Asynchronous Sequential Logic

LECTURE 4

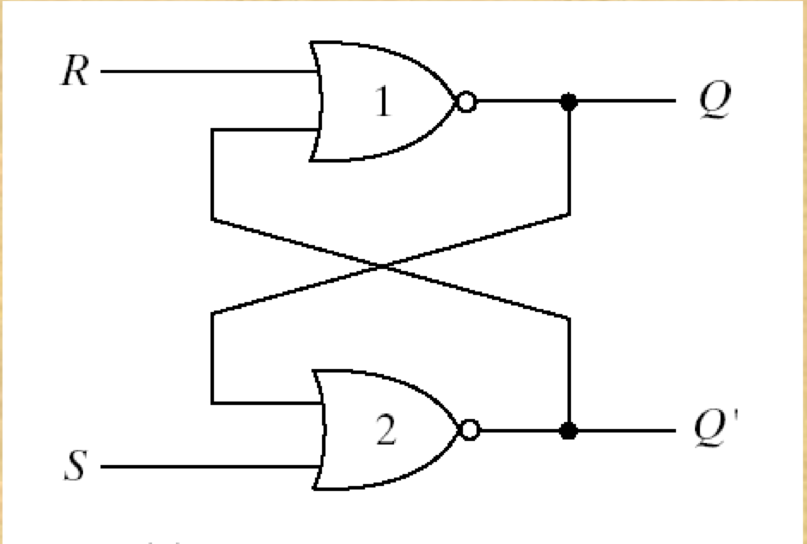
- **Circuits With Latches**
- **Design Procedure**

Dronacharya Group of Institutions

Circuits with Latches

SR Latch

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



(a) Crossed-coupled circuit

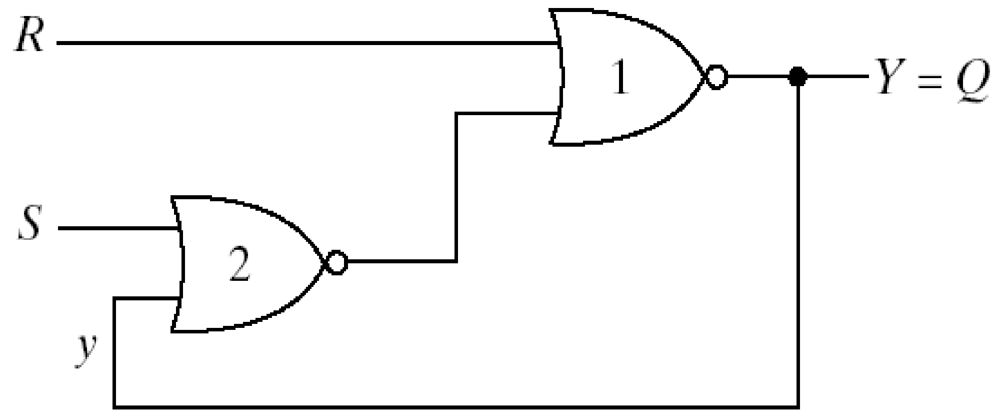
S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(After SR = 10)
0	1	0	1	
0	0	0	1	(After SR = 01)
1	1	0	0	

