CHAPTER 9

Simplification of Sequential Circuits

In this chapter, we will first look at a technique to remove redundant states from sequential systems. We will then introduce the concept of partitions as another approach to reducing the number of states and as a technique to find state assignments that reduce the amount of combinational logic.

Two states of a sequential system are said to be *equivalent* if every input sequence will produce the same output sequence starting in either state. If the output sequence is the same, then we don't need to know in which of the two states we started. That definition is rather hard to apply, since we must try a very long input sequence or a large number of shorter sequences to be sure that we satisfied the definition.

A more practical definition is:

> Two states of a sequential system are *equivalent* if, starting in either state, any one input produces the same output and equivalent next states.

If two states are equivalent, we can remove one of them and have a system with fewer states. Usually, systems with fewer states are less expensive to implement. This is particularly true if the reduced system requires fewer state variables. For example, reducing a system from six states to four states reduces the number of flip flops required to store the state from three to two. If the system is built with *JK* flip flops and there is one input, *x*, and one output, *z*, we have only five functions to implement instead of seven. Furthermore, the *J* and *K* inputs are two-variable functions rather than three and the output is also a function of one less variable (three for a Mealy and two for a Moore system). Fewer variables usually means less combinational logic.

Occasionally, we can tell states are equivalent by just inspecting the state table. We will look at the simple example of Table 9.1 to illustrate this approach.

Table 9.1 A state table.

q	$q^\star$ $x = 0$	$x = 1$	z $x = 0$	$x = 1$
A	C	B	0	0
B	E	D	0	0
C	A	D	0	1
D	A	B	0	1
E	A	B	0	1

Note that for states D and E, the next state is the same (A) for $x = 0$ and is also the same (B) for $x = 1$. Also, the outputs are the same for each state, for both $x = 0$ and $x = 1$. Thus, we can delete one of the states. We will remove state E and obtain Table 9.2.

Table 9.2 Reduced state table.

q	$q^\star$ $x = 0$	$x = 1$	z $x = 0$	$x = 1$
A	C	B	0	0
B	D	D	0	0
C	A	D	0	1
D	A	B	0	1

We replaced each appearance of E in the state table by D. Although it is not obvious, no further reduction is possible. Often, we cannot see the equivalences so easily.

EXAMPLE 9.1

q	$q^\star$ $x = 0$	$x = 1$	z $x = 0$	$x = 1$
A	C	B	0	0
B	D	D	0	0
C	A	D	0	1
D	A	C	0	1

States C and D are equivalent. They both have a 0 output for $x = 0$, and a 1 output for $x = 1$. Both go to A when $x = 0$, and they go to either C or D when $x = 1$. We could say that C and D are equivalent if D is equivalent to C; but that is a truism. Thus, this system can be reduced to three states:

q	$q^\star$ $x = 0$	$x = 1$	z $x = 0$	$x = 1$
A	C-D	B	0	0
B	C-D	C-D	0	0
C-D	A	C-D	0	1

where we have named the state resulting from the equivalence of *C* and *D* by a compound name *C*-*D*. (We will do that some of the time, but it often gets cumbersome, and we will name the state in the reduced system using the name of the first state in the group.)

More commonly, equivalences are not so obvious. Therefore, we will develop two algorithmic methods in the next two sections.

9.1 A TABULAR METHOD FOR STATE REDUCTION

In this section, we will develop a technique using a chart with one square for each possible pairing of states. We will enter in that square an X if those states cannot be equivalent because the outputs are different, a √ if the states are equivalent (because they have the same output and go to the same state or to each other for each input), and otherwise the conditions that must be met for those two states to be equivalent (that is, which states must be equivalent to make these equivalent).

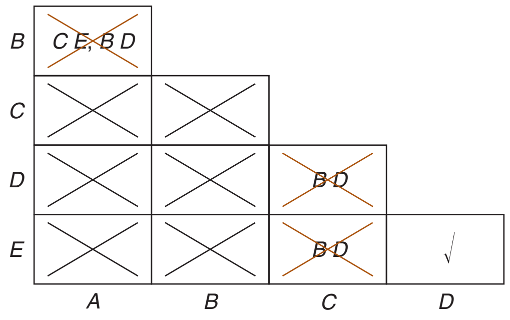
The chart has one row for each state except the first and one column for each state except the last; only the lower half of the chart is necessary to include all pairs of states. For the state table of Table 9.1, we first get the chart of Figure 9.1.

