. Assign a don't-care condition to an output variable associated with an unstable state that is a transient state between two stable states that have different values in the corresponding output variable.

Reduction of State and Flow Tables

Equivalent States

Two states are equivalent if for each possible input, they give exactly the same output and go to the same next states or to equivalent next states.

The characteristic of equivalent states is that if (a,b) imply (c,d) and (c,d) imply (a,b), then both pairs of states are equivalent.

Reduction of State and Flow Tables

Implication Table

Two states are equivalent if for each possible input, they give exactly the same output and go to the same next states or to equivalent next states.

The characteristic of equivalent states is that if (*a*,*b*) imply (*c*,*d*) and (*c*,*d*) imply (*a*,*b*), then both pairs of states are equivalent.

Reduction of State and Flow Tables

Implication

Without the first

Example

S_n \ X_1X_2	00	01	11	10
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

S_{n+1}/Z_n

Without the last

B	×						
C	×	AF					
D	BD AF	×	×				
E	×	DF AF	DF	×			
F	√	×	×	BD	×		
G	DG AF	×	×	BG AF	×	DG AF	
H	×	BC AF	BC	×	BC DF	×	×
	A	B	C	D	E	F	G

Reduction of State and Flow Tables

Implied Pairs

example

X_1X_2 / S_n	00	01	11	10
A	D/0	D/0	F/0	A/0
B	C/1	D/0	E/1	F/0
C	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	F/0
E	C/1	F/0	E/1	A/0
F	D/0	D/0	A/0	F/0
G	G/0	G/0	A/0	A/0
H	B/1	D/0	E/1	A/0

S_{n+1}/Z_n

	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	BD AF ×	×	×				
E	×	DF AF ×	DF ×	×			
F	√	×	×	BD ×	×		
G	DG AF ×	×	×	BG AF ×	×	DG AF ×	
H	×	BC AF	BC	×	BC DF ×	×	×

Reduction of State and Flow Tables

The equivalent states: [A, F]、[B, H]、[B, C]、[C, H]。

Combined into [B,C,H]

Denoted by A

Denoted by B

rou

e ivalent state

[A, F], [B, C, H],
[D], [E], [G]

S_n \ X_1X_2	00	01	11	10
A	D/0	D/0	A/0	A/0
B	C/1	D/0	E/1	A/0
D	D/0	B/0	A/0	A/0
E	B/1	A/0	E/1	A/0
G	G/0	G/0	A/0	A/0

S_{n+1}/Z_n

	A	B	C	D	E	F	G
B	×						
C	×	AF					
D	BD AF ×	×	×				
E	×	DF AF ×	DF ×	×			
F	√	×	×	BD ×	×		
G	DG AF ×	×	×	BG AF ×	×	DG AF ×	
H	×	BC AF	BC	×	BC DF ×	×	×

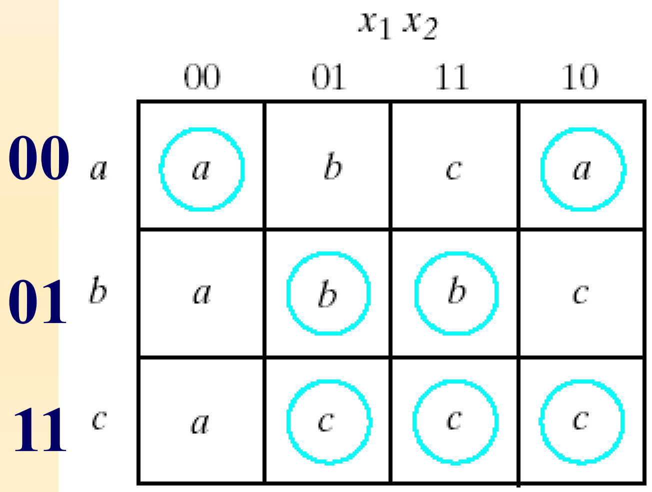
Race-Free State Assignment

The primary objective in choosing a proper binary state assignment is the prevention of critical races.

Critical races can be avoided by making a binary state assignment in such a way that only one variable changes at any given time when a state transition occurs in the flow table.

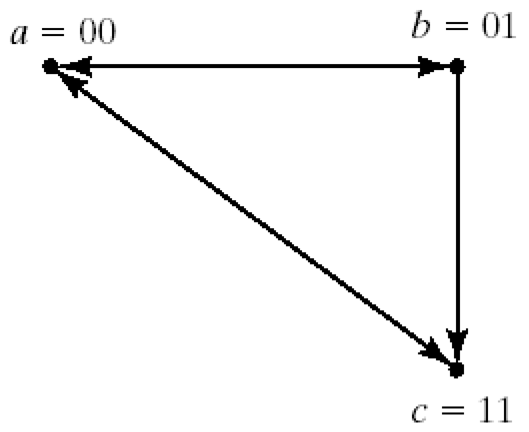
Race-Free State Assignment

Three-Row Flow-Table Example



		$x_1 x_2$ 00	01	11	10
00	a	ⓐ	b	c	ⓐ
01	b	a	ⓑ	ⓑ	c
11	c	a	ⓒ	ⓒ	ⓒ

(a) Flow table

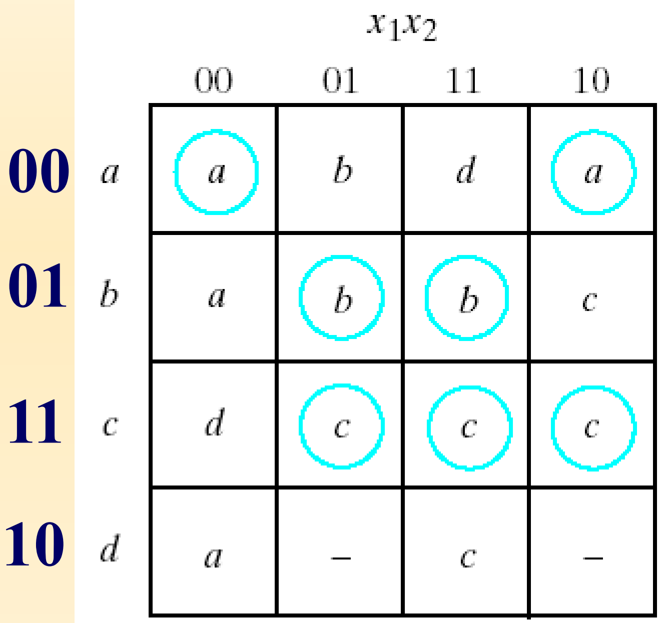


(b) Transition diagram

This assignment will cause a critical race during the transition from *a* to *c*.