(b) Truth table

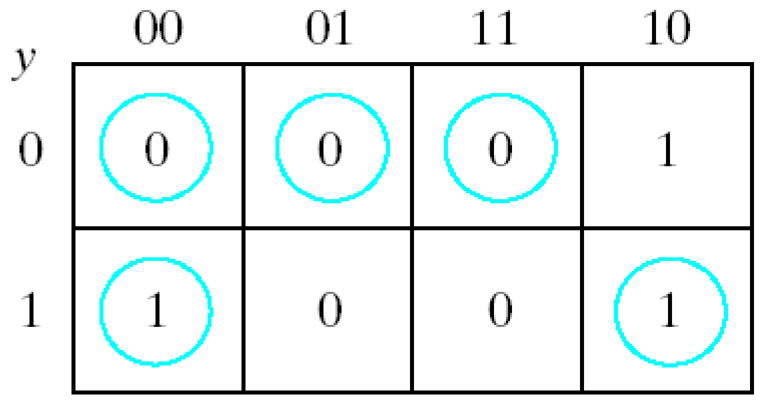
Circuits with Latches

SR Latch

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



(c) Circuit showing feedback



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

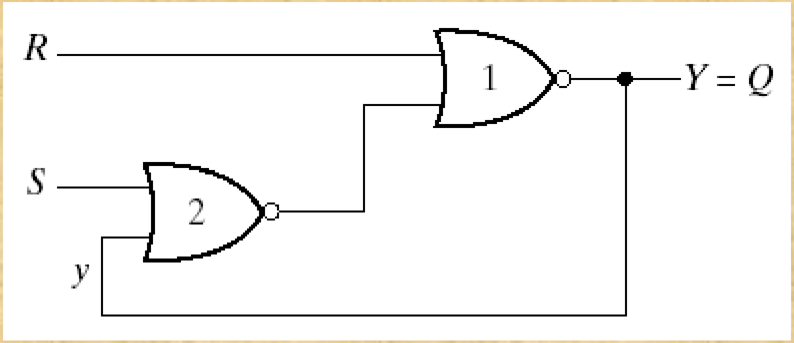
$Y = SR' + R'y$

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

(d) Transition table

Circuits with Latches

SR Latch



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

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

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