Figure 9.1 Chart for Table 9.1.

B	*C E*, *B D*			
C	X	X		
D	X	X	*B D*	
E	X	X	*B D*	√
	A	*B*	*C*	*D*

In order for states *A* and *B* to be equivalent, they must have the same output for both $x = 0$ and $x = 1$ (which they do) and must go to equivalent states. Thus, *C* must be equivalent to *E* and *B* must be equivalent to *D*, as shown in the first square. Each square in the balance of that column and the whole next column contains an X since states *A* and *B* have a 0 output for $x = 1$ and states *C*, *D*, and *E* have a 1 output. In the *CD* square, we place *BD* since *C* goes to *D* and *D* goes to *B* when $x = 1$. That is also the case in the *CE* square. Finally, in the *DE* square, we place a check (√), since both states have the same output and next state for each input. We must now go back through the table to see if the conditions are met. Since *B* cannot be equivalent to *D* (there is already an X in the *BD* square), none of the three pairs can be equivalent. We thus cross out those squares, leaving only one check, as shown in Figure 9.2.

Figure 9.2 Reduced chart with states crossed out.



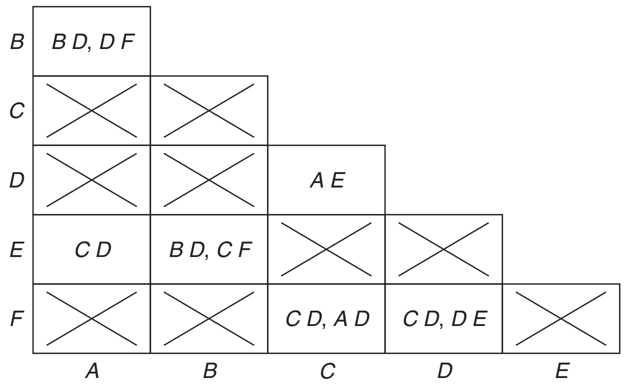
The states *D* and *E* can be combined. The reduced table can then be produced. In the process of doing that, there is a check that there was no mistake in the first part. As entries are made for combined states (such as *D*-*E*), each of the original states must go to the same state in the reduced table and they must have the same output. The reduced table was shown in the last section (Table 9.2).

The process is not always as easy as this. Example 9.2 will illustrate some further steps that are necessary.

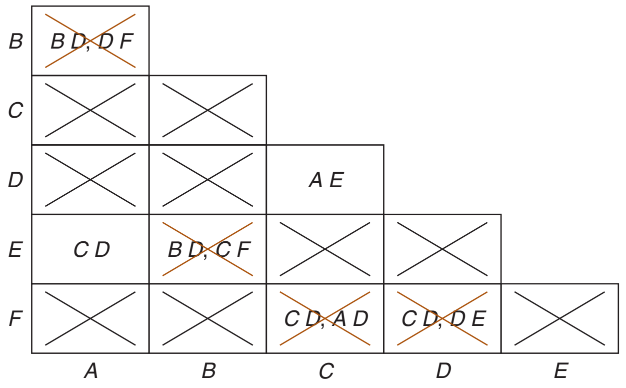
EXAMPLE 9.2

q	$q^{\star}$ $x = 0$	$x = 1$	z
A	*B*	*D*	1
B	*D*	*F*	1
C	*D*	*A*	0
D	*D*	*E*	0
E	*B*	*C*	1
F	*C*	*D*	0

The chart for this table is



Note that the first entry could have been written *BDF*, that is, all three states must be equivalent. (Indeed, since this is the condition for *A* and *B* to be equivalent, we then require that *A*, *B*, *D*, and *F* all be equivalent.) Going through the table, we see that *B* and *D* cannot be equivalent; neither can *A* and *D*, nor *D* and *E*. That reduces the table to the following:



What remains is that *A* is equivalent to *E* if *C* is equivalent to *D* and that *C* is equivalent to *D* if *A* is equivalent to *E*. That allows us to check off both of these, producing the reduced table:

	$q^\star$		
q	$x = 0$	$x = 1$	z
A-E	*B*	*C-D*	1
B	*C-D*	*F*	1
C-D	*C-D*	*A-E*	0
F	*C-D*	*C-D*	0

Before looking at some more complex examples, we want to emphasize the effect that the output column has on the process. The state table and chart of Example 9.3 correspond to a system with the same next state behavior as that of Example 9.2, but a different output column.

