State Reduction and Assignment

Sparsh Mittal

State reduction

State reduction

Two sequential circuits may exhibit the same input–output behavior, but have a different number of internal states in their state diagram.

Lets discuss properties of sequential circuits that may simplify a design by reducing the number of gates and flip-flops it uses. Reducing the number of flip-flops (called state reduction) reduces the cost of a circuit.

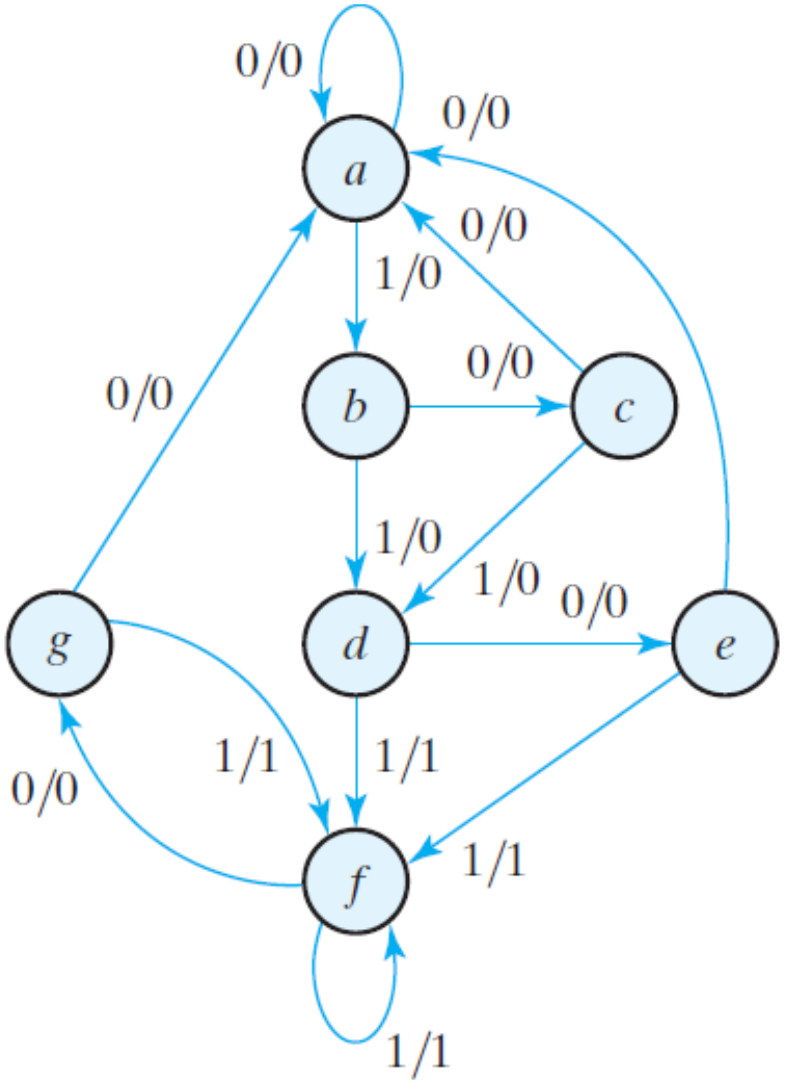
An unpredictable effect in reducing the number of flip-flops is that sometimes the equivalent circuit (with fewer flip-flops) may require more combinational gates to realize its next state and output logic.

Solved example

Only the input–output sequences are important; the internal states are used merely to provide

the required sequences.