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

Circuits with Latches

Analysis Example

$S_1 = x_1 y_2$

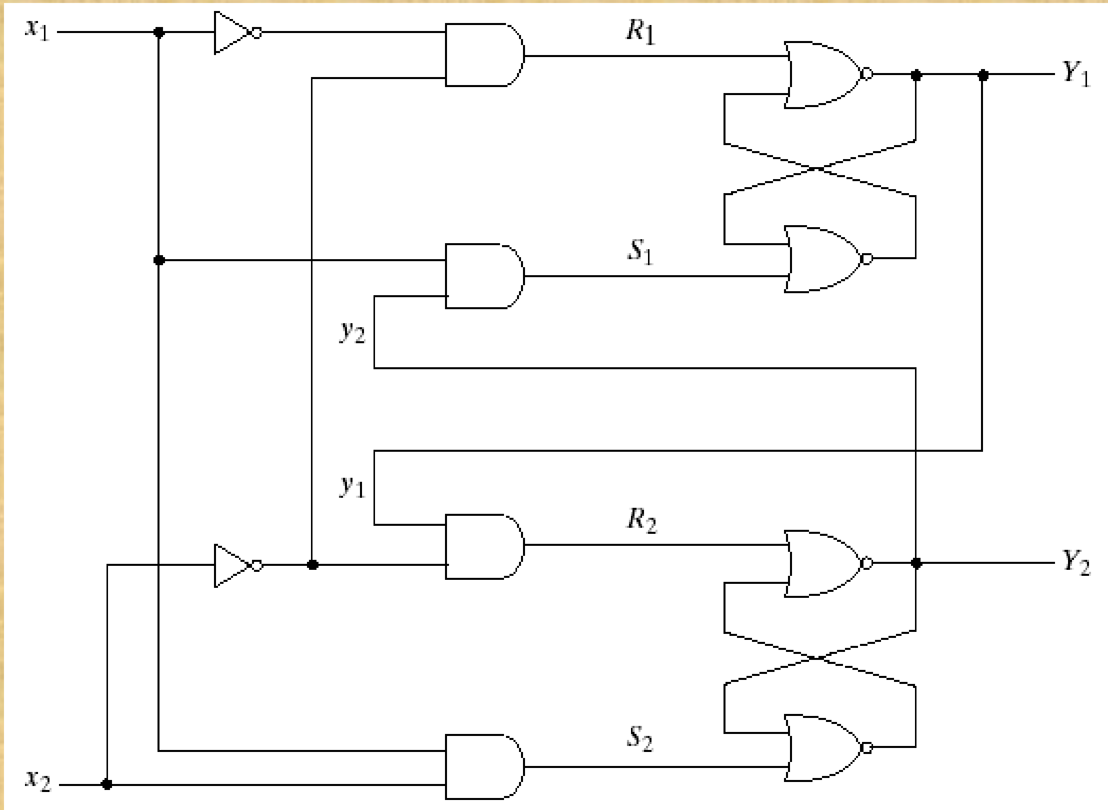
$S_2 = x_1 x_2$

$R_1 = x'_1 x'_2$

$R_2 = x'_2 y_1$

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

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



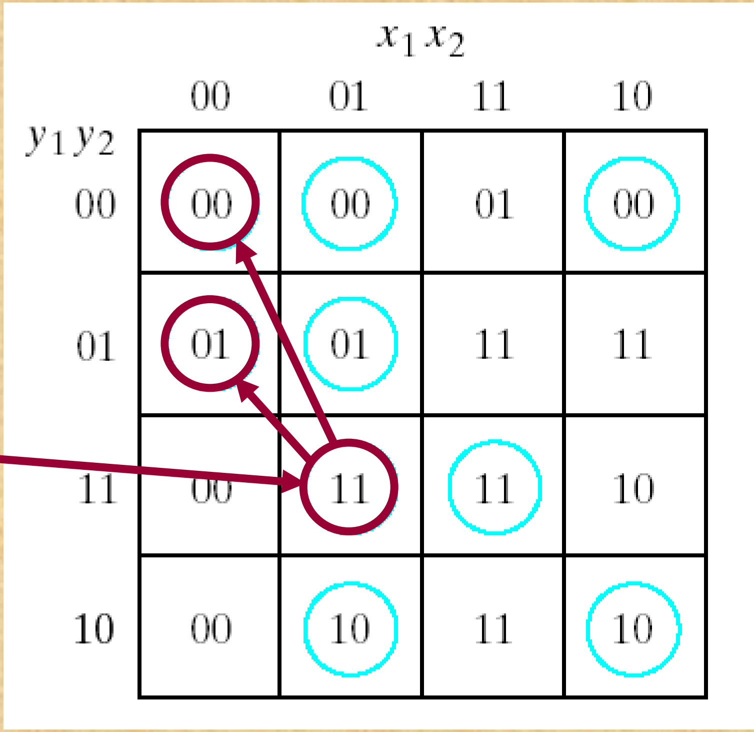
Circuits with Latches

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



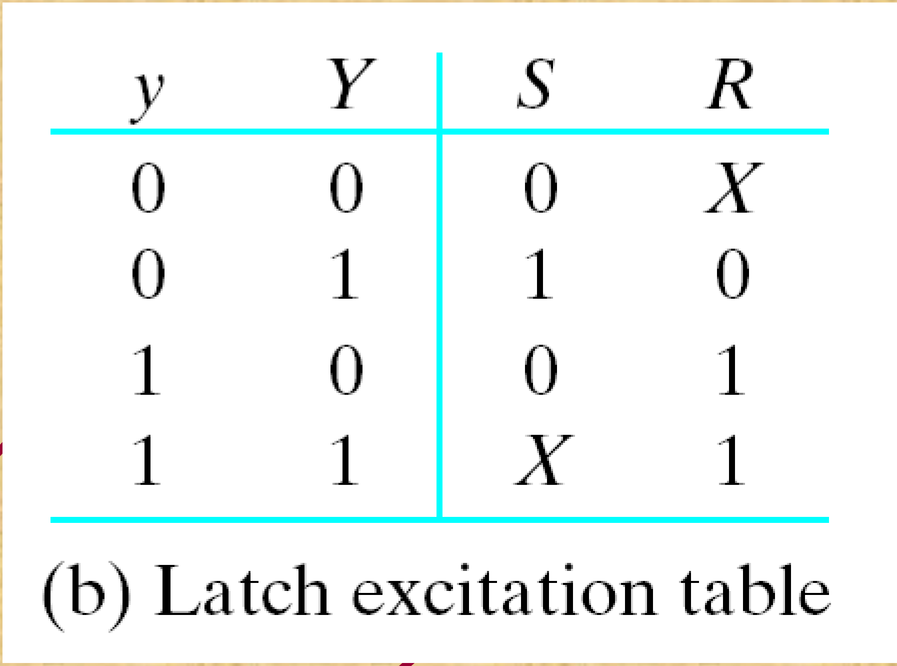
$y_1 y_2$ \ $x_1 x_2$	00	01	11	10
00	00	00	01	00
01	01	01	11	11
11	00	11	11	10
10	00	10	11	10