EXAMPLE 9.3

	$q^\star$		
q	$x = 0$	$x = 1$	z
A	*B*	*D*	1
B	*D*	*F*	1
C	*D*	*A*	1
D	*D*	*E*	0
E	*B*	*C*	0
F	*C*	*D*	0

The chart is different, because the pairings that are automatically X'd (due to the output) are different.

B	*B D F*				
C	*A B D*	*A F*			
D	✗	✗	✗		
E	✗	✗	✗	*B D, C E*	
F	✗	✗	✗	*C D E*	*B C D*
	A	*B*	*C*	*D*	*E*

None of the conditions can be satisfied, and thus, no states can be combined and the state table cannot be reduced.

Sometimes, when the obvious unequivalences are crossed off, there may be some doubt as to whether the remaining states can be combined. We could try to combine them all and develop the reduced state table. If we were mistaken, it will quickly become evident. Also, if we find one or more equivalences before completing the process, we can reduce the table and start the process over. That is somewhat more work, but the chart for the reduced table is much smaller and may be easier to work with.

EXAMPLE 9.4

	$q^\star$		z	
q	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A	*F*	*B*	0	0
B	*E*	*G*	0	0
C	*C*	*G*	0	0
D	*A*	*C*	1	1
E	*E*	*D*	0	0
F	*A*	*B*	0	0
G	*F*	*C*	1	1

The chart for this table is

B	*E F, B G*					
C	*B G, C F*	*C E*				
D	✗	✗	✗			
E	*E F, B D*	*D G*	*D G*	✗		
F	√	*A E, B G*	*A C, B G*	✗	*A E, B D*	
G	✗	✗	✗	*A F*	✗	✗
	A	*B*	*C*	*D*	*E*	*F*

Note that we checked *AF*, since the requirement that *A* is equivalent to *F* is just that *F* is equivalent to *A*. That is always true. We first go through the chart to find which conditions cannot be met, crossing them out. We also note that condition *AF* has already been checked and therefore *D* and *G* are equivalent. During this pass, we may take advantage of the new equivalences and the cross outs or we may wait until the next pass. Waiting until the next pass, the table becomes

B	~~*E F*, *B G*~~					
C	~~*B G*, *C F*~~	*C E*				
D	✗	✗	✗			
E	~~*E F*, *B D*~~	*D G*	*D G*	✗		
F	√	~~*A E*, *B G*~~	~~*A C*, *B G*~~	✗	~~*A E*, *B D*~~	
G	✗	✗	✗	√ *A F*	✗	✗
	A	*B*	*C*	*D*	*E*	*F*

We now have *A* equivalent to *F* and *D* equivalent to *G*. The latter satisfies the condition for *C* being equivalent to *E* and *B* being equivalent to *E*. Finally, since *C* and *E* are equivalent, *B* is equivalent to *C*. That makes *B*, *C*, and *E* all equivalent. Thus, the reduced table has three states—*A* (*A*-*F*), *B* (*B*-*C*-*E*), and *D* (*D*-*G*).

	$q^\star$		z	
q	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A	*A*	*B*	0	0
B	*B*	*D*	0	0
D	*A*	*B*	1	1

With the last chart above, we could have reduced the system to one with five states, just using the equivalences checked (*A*-*F* and *D*-*G*). That would produce the new table

	$q^\star$		z	
q	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A	*A*	*B*	0	0
B	*E*	*D*	0	0
C	*C*	*D*	0	0
D	*A*	*C*	1	1
E	*E*	*D*	0	0

We can now construct a new chart

B	*A E*, *B D*			
C	*B D*	*C E*		
D	✗	✗	✗	
E	*B D*	√	√	✗
	A	*B*	*C*	*D*

However, *B* and *D* cannot be equivalent; *C* and *E* are. Thus, states *B*, *C*, and *E* can be replaced by one state, *B*; that will allow us to reduce this table to the three-state one we have already obtained.