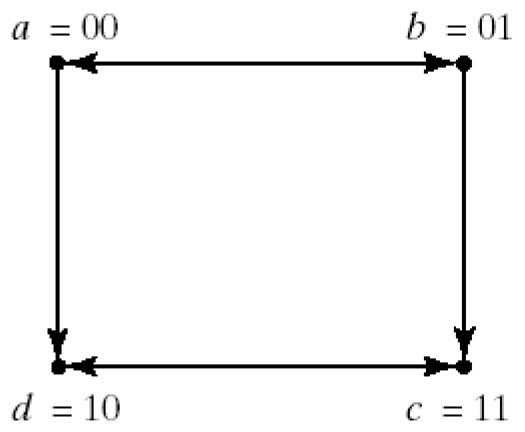
Race-Free State Assignment

Three-Row Flow-Table Example



		x_1x_2 00	01	11	10
00	*a*	*a*	*b*	*d*	*a*
01	*b*	*a*	*b*	*b*	*c*
11	*c*	*d*	*c*	*c*	*c*
10	*d*	*a*	–	*c*	–

(a) Flow table



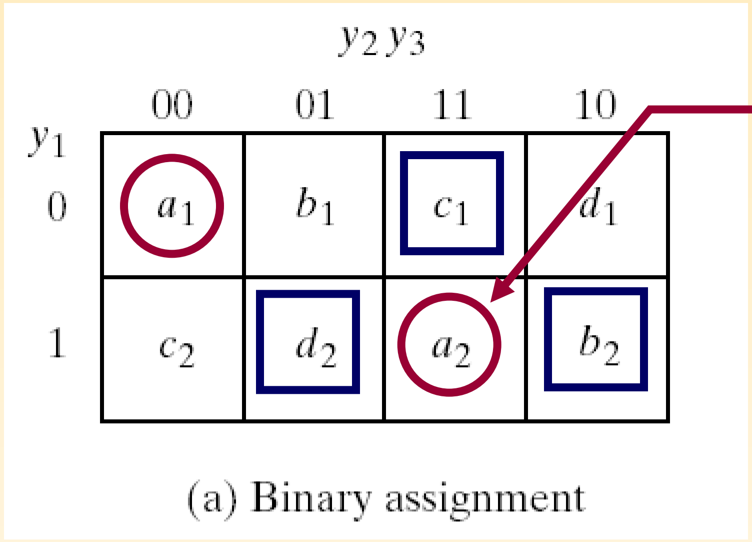
$$\begin{tikzcd}
a = 00 \arrow[r] \arrow[d] & b = 01 \arrow[l] \arrow[d] \\
d = 10 \arrow[r] & c = 11 \arrow[l]
\end{tikzcd}$$

The transition from ***a*** to ***c*** must now go through ***d***, thus avoiding a critical race.

Race-Free State Assignment

Multiple-Row Method

In the multiple-row assignment, each state in the original flow table is replaced by two or more combinations of state variables.



(a) Binary assignment

Note that a2 is adjacent to d2, c1, b2.

Race-Free State Assignment

Multiple-Row Method

	00	01	11	10
a	b	a	d	a
b	b	d	b	a
c	c	a	b	c
d	c	d	d	c

(a) Flow table

The original flow table

$\Rightarrow$

	00	01	11	10
$000 = a_1$	b_1	a_1	d_1	a_1
$111 = a_2$	b_2	a_2	d_2	a_2
$001 = b_1$	b_1	d_2	b_1	a_1
$110 = b_2$	b_2	d_1	b_2	a_2
$011 = c_1$	c_1	a_2	b_1	c_1
$100 = c_2$	c_2	a_1	b_2	c_2
$010 = d_1$	c_1	d_1	d_1	c_1
$101 = d_2$	c_2	d_2	d_2	c_2

(b) Flow table

Hazards

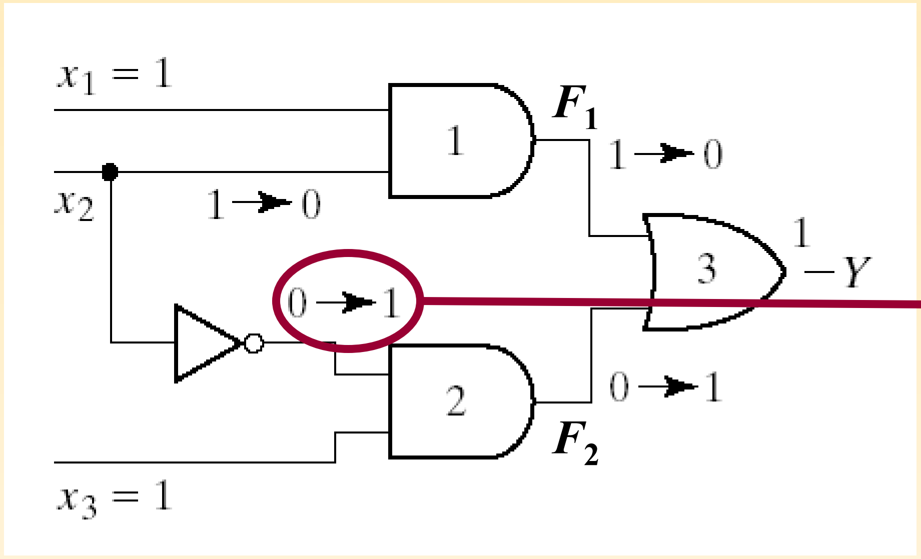
Hazards are unwanted switching transients that may appear at the output of a circuit because different paths exhibit different propagation delays.