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

x_1x_2

y	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	1

(b) Latch excitation table

x_1x_2

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

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

x_1x_2

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

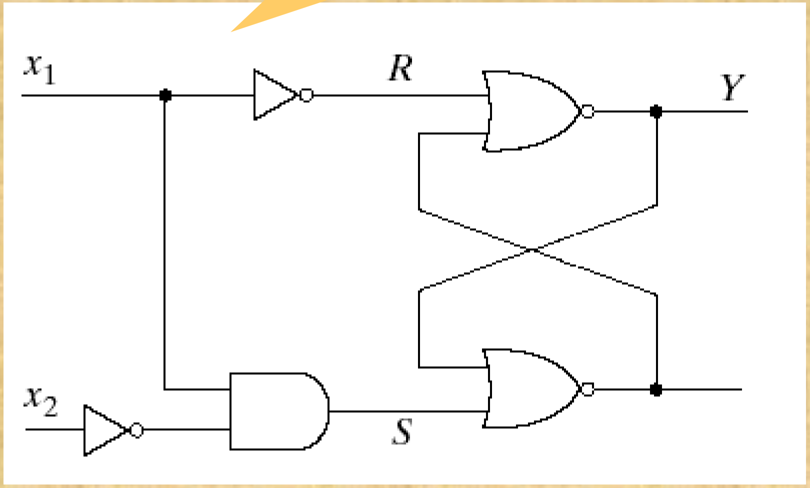
(d) Map for $R = x'_1$

Circuits with Latches

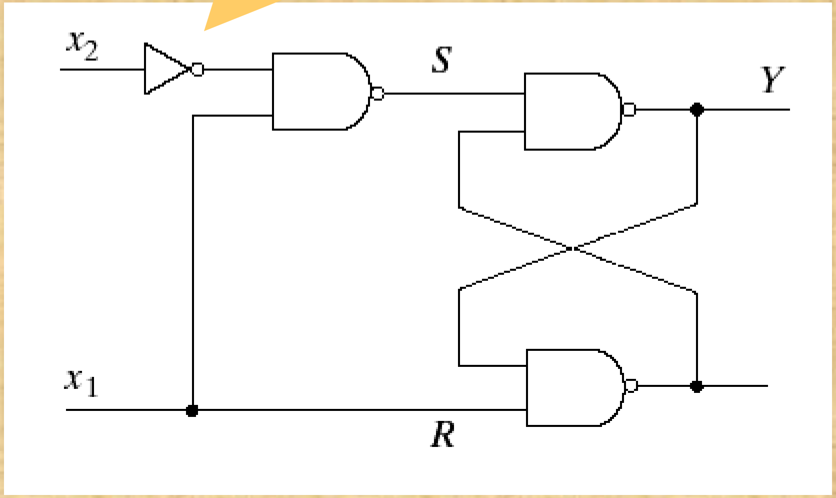
Implementation Example

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

Circuit with NOR latch



Circuit with NAND latch



Design Procedure

Design Example

Design a gated latch circuit with two inputs G (gate) and D (data), and one output Q.

