Digital Logic and Circuit
Paper Code: CS-102

Outline

- **Sequential Circuit**
- **State Reduction and Assignment**
- **FlipFlop Excitation Table**

STATE REDUCTION AND ASSIGNMENT

Memory elements can store binary information

- This information at any given time determines the state of the circuit at that time

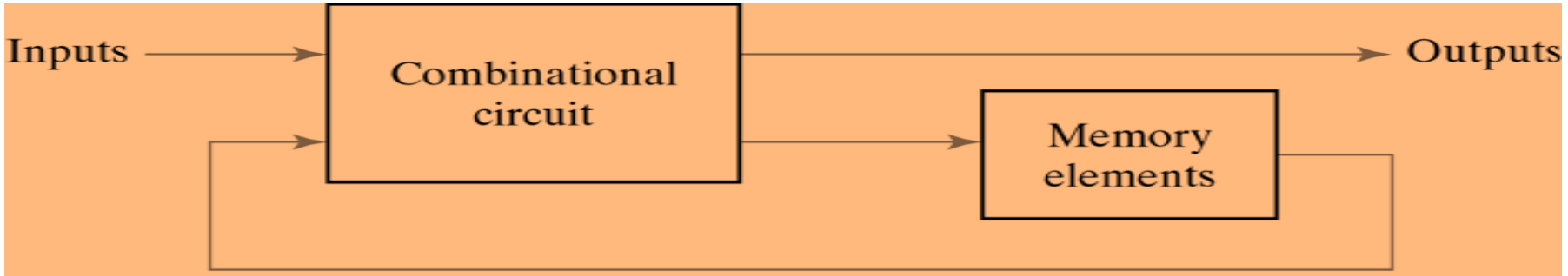


Fig. 5-1 Block Diagram of Sequential Circuit

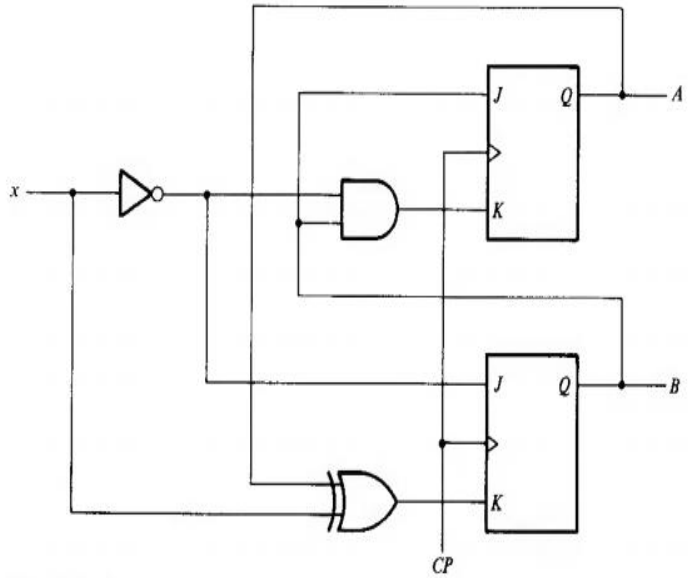
State Reduction

- Any design process must consider the problem of minimizing the cost of the final circuit.
- The two most obvious cost reductions are reductions in the number of flip-flops and the number of gates. Because these two items seem the most obvious, they have been extensively studied and investigated.
- The reduction of the number of flip-flops in a sequential circuit is referred to as the state-reduction problem.
- State-reduction algorithms are concerned with procedures for reducing the number of states in a state table while keeping the external input-output requirements unchanged.

Inputs to flipflops

JA=B JB = x'