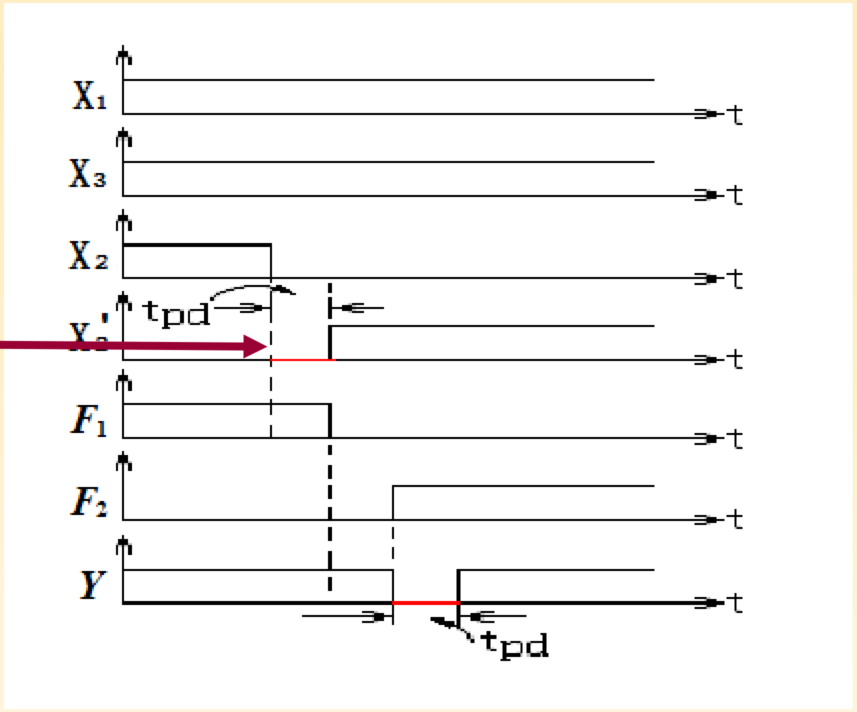
Hazards occur in combinational circuits, where they may cause a temporary false-output value.

When hazards occur in sequential circuits, it may result in a transition to a wrong stable state.

Hazards

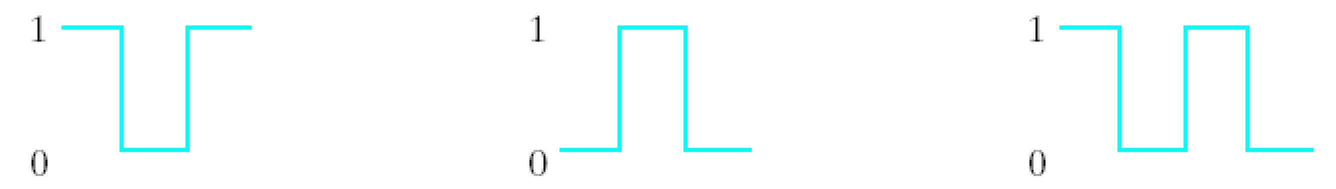
Hazards in Combinational Circuits





Hazards

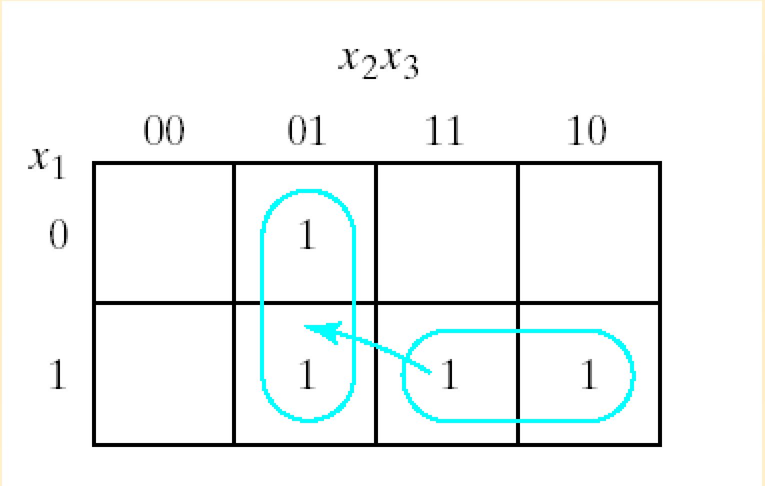
Hazards in Combinational Circuits



(a) Static 1-hazard (b) Static 0-hazard (c) Dynamic hazard

Hazards

Hazards in Combinational Circuits

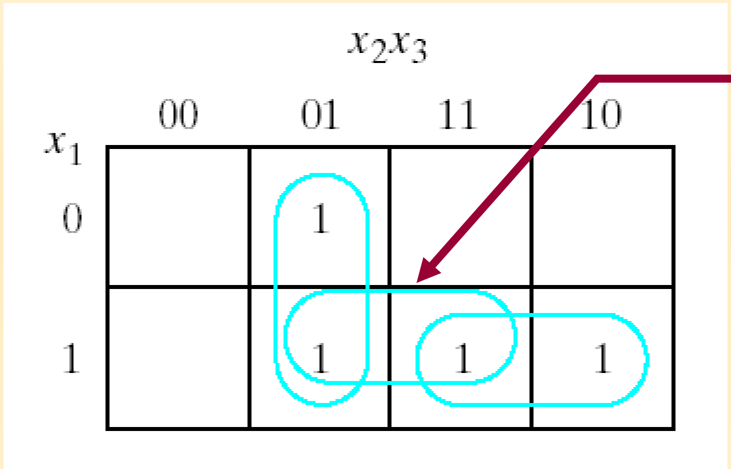


(a) $Y = x_1 x_2 + x'_2 x_3$

The hazard exists because the change of input results in a different product term covering the two minterms.

Hazards

Hazards in Combinational Circuits



(b) $Y = x_1x_2 + x'_2x_3 + x_1x_3$

The remedy for eliminating a hazard is to enclose the two minterms in question with another product term that overlap both grouping.