Gated-Latch Total States

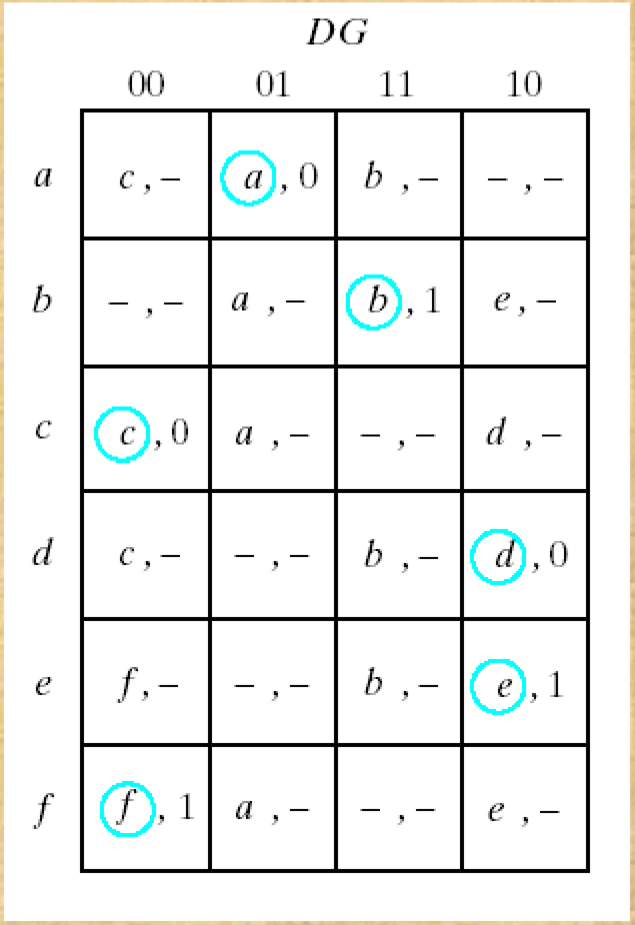
	Inputs		Output	
State	D	G	Q	**comments**
a	0	1	0	$D = Q$ because $G = 1$
b	1	1	1	$D = Q$ because $G = 1$
c	0	0	0	After state *a* or *d*
d	1	0	0	After state *c*
e	1	0	1	After state *b* or *f*
f	0	0	1	After state *e*

Design Procedure

Design Example

State	Inputs		Output
	D	*G*	*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

$\Longrightarrow$



	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* , –

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.

	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* , –

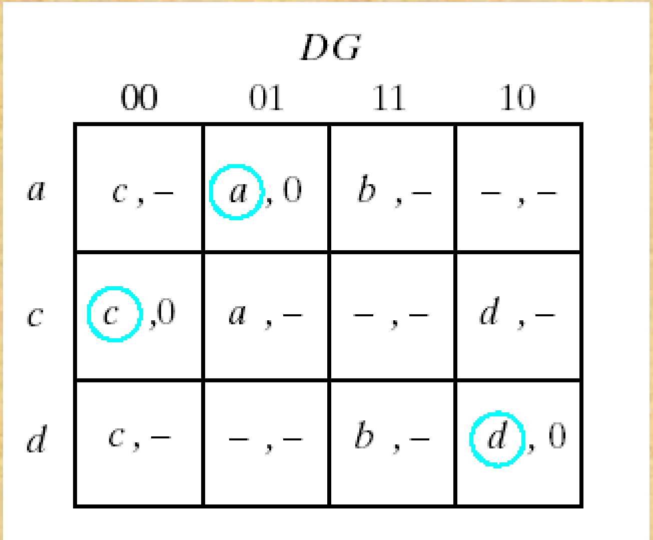
	DG			
	00	01	11	10
a	*c* , –	$\textcircled{a}$, 0	*b* , –	– , –
c	$\textcircled{c}$,0	*a* , –	– , –	*d* , –
d	*c* , –	– , –	*b* , –	$\textcircled{d}$, 0