KA= Bx‘ KB = A’x + Ax' = A XOR x

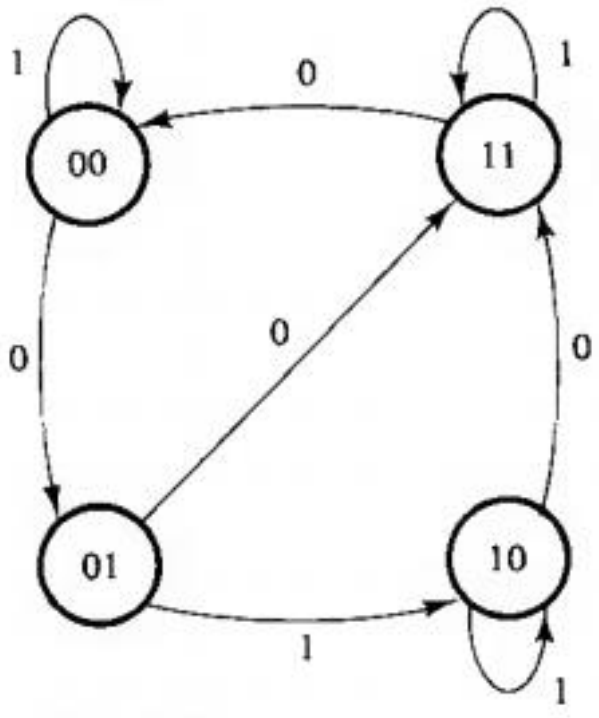


State Table for Sequential Circuit with *JK* flip-Flops

Present state		Input	Next state		Flip-flop inputs			
A	*B*	*x*	*A*	*B*	*JA*	*KA*	*JB*	*KB*
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

JK Flip-Flop		
J *K*	*Q*(*t* + 1)	
0 0	*Q*(*t*)	No change
0 1	0	Reset
1 0	1	Set
1 1	*Q*′(*t*)	Complement

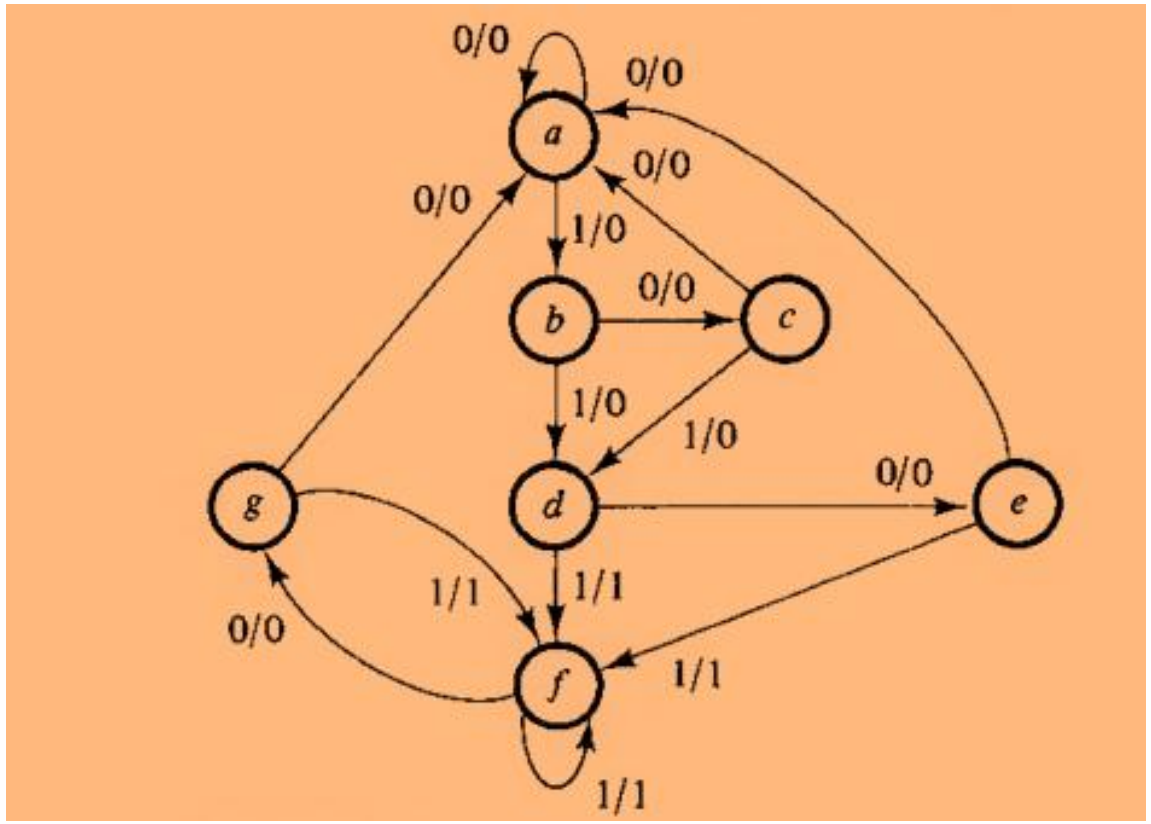
State diagram



State Table for Sequential Circuit with *JK* flip-Flops

Present state		Input	Next state		Flip-flop inputs			
A	*B*	*x*	*A*	*B*	*JA*	*KA*	*JB*	*KB*
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