Hazards

Hazards in Combinational Circuits

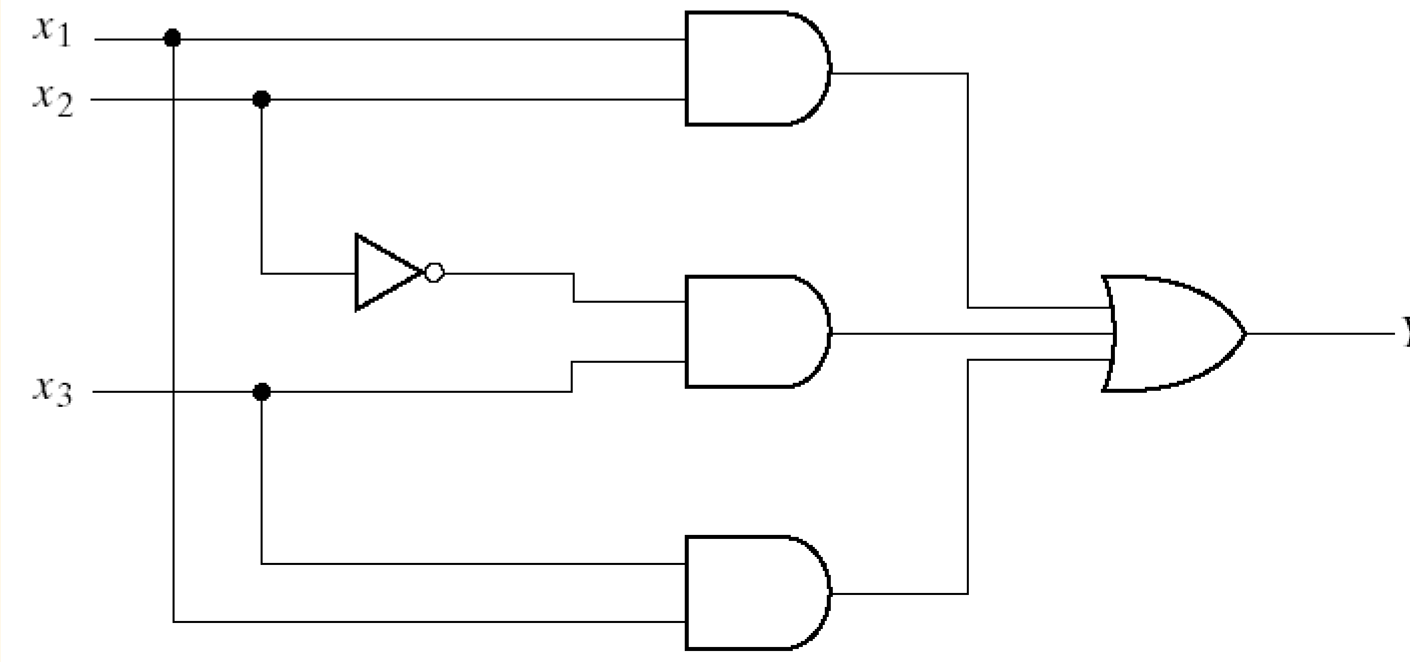


Fig. 9-36 Hazard-Free Circuit

Hazards

Implementation with SR Latches

Consider a NAND *SR*

Boolean functions fo

Since this is a NAND latch, we apply the complemented values to the inputs:

$S = AB + CD$

$R = A'C$

$S = (AB + CD)' = (AB)'\,(CD)'$

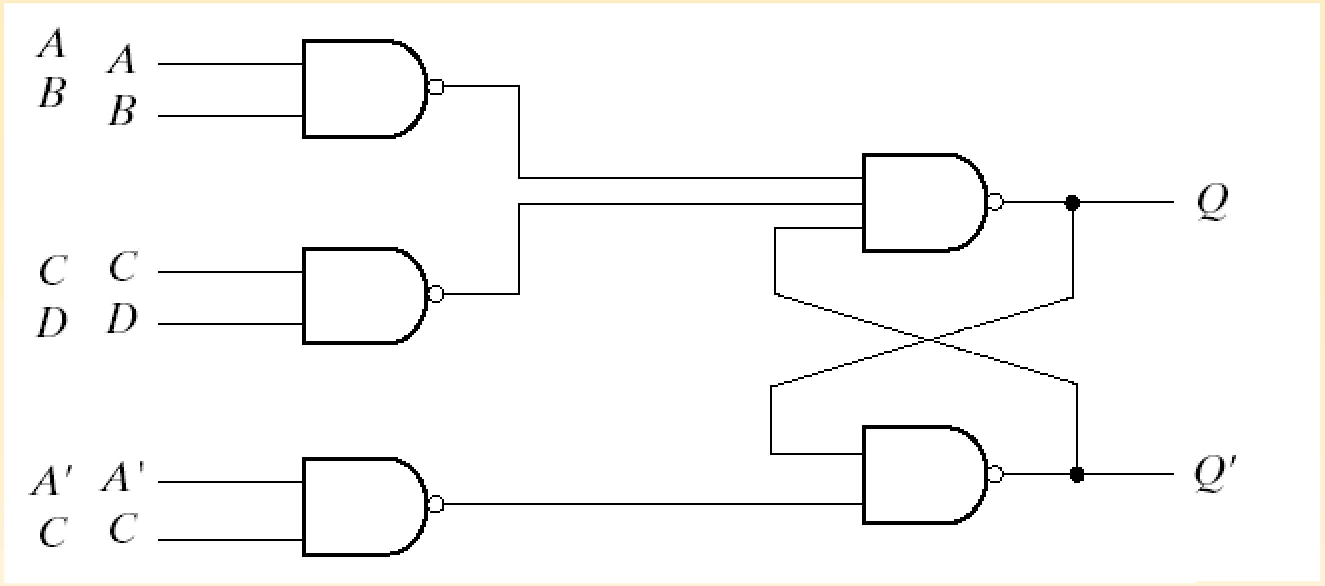
$R = (A'C)'$

$Q = (Q'S)' = [Q'\,(AB)'\,(CD)'\,]'$

Hazards

Implementation with SR Latches

$$Q = (Q'S)' = [Q'(AB)'(CD)']'$$



Design Example

The Recommended Procedure

1. State the design specifications
2. Derive a primitive flow table
3. Reduce the flow table by merging the rows
4. Make a race-free binary state assignment
5. Obtain the transition table and output map
6. Obtain the logic diagram using *SR* latch

Design Example

Design Specifications

It is necessary to design a negative-edge-triggered flip-flop. The circuit has two inputs, *T* (toggle) and *C* (clock), and one output, *Q*.