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

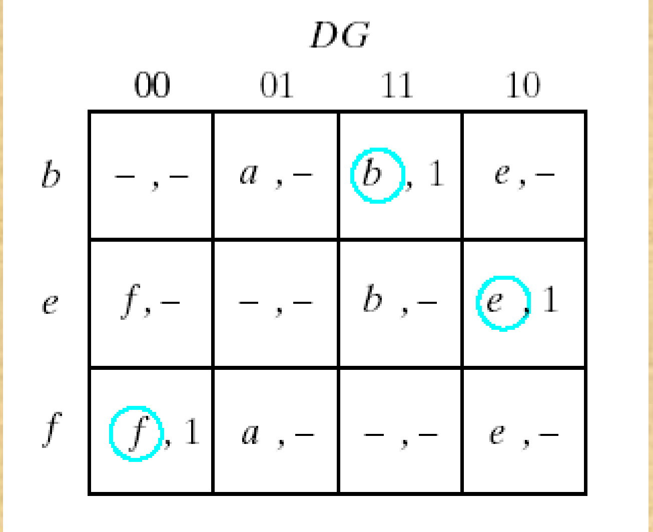
Reduction of the Primitive Flow Table

Design Procedure

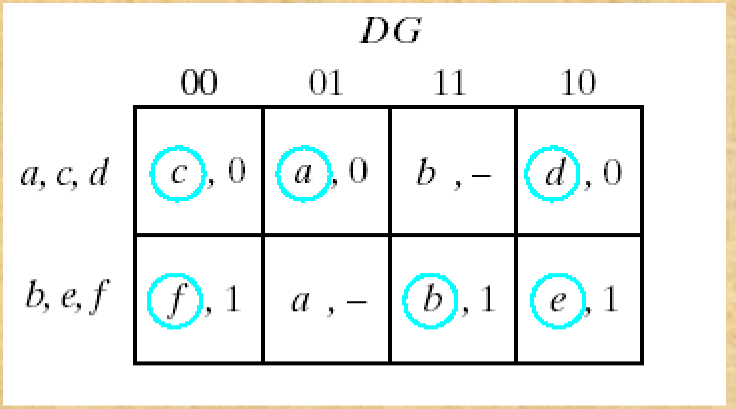
Reduction of the Primitive Flow Table



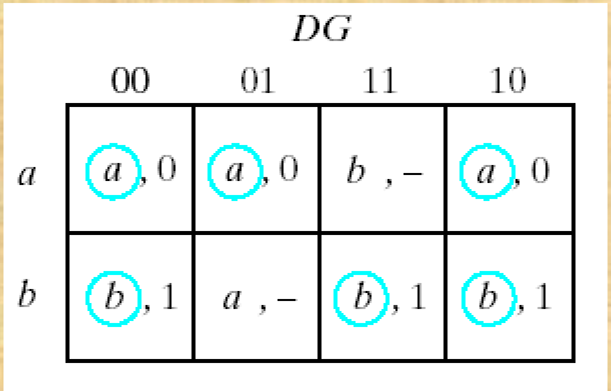
	DG 00	01	11	10
a	c, –	$\textcircled{a}$, 0	b, –	–, –
c	$\textcircled{c}$, 0	a, –	–, –	d, –
d	c, –	–, –	b, –	$\textcircled{d}$, 0