Example: Consider the state diagram

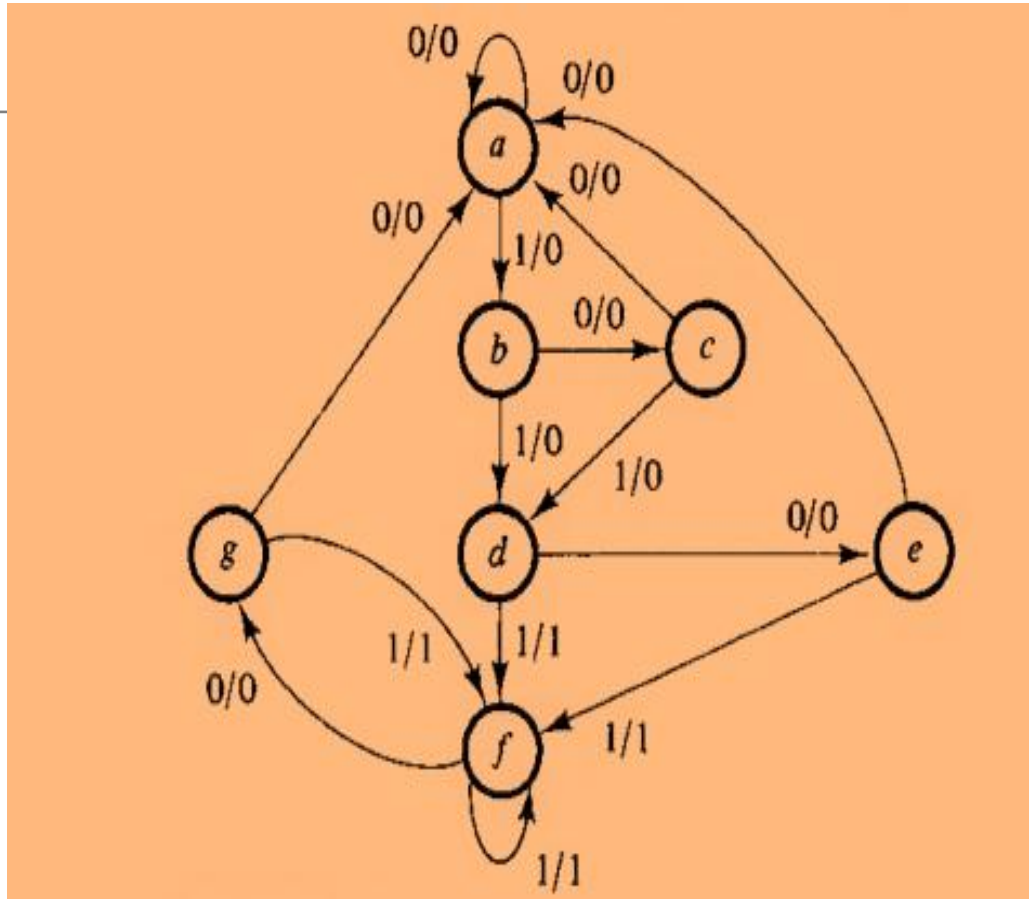


- There are an infinite number of input sequences that may be applied to the circuit; each results in a unique output sequence.

- As an example, consider the input sequence 01010110100 starting from the initial state a. Each input of 0 or 1 produces an output of 0 or 1 and causes the circuit to go to the next state.

- With the circuit in initial state a , an input of 0 produces an output of 0 and the circuit remains in state a .

state	*a*	*a*	*b*	*c*	*d*	*e*	*f*	*f*	*g*	*f*	*g*	*a*
input	0	1	0	1	0	1	1	0	1	0	0	
output	0	0	0	0	0	1	1	0	1	0	0	