Design Example

Primitive Flow Table

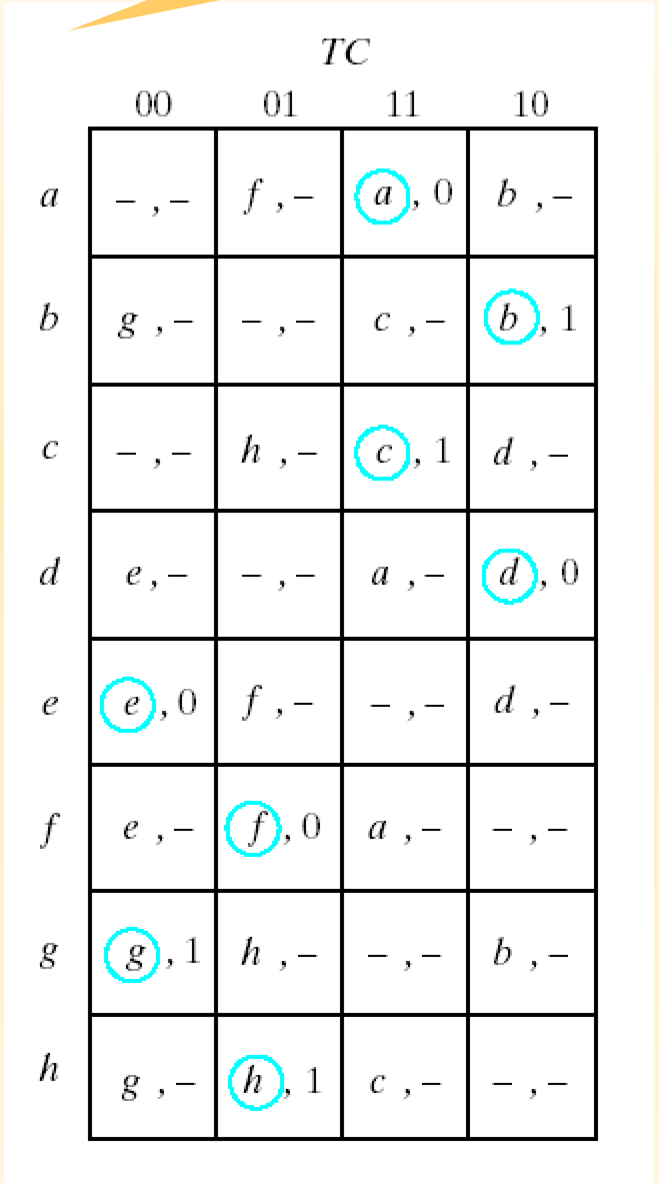
Specification of Total States

State	Inputs		Output	Comments
	T	C	Q	
a	1	1	0	Initial input is 0
b	1	0	1	After state a
c	1	1	1	Initial input is 1
d	1	0	0	After state c
e	0	0	0	After state d or f
f	0	1	0	After state e or a
g	0	0	1	After state b or h
h	0	1	1	After state g or c

Primitive Flow Table

Design Example

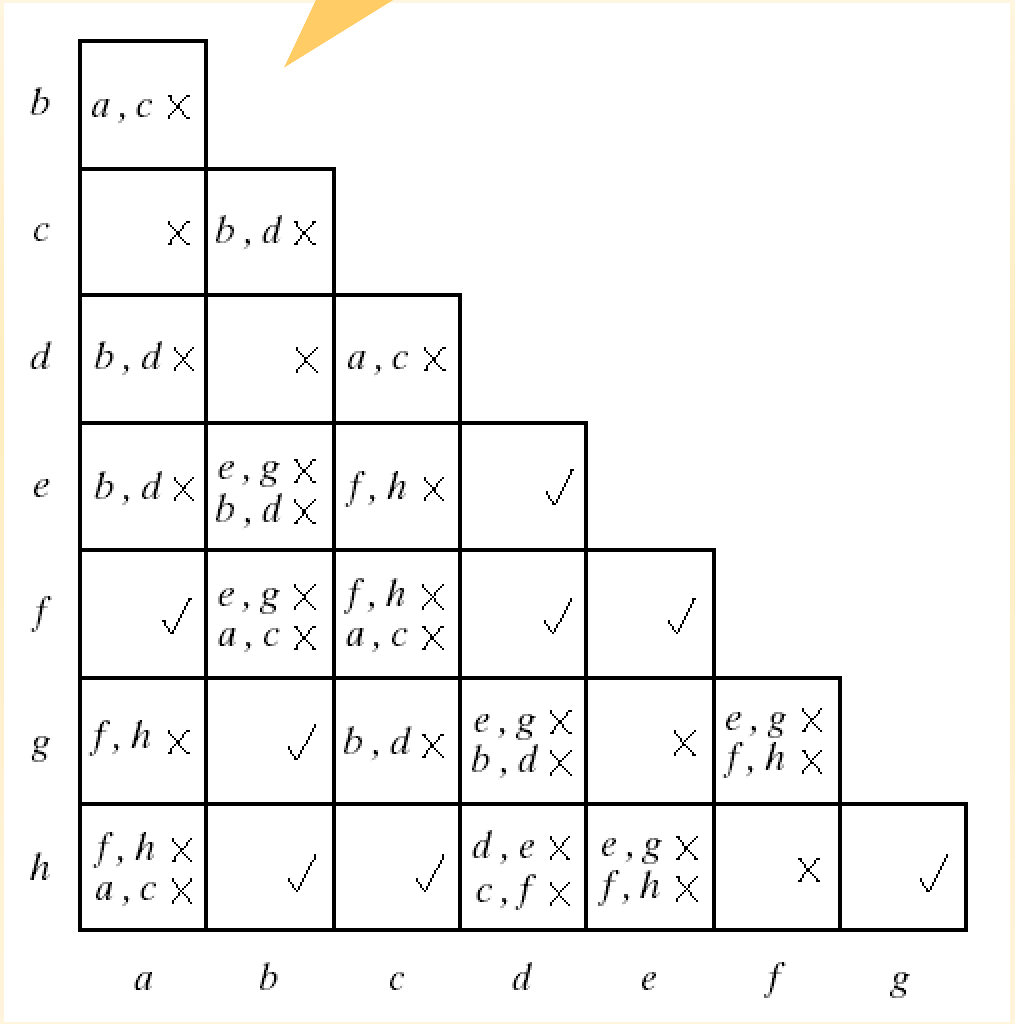
Primitive Flow Table



	TC 00	01	11	10
a	–, –	*f*, –	$\textcircled{a}$, 0	*b*, –
b	*g*, –	–, –	*c*, –	$\textcircled{b}$, 1
c	–, –	*h*, –	$\textcircled{c}$, 1	*d*, –
d	*e*, –	–, –	*a*, –	$\textcircled{d}$, 0
e	$\textcircled{e}$, 0	*f*, –	–, –	*d*, –
f	*e*, –	$\textcircled{f}$, 0	*a*, –	–, –
g	$\textcircled{g}$, 1	*h*, –	–, –	*b*, –
h	*g*, –	$\textcircled{h}$, 1	*c*, –	–, –