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



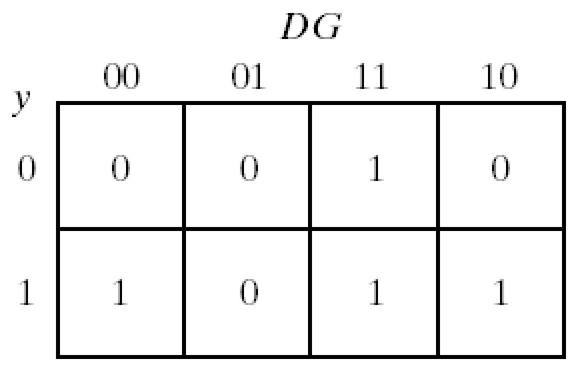
	DG 00	01	11	10
a, c, d	$\textcircled{c}$, 0	$\textcircled{a}$, 0	b, –	$\textcircled{d}$, 0
b, e, f	$\textcircled{f}$, 1	a, –	$\textcircled{b}$, 1	$\textcircled{e}$, 1



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

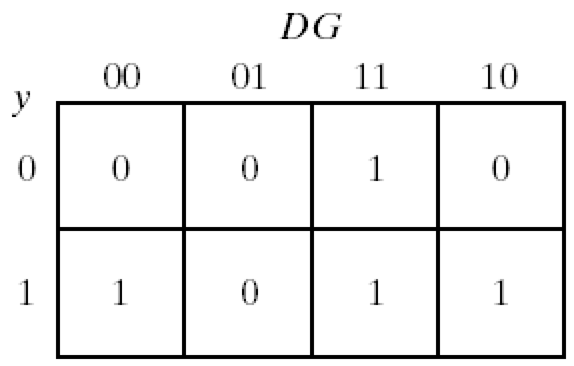
Design Procedure

Transition Table and Logic Diagram