EXAMPLE 9.5

	$q^\star$		z	
q	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A	*B*	*D*	0	0
B	*E*	*D*	1	0
C	*B*	*C*	0	0
D	*F*	*A*	0	0
E	*A*	*B*	1	1
F	*E*	*C*	1	0

We first construct the chart

B	✗				
C	*C D*	✗			
D	*B F*	✗	*B F*, *A C*		
E	✗	✗	✗	✗	
F	✗	*C D*	✗	✗	✗
	A	*B*	*C*	*D*	*E*

None of the conditions in the squares are contradicted. The chart says that for *A* to be equivalent to *C*, *C* must be equivalent to *D*. That would make *A*, *C*, and *D* one group. For *A* to be equivalent to *D*, *B* must be equivalent to *F*, and for *B* to be equivalent to *F*, *C* must be equivalent to *D*. Finally, *C* is equivalent to *D* if *B* is equivalent to *F* and *A* is equivalent to *C*. All of these can be true, producing a reduced state table with only three states, *A* (*A*-*C*-*D*), *B* (*B*-*F*), and *E*. (As we construct that table, we can check our conclusions from the chart by making sure that all states in one group go to a state in a single group for each input.)

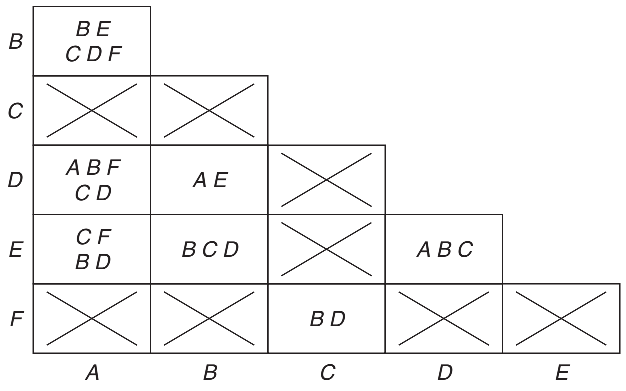
	$q^\star$		z	
q	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A	*B*	*A*	0	0
B	*E*	*A*	1	0
E	*A*	*B*	1	1

EXAMPLE 9.6

As our last example of this technique, we will consider a system with two inputs, *x y*. Thus, there are four columns in the next state section of the table.

	$q^\star$				
q	$x\,y$ 0 0	0 1	1 0	1 1	z
A	*B*	*A*	*F*	*D*	1
B	*E*	*A*	*D*	*C*	1
C	*A*	*F*	*D*	*C*	0
D	*A*	*A*	*B*	*C*	1
E	*B*	*A*	*C*	*B*	1
F	*A*	*F*	*B*	*C*	0

The charting problem is really no different than before; we just have more conditions, since equivalent states must go to equivalent states for all four input combinations (that is, all four columns). The chart then becomes



None of the groups of three states shown in the chart can be equivalent; one of each has a different output than the other two. Crossing out those squares, we have

B	~~*B E* *C D F*~~				
C	✗	✗			
D	~~*A B F* *C D*~~	*A E*	✗		
E	*C F* *B D*	~~*B C D*~~	✗	~~*A B C*~~	
F	✗	✗	*B D*	✗	✗
	A	*B*	*C*	*D*	*E*

That leaves three pairings intact, *A E*, *B D*, and *C F*. The only requirement for any of these is that one of the others be equivalent. Thus, we can reduce this to three states as follows:

	x y		***q****		
q	**0 0**	**0 1**	**1 0**	**1 1**	***z***
A	*B*	*A*	*C*	*B*	1
B	*A*	*A*	*B*	*C*	1
C	*A*	*C*	*B*	*C*	0

[SP 1; EX 1]

9.2 PARTITIONS

A *partition* on the states of a system is a grouping of the states of that system into one or more blocks. Each state must be in one and only one block. For a system with four states, *A*, *B*, *C*, and *D*, the complete list of partitions is

$P_0 = (A)(B)(C)(D)$ $\quad P_8 = (AC)(BD)$
$P_1 = (AB)(C)(D)$ $\quad P_9 = (AD)(BC)$
$P_2 = (AC)(B)(D)$ $\quad P_{10} = (ABC)(D)$
$P_3 = (AD)(B)(C)$ $\quad P_{11} = (ABD)(C)$
$P_4 = (A)(BC)(D)$ $\quad P_{12} = (ACD)(B)$
$P_5 = (A)(BD)(C)$ $\quad P_{13} = (A)(BCD)$
$P_6 = (A)(B)(CD)$ $\quad P_N = (ABCD)$
$P_7 = (AB)(CD)$

This list of partitions does not depend on the details of the state table, only on the list of states. P_0 is the partition with each state in a separate block; P_N is the partition with all of the states in the same block. We will be concerned with partitions that have special properties for a particular state table. There are three categories of partitions that will be of interest. To illustrate these, we will use Table 9.3.