Design Example

Implication Table



b	*a*, *c* ×						
c	×	*b*, *d* ×					
d	*b*, *d* ×	×	*a*, *c* ×				
e	*b*, *d* ×	*e*, *g* × *b*, *d* ×	*f*, *h* ×	✓			
f	✓	*e*, *g* × *a*, *c* ×	*f*, *h* × *a*, *c* ×	✓	✓		
g	*f*, *h* ×	✓	*b*, *d* ×	*e*, *g* × *b*, *d* ×	×	*e*, *g* × *f*, *h* ×	
h	*f*, *h* × *a*, *c* ×	✓	✓	*d*, *e* × *c*, *f* ×	*e*, *g* × *f*, *h* ×	×	✓
	a	*b*	*c*	*d*	*e*	*f*	*g*

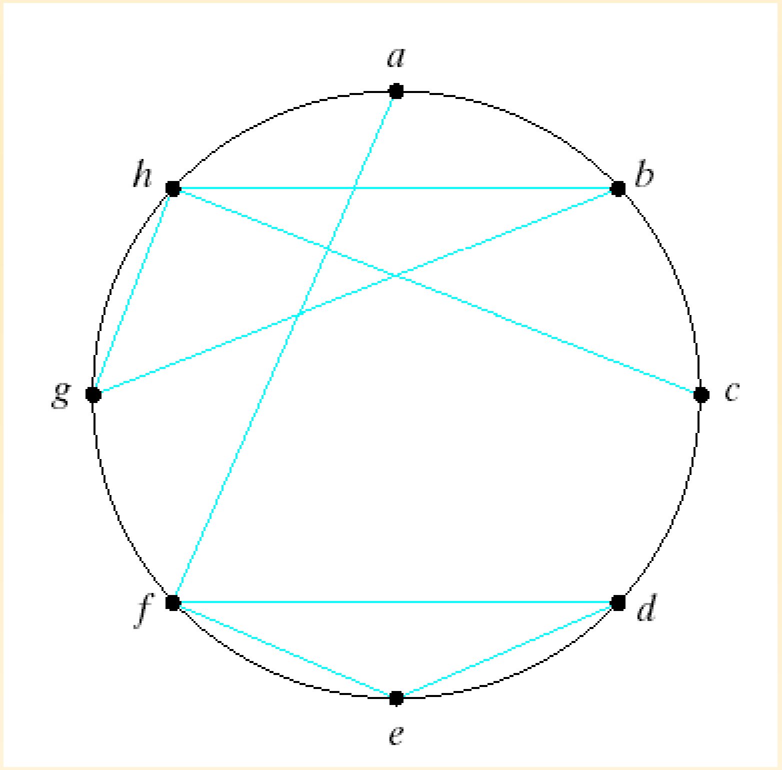
Merging the Flow Table

The compatible pairs:

(*a, f*) (*b, g*) (*b, h*)
(*c, h*) (*d, e*) (*d, f*)
(*e, f*) (*g, h*)

Design Example

Merging the Flow Table



The maximal compatible set:

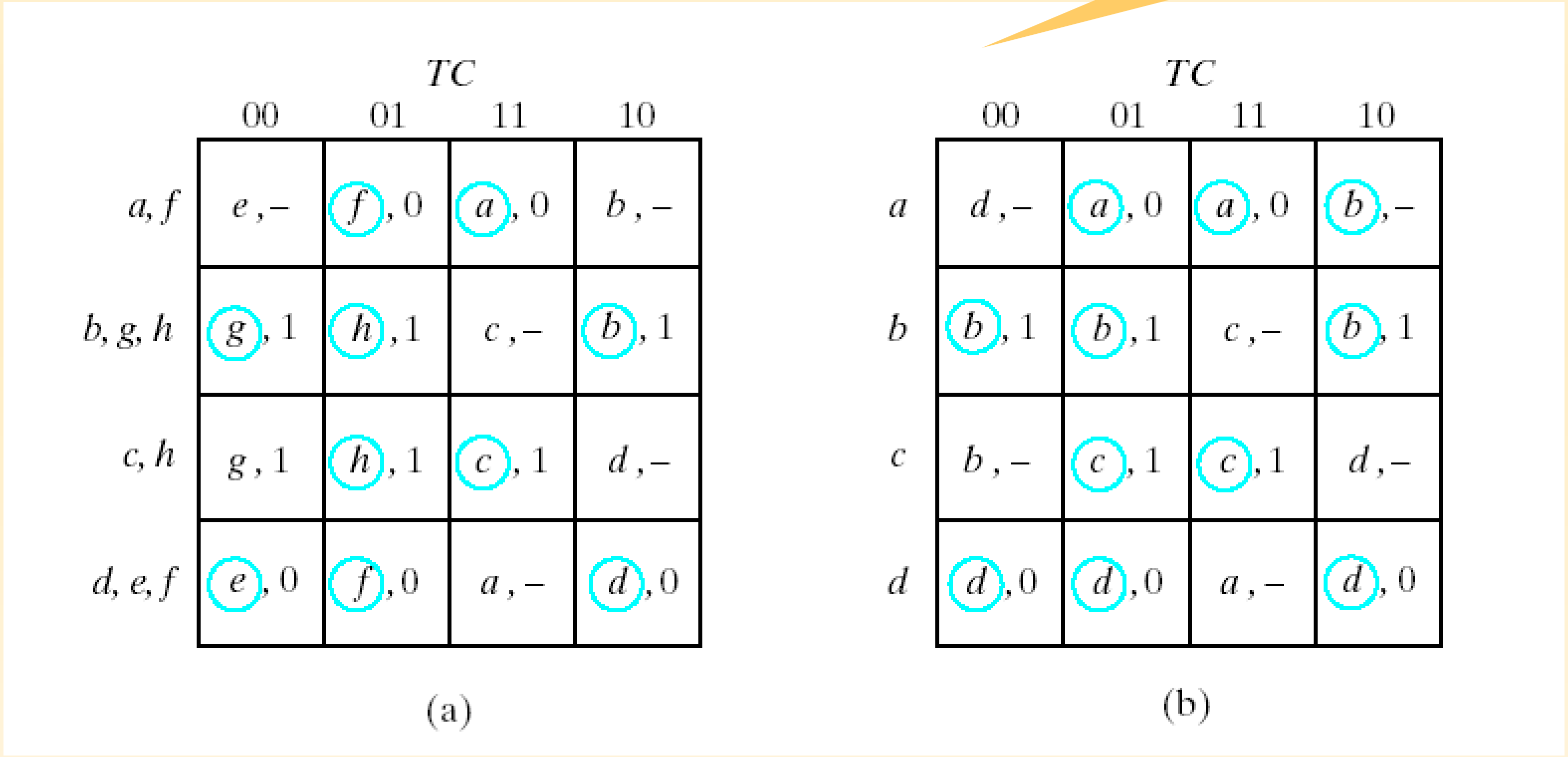
(*a*, *f*) (*b*, *g*, *h*) (*c*, *h*)