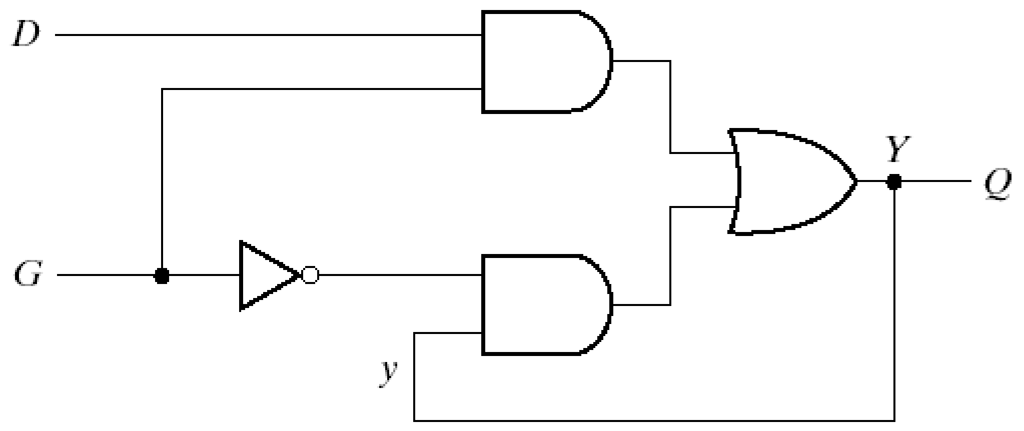
y \ DG	00	01	11	10
0	0	0	1	0
1	1	0	1	1

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



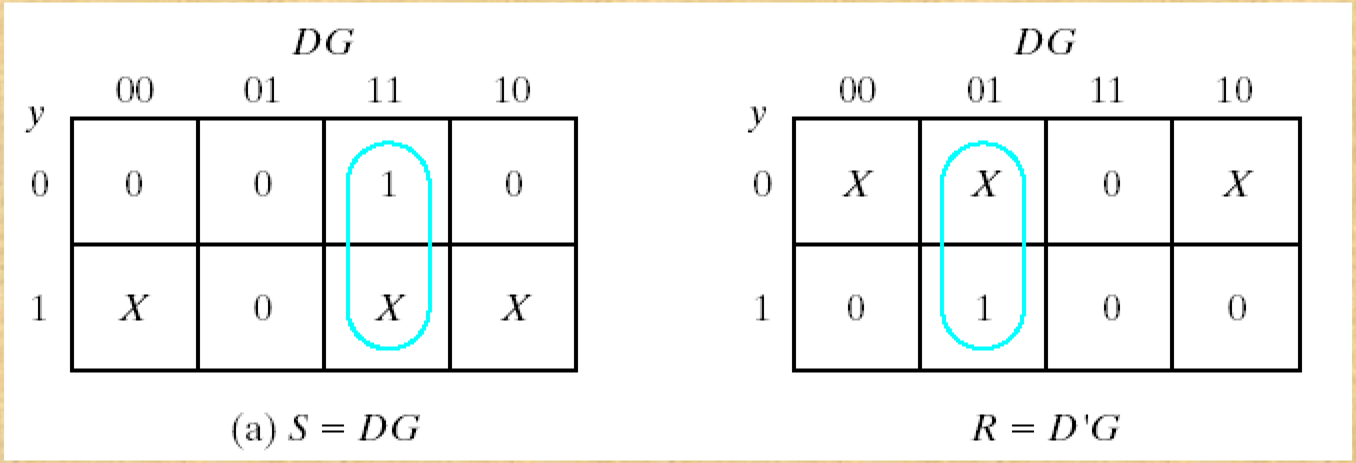
y \ DG	00	01	11	10
0	0	0	1	0
1	1	0	1	1

(b) $Q = Y$



Design Procedure

Circuit With *SR* Latch

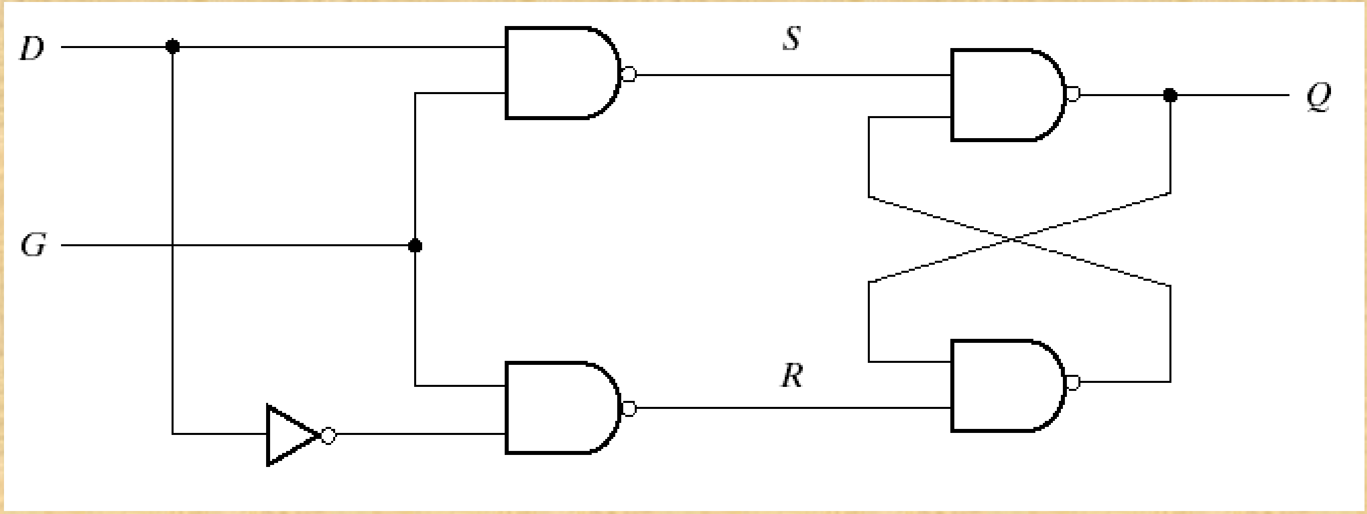


y \ DG	00	01	11	10
0	0	0	1	0
1	X	0	X	X

(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.