Any partition with two blocks can be used to assign one of the state variables. Those states in the first block would be assigned 0 and those in the second block 1 (or vice versa). P_7 through P_{13} meet that requirement. (We write partitions in alphabetic order; thus, state A will usually be assigned all 0's.) In a four-state system, there are only three pairs of partitions that can be used for a two-variable state assignment, P_7 and P_8, P_7 and P_9, and P_8 and P_9. The three are shown in Table 9.4.

Table 9.3 State table to illustrate types of partitions.

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	A	1
B	D	B	0
C	A	B	1
D	B	A	0

Table 9.4 State assignments for four states.

(a)

q	q_1	q_2
A	0	0
B	0	1
C	1	0
D	1	1
	P_7	P_8

(b)

q	q_1	q_2
A	0	0
B	0	1
C	1	1
D	1	0
	P_7	P_9

(c)

q	q_1	q_2
A	0	0
B	1	1
C	0	1
D	1	0
	P_8	P_9

If we try any other pair of two-block partitions, we do not have an adequate state assignment. For example, using P_8 and P_{11}, we get the assignment of Table 9.5. Note that states B and D have the same assignment.

Table 9.5 An unsuccessful assignment.

q	q_1	q_2
A	0	0
B	1	0
C	0	1
D	1	0
	P_8	P_{11}

A second useful class of partitions are those for which all of the states in each block have the same output for each of the inputs. Such partitions are referred to as *output consistent*. P_0 is always output consistent; for Table 9.3, the other output consistent partitions are

$$P_2 = (AC)(B)(D)$$
$$P_5 = (A)(BD)(C)$$
$$P_8 = (AC)(BD)$$

Knowing the block of an output consistent partition and the input is enough information to determine the output (without having to know which state within a block).

For some partitions, knowing the block of the partition and the input is enough information to determine the block of the next state. Such a partition is said to have the *substitution property* and is referred to as an *SP partition*. P_N is always SP since all states are in the same block, and P_0 is always SP since knowing the block is the same as knowing the state. Others may be SP, depending on the details of the state table. For this state table, there are two nontrivial SP partitions (those other than P_0 and P_N), namely,

$$P_7 = (AB)(CD)$$
$$P_9 = (AD)(BC)$$

If a partition other than P_0 is both SP and output consistent, then we can reduce the system to one having just one state for each block of that partition. (That should be obvious since knowing the input and the block of the partition is all we need to know to determine the output, since it is output consistent, and to determine the next state, since it is SP). For this example, neither of the SP partitions is also output consistent. In Example 9.1, the partition

$$(A)(B)(CD)$$

is both SP and output consistent; thus, we were able to reduce the system to one with only three states.

Before developing a method for finding all SP partitions, we will look at Example 9.7. It will help us understand the application of the various categories of partitions.

EXAMPLE 9.7

q	$q^\star$ $x = 0$	$q^\star$ $x = 1$	z
A	C	D	0
B	C	E	1
C	A	D	1
D	B	E	1
E	B	E	0

Assignment 1

q	q_1	q_2	q_3
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0

For this state assignment, we obtain the output equation and the inputs to D flip flops

$$z = q_2 + q_3$$
$$D_1 = xq_1 + xq_3$$
$$D_2 = x'q_1'q_2' + xq_1'q_3'$$
$$D_3 = x'q_2q_3 + x'q_1 + xq_1'q_3'$$

This requires 11 gates and 25-gate inputs (for either AND and OR or NAND, including a NOT for x'). We would need four 7400 series integrated circuit packages to implement the combinational logic with NAND gates. (The development of the equations and the gate count is left as an exercise.)