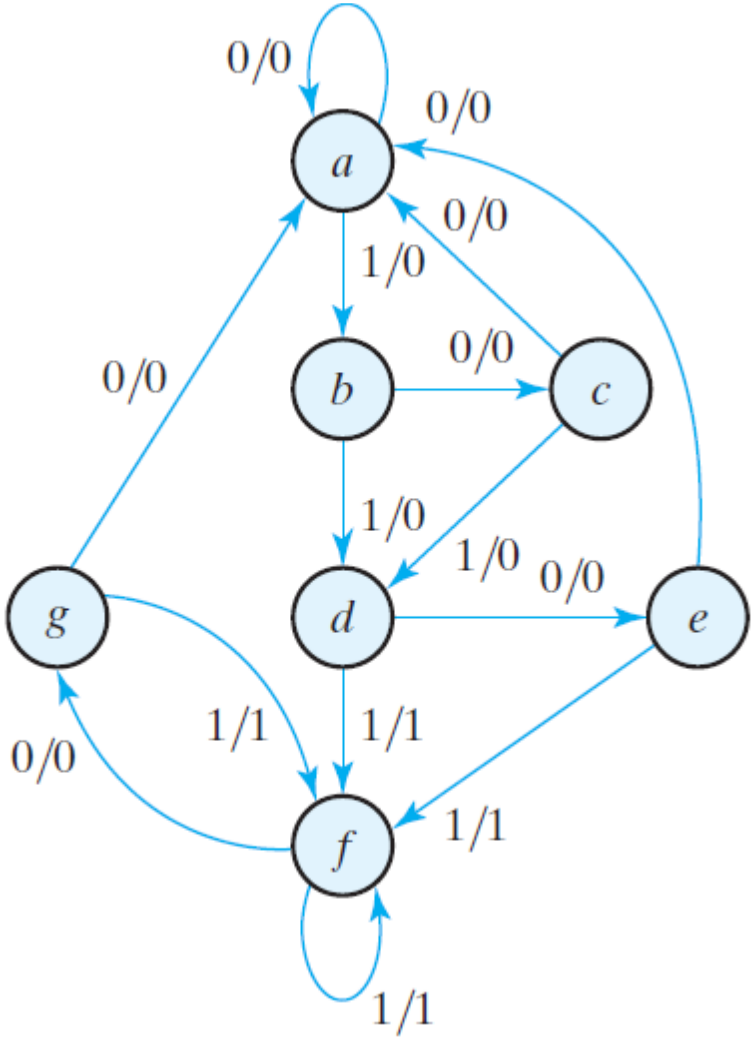
==> States marked inside the circles are denoted by letter symbols instead of their binary values.



Solved example

Infinite input sequences can be applied here, each leading to a unique output

Consider the input sequence 01010110100 starting from the initial 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	

State reduction task

Assume a circuit whose state diagram has <7 states.

If that circuit produces identical outputs for all input sequences, the two circuits are equivalent as far as input–output is concerned.

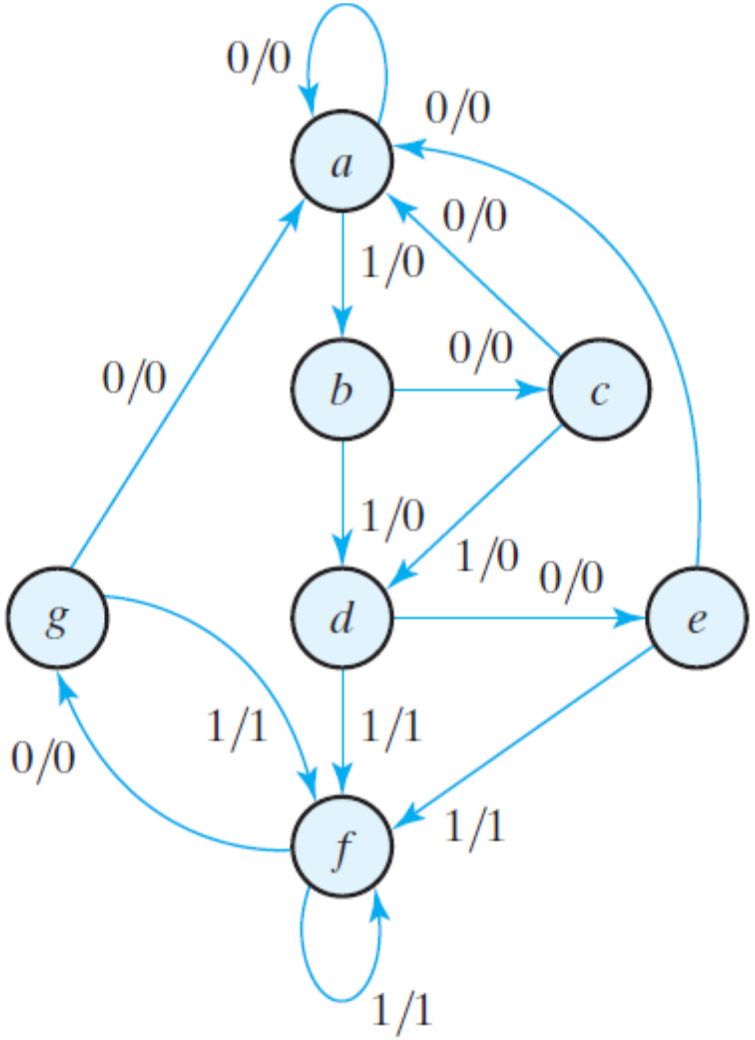
State reduction task: reduce #states in a sequential circuit without altering input–output relationships.