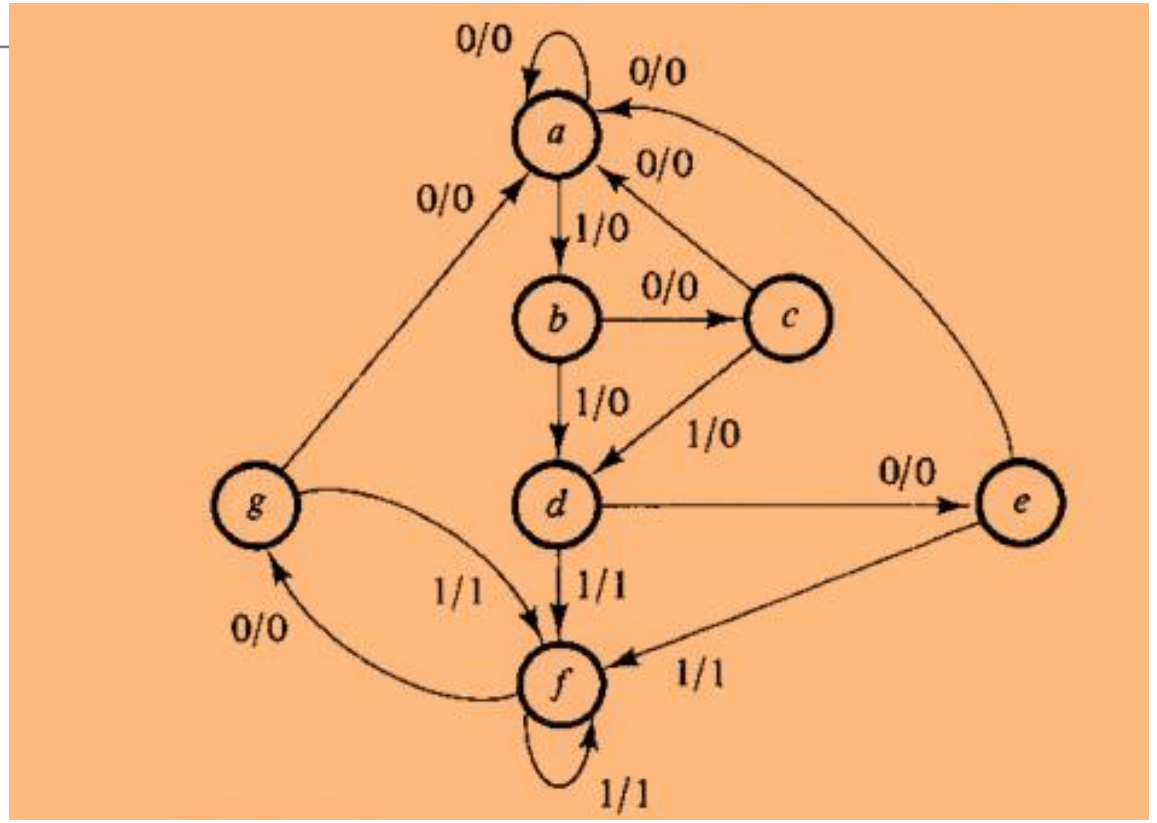
- In each column, we have the present state, input value, and output value. The next state is written on top of the next column.

- Now let us assume that we have found a sequential circuit whose state diagram has less than seven states and we wish to compare it with the circuit whose state diagram is given in the above figure.
- If identical input sequences are applied to the two circuits and identical outputs occur for all input sequences, then the two circuits are said to be equivalent (as far as the input-output is concerned) and one may be replaced by the other.
- The problem of state reduction is to find ways of reducing the number of states in a sequential circuit without altering the input-output relationships.

State table

State Table

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



- An algorithm for the state reduction of a completely specified state table is given here without proof:

"Two states are said to be equivalent if, for each member of the set of inputs, they give exactly the same output and send the circuit either to the same state or to an equivalent state. When two states are equivalent, one of them can be removed without altering the input-output relationships."

- we look for two present states that go to the same next state and have the same output for both input combinations.
- States g and e are two such states: they both go to states a and f and have outputs of 0 and 1 for x = 0 and x = 1, respectively.
- Therefore, states g and e are equivalent; one can be removed.

State Table

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

State Table

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

Reducing the State Table

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

Reduced State Table

State Table

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

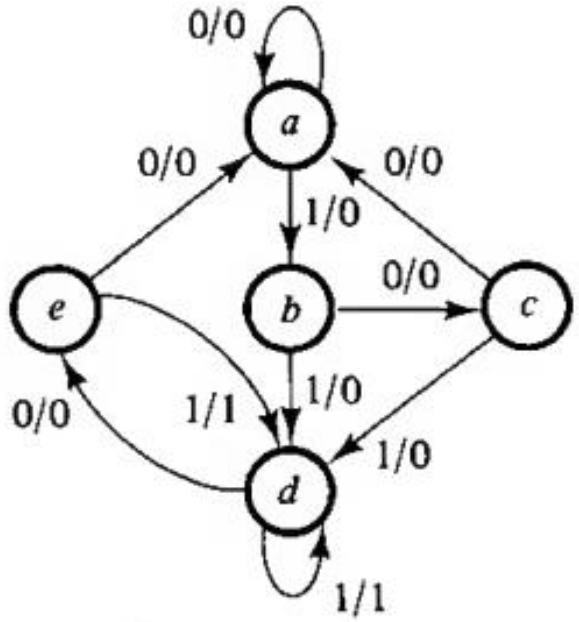
Reducing the State Table

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

Reduced State Table

Present State	Next state $x=0$	Next state $x=1$	Output $x=0$	Output $x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