(*d*, *e*, *f*)

Design Example

Merging the Flow Table

Reduced flow table



	TC 00	01	11	10
a, f	e , –	f , 0	a , 0	b , –
b, g, h	g , 1	h , 1	c , –	b , 1
c, h	g , 1	h , 1	c , 1	d , –
d, e, f	e , 0	f , 0	a , –	d , 0

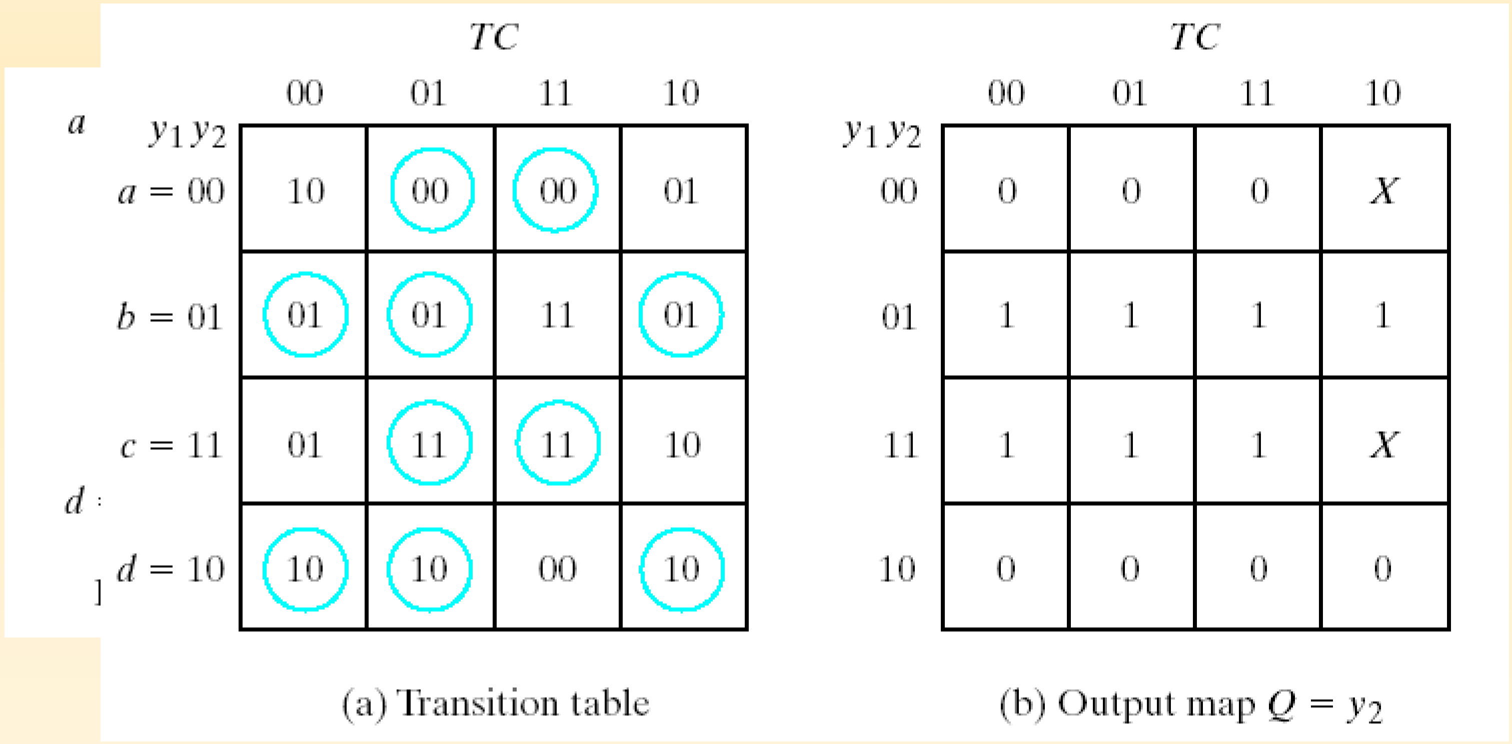
(a)

	TC 00	01	11	10
a	d , –	a , 0	a , 0	b , –
b	b , 1	b , 1	c , –	b , 1
c	b , –	c , 1	c , 1	d , –
d	d , 0	d , 0	a , –	d , 0

(b)

Design Example

State Assignment and Transition Table



y_1y_2 \ TC	00	01	11	10
$a = 00$	10	00	00	01
$b = 01$	01	01	11	01
$c = 11$	01	11	11	10
$d = 10$	10	10	00	10