Theorem: 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

Create a State Table

Table 5.6
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



Perform state reduction

Table 5.6
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	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Look for two present states that go to same next state and have same output for both input combinations. Answer: e and g. They are equivalent.

Remove state g and replace it by e.

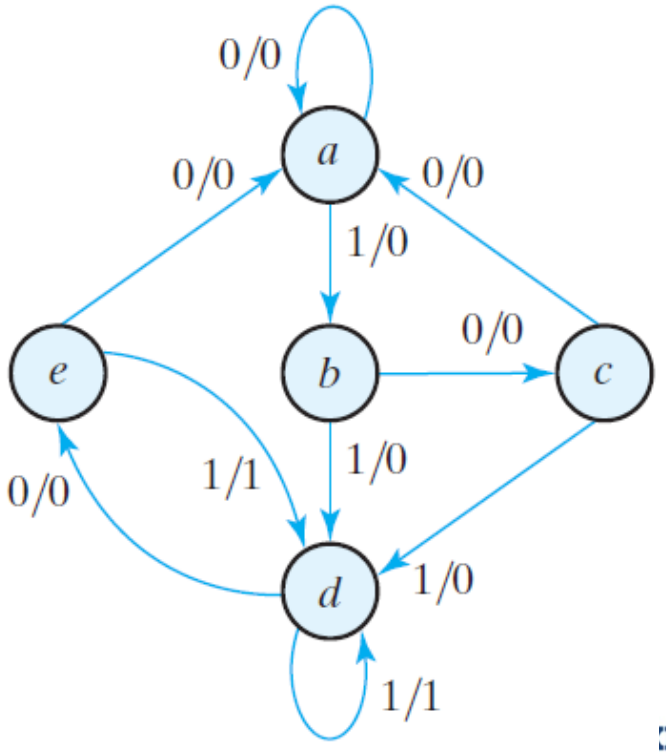
Reducing the State Table

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	e	f	0	1

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

States f and d are equivalent, and state f can be removed and replaced by d .



Summary

The number of states is reduced from 7 to 5.

In general, reducing #states in a state table may result in a circuit with less equipment. However, it does not guarantee a saving in the number of flip-flops or the number of gates.

In actual practice designers may skip this step because target devices are rich in resources.

State assignment

Need of assignment

To design a sequential circuit with physical components, we must assign unique coded binary values to the states.

For m states, the codes must contain n bits, where $2^n >= m$. If m=7, n=3 and one state is unused.

Unused states are treated as don't-care conditions during the design.

Since don't-care conditions usually help in obtaining a simpler circuit, it is more likely but not certain that the circuit with five states will require fewer combinational gates than the one with seven states.

Ways of assignment

Three Possible Binary State Assignments

State	Assignment 1, Binary	Assignment 2, Gray Code	Assignment 3, One-Hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

One-hot encoding

It uses as many bits as there are states in the circuit.

At any given time, only one bit is equal to 1 while all others are kept at 0.

It uses one flip-flop per state, which is not an issue for register-rich FPGAs

Benefits: simpler decoding logic for the next state and output.

Usually faster than machines with sequential binary encoding,

Area required by the extra flip-flops can be offset by the area saved by using simpler decoding logic

Reduced state table with binary assignment 1

Reduced State Table with Binary Assignment 1

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

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

The binary form of state table is used to derive next state and output-forming combinational logic part of sequential circuit.

The complexity of combinational circuit depends on the binary state assignment chosen.