The state diagram for the reduced table consists of only five states



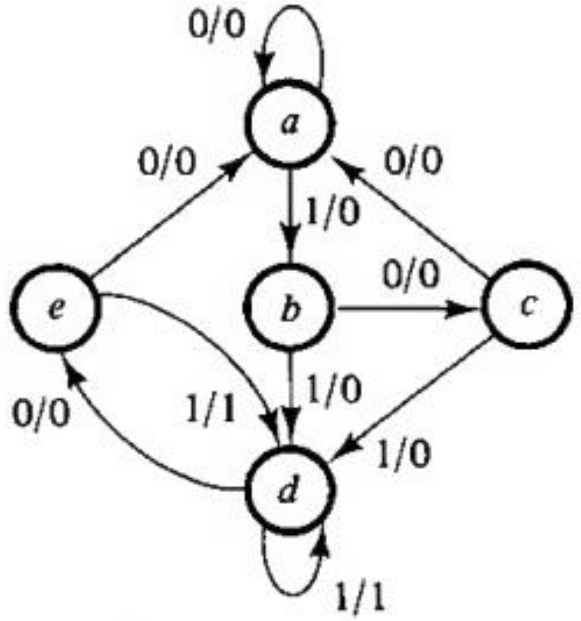
Reduced State Table

Present State	Next state		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

- As an example, consider the input sequence 01010110100 starting from the initial state a. Each input of 0 or 1 produces an output of 0 or 1 and causes the circuit to go to the next state.

- we note that the same output sequence results although the stat sequence is different

state	*a*	*a*	*b*	*c*	*d*	*e*	*d*	*d*	*e*	*d*	*e*	*a*
input	0	1	0	1	0	1	1	0	1	0	0	
output	0	0	0	0	0	1	1	0	1	0	0	



State Assignment

- The cost of the combinational-circuit part of a sequential circuit can be reduced by using the known simplification methods for combinational circuits.
- However, there is another factor, known as the state -assignment problem, that comes into play in minimizing the combinational gates.
- State-assignment procedures are concerned with methods for assigning binary values to states in such a way as to reduce the cost of the combinational circuit that drives the flip-flops.
- This is particularly helpful when a sequential circuit is viewed from its external input-output terminals.
- Such a circuit may follow a sequence of internal states, but the binary values of the individual states may be of no consequence as long as the circuit produces the required sequence of outputs for any given sequence of inputs.
- This does not apply to circuits whose external outputs are taken directly from flip-flops with binary sequences fully specified.

For the previous example

Three Possible Binary State Assignments

State	Assignment 1	Assignment 2	Assignment 3
a	001	000	000
b	010	010	100
c	011	011	010
d	100	101	101
e	101	111	011

Three Possible Binary State Assignments

State	Assignment 1	Assignment 2	Assignment 3
a	001	000	000
b	010	010	100
c	011	011	010
d	100	101	101
e	101	111	011

Reduced State Table

	Next state		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	*a*	*b*	0	0
b	*c*	*d*	0	0
c	*a*	*d*	0	0
d	*e*	*d*	0	1
e	*a*	*d*	0	1

Reduced State Table with Binary Assignment 1

	Next State		Output	
Present state	$x = 0$	$x = 1$	$x = 0$	$x = 1$
001	001	010	0	0
010	011	100	0	0
011	001	100	0	0
100	101	100	0	1
101	001	100	0	1