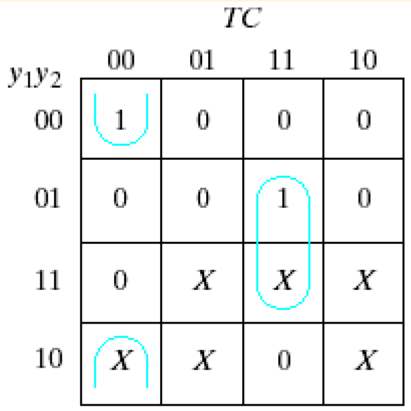
(a) Transition table

y_1y_2 \ TC	00	01	11	10
00	0	0	0	X
01	1	1	1	1
11	1	1	1	X
10	0	0	0	0

(b) Output map $Q = y_2$

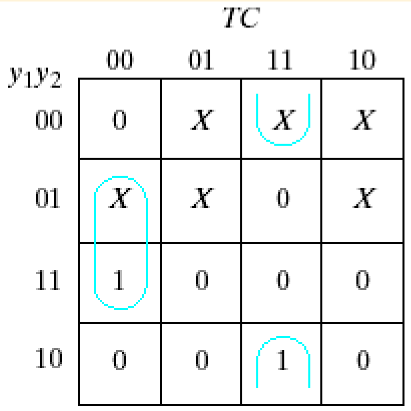
Design Example

Logic Diagram



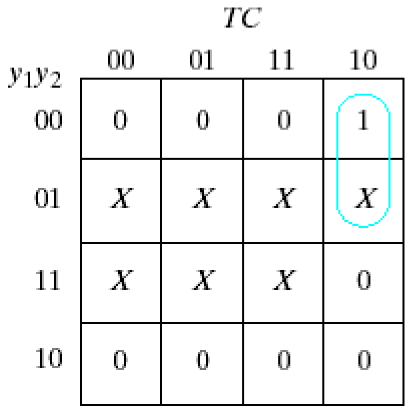
y_1y_2 \ TC	00	01	11	10
00	1	0	0	0
01	0	0	1	0
11	0	X	X	X
10	X	X	0	X

(a) $S_1 = y_2\,TC + y'_2\,T'C'$



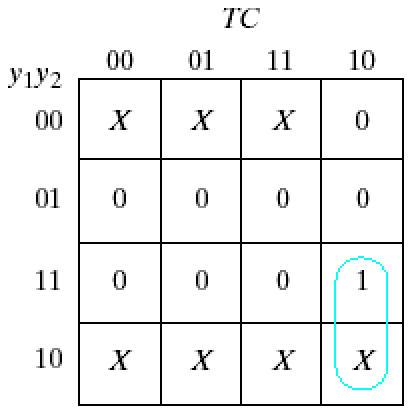
y_1y_2 \ TC	00	01	11	10
00	0	X	X	X
01	X	X	0	X
11	1	0	0	0
10	0	0	1	0

(b) $R_1 = y_2\,T'C' + y'_2\,TC$



y_1y_2 \ TC	00	01	11	10
00	0	0	0	1
01	X	X	X	X
11	X	X	X	0
10	0	0	0	0

(c) $S_2 = y'_1\,TC'$

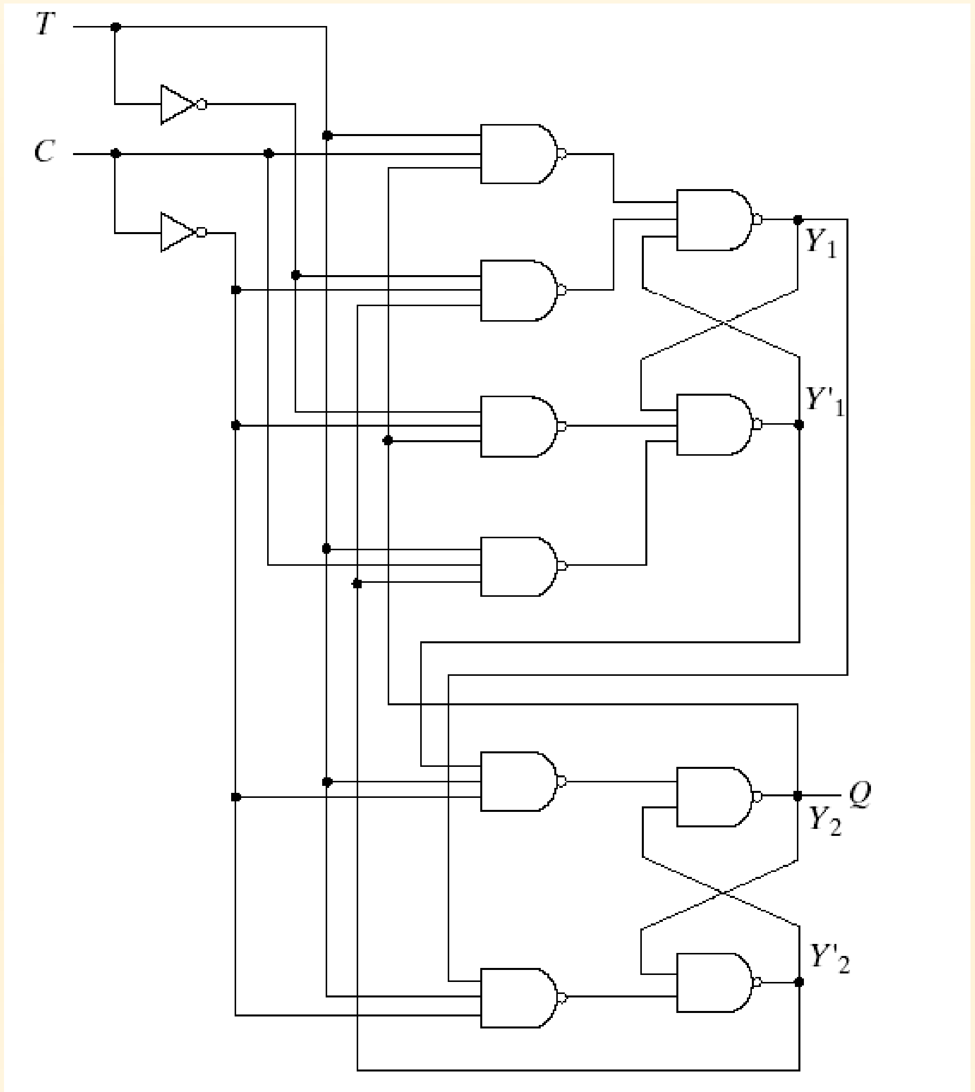


y_1y_2 \ TC	00	01	11	10
00	X	X	X	0
01	0	0	0	0
11	0	0	0	1
10	X	X	X	X

(d) $R_2 = y_1\,TC'$

Design Example

Logic Diagram



T
C
Y1
Y'1
Q
Y2
Y'2