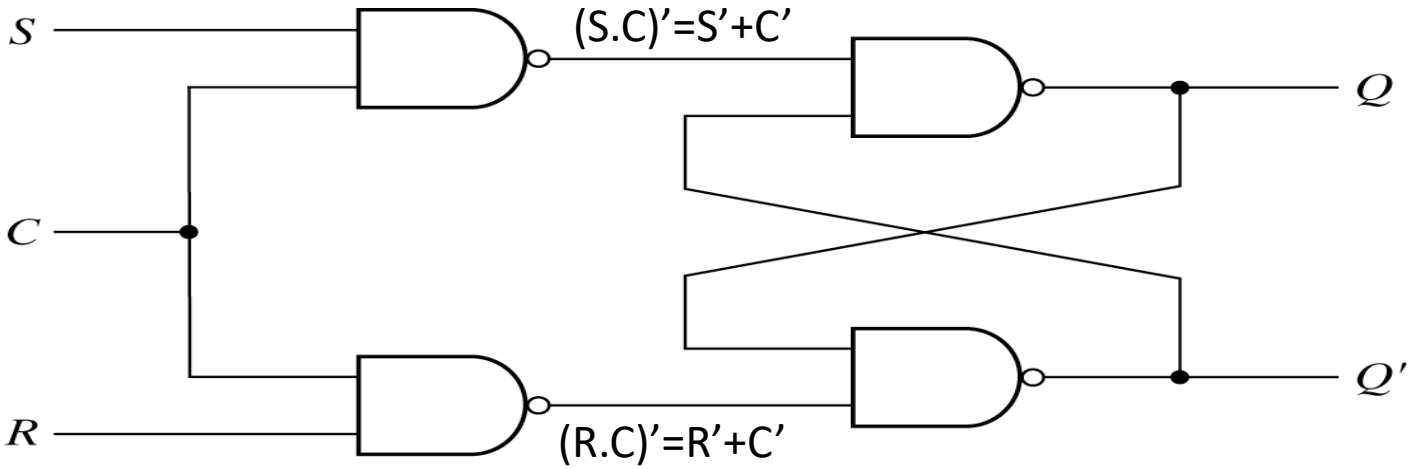
FLIP-FLOP EXCITATION TABLES

- The characteristic table is useful for analysis and for defining the operation of the flipflop.
- It specifies the next state when the inputs and present state are known.
- During the design process, we usually know the transition from present state to next state and wish to find the flip-flop input conditions that will cause the required transition.
- For this reason, we need a table that lists the required inputs for a given change of state. Such a list is called an excitation table.

NAND Latch

S	R	Q	Q′
1	0	0	1
1	1	0	1
0	1	1	0
1	1	1	0
0	0	1	1

RS Latch with Control Input



(a) Logic diagram

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; Reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

(b) Function table

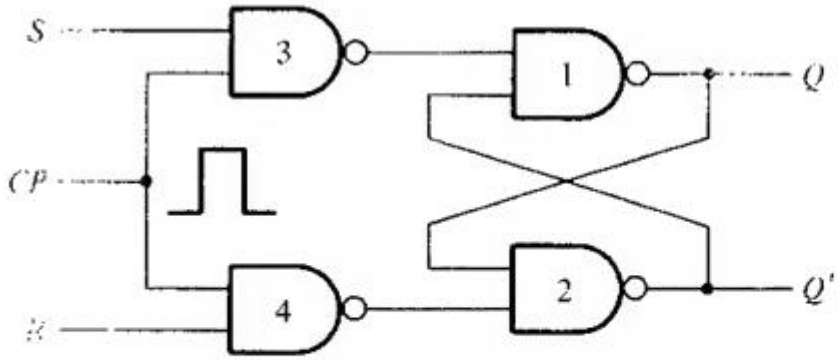
Fig. 5-5 SR Latch with Control Input

We want to change the input when it is required

NAND TABLE

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Characteristic Table of RS flipflop



(a) Logic diagram

Q	S	R	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Indeterminate

(b) Characteristic table

RS FlipFlop Excitation table

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; Reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

(a) Truth table

Q	S	R	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Indeterminate

(b) Characteristic table

$Q(t)$	$Q(t+1)$	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

(c) Excitation table

Excitation table for RS, JK, D and T flip flop

Flip-Flop Excitation Tables

$Q(t)$	$Q(t+1)$	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

(a) *RS*

$Q(t)$	$Q(t+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(b) *JK*

$Q(t)$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

(c) *D*

$Q(t)$	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

(b) *T*

Characteristic Tables

Flip-Flop Characteristic Tables

RS Flip-Flop

S	R	Q(t + 1)	
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Unpredictable

JK Flip-Flop

J	K	Q(t + 1)	
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

D Flip-Flop

D	Q(t + 1)	
0	0	Reset
1	1	Set

T Flip-Flop

T	Q(t + 1)	
0	Q(t)	No change
1	Q'(t)	Complement

Suggested Reading

- M. Morris Mano, Digital Logic and Computer Design, PHI.

Thank you