If we make the state assignment using the following three partitions:

$P_1 = (ABC)(DE)$	SP
$P_2 = (AB)(CDE)$	SP
$P_3 = (AE)(BCD)$	output consistent

we have

Assignment 2

q	q_1	q_2	q_3
A	0	0	0
B	0	0	1
C	0	1	1
D	1	1	1
E	1	1	0

Logic Equations

$$z = q_3$$
$$D_1 = x$$
$$D_2 = x + q_2'$$
$$D_3 = x'q_1 + x'q_2' + \{q_2'q_3' \text{ or } q_1'q_3'\} + xq_1'q_2$$

This requires only seven gates, 16 inputs, and three 7400 series NAND gate integrated circuit packages. D_3 is the only complex function.

We saw the advantage of using an output consistent partition, that the output is just equal to that variable or its complement. Thus, z is only a function of q_3.

We saw the advantage of using an SP partition to assign a state variable, that the next state of that variable is only a function of the input, x, and that variable (no matter how many flip flops are needed to implement that system). Since q_1 and q_2 are assigned using SP partitions, D_1 is only a function of x (could also depend on q_1), and D_2 is only a function of x and q_2. For JK flip flops, the inputs are functions only of x (x, x', and 1 are the only possibilities). Indeed, we do not need to deal with the whole state table for the implementation of flip flops assigned according to SP partitions; we only need a block table. Thus, for the flip flops assigned according to partition $P_1 = (ABC)(DE)$, where the first block is assigned 0, we would have Table 9.6, which gives

$$D_1 = q_1^\star = x \qquad J_1 = x \qquad K_1 = x'$$

Table 9.6 Truth table for q_1.

	x	q_1	$q_1^\star$	J	K
(ABC)	0	0	0	0	X
(DE)	0	1	0	X	1
(ABC)	1	0	1	1	X
(DE)	1	1	1	X	0

This is, of course, the same answer we got for D_1 before. We could follow this approach for D_2 or J_2 and K_2 as well, but would need the full four-variable truth table to solve for the inputs to q_3.

9.2.1 Properties of Partitions

First, we will define, for pairs of partitions, the relationship *greater than or equal* ($\geq$) and two operators, the product and the sum.

- $P_a \geq P_b$ if and only if all states in the same block of P_b are also in the same block of P_a.

For example,

$$P_{10} = (ABC)(D) \geq P_2 = (AC)(B)(D)$$

since the only states in the same block of P_2 (A and C) are also in the same block of P_{10}. P_0 is the smallest partition; all other partitions are greater than it. P_N is the largest partition; it is greater than all others. Not all partitions are ordered. For example, P_1 is neither $\geq$ nor $\leq P_2$.

- The *product* of two partitions is written $P_c = P_a P_b$.

  Two states are in the same block of the product P_c if and only if they are in the same block of both P_a and P_b.

For example,

$$P_{12}P_{13} = \{(ACD)(B)\}\{(A)(BCD)\} = (A)(B)(CD) = P_6$$

The only states that are in the same block of both P_{12} and P_{13} are C and D; they are then together in the product. The partitions P_a and P_b are always greater than or equal to the product P_c. If the two partitions are ordered, the product is equal to the smaller one. For example, $P_6 < P_{13}$ and thus, $P_6P_{13} = P_6$. It is also clear from the definitions that, for any partition, P_a

$$P_aP_0 = P_0 \quad \text{and} \quad P_aP_N = P_a$$

■ The *sum* of two partitions is written $P_c = P_a + P_b$.

Two states are in the same block of the sum P_d if they are in the same block of either P_a or P_b or both.

For example,

$$P_2 + P_5 = \{(AC)(B)(D)\} + \{(A)(BD)(C)\} = P_8 = (AC)(BD)$$

The sum sometimes brings together states that are not in the same block of either since whole blocks are combined. Consider the following example:

$$P_a = (AB)(C)(DF)(EG)$$
$$P_b = (ACD)(BG)(E)(F)$$
$$P_a + P_b = (ABCDEFG) = P_N$$

Since A and B are in the same block of P_a and A, C, and D are in the same block of P_b, then $ABCD$ are in one block of the sum. But F is in the same block as D in P_a and G is in the same block as B of P_b; so they must be included with $ABCD$. Finally, E is in the same block as G in P_a, producing a sum of P_N. The sum P_c is always greater than or equal to both P_a and P_b. If P_a and P_b are ordered, the sum equals the greater. Thus, $P_6 + P_{13} = P_{13}$. Also,

$$P_a + P_0 = P_a \quad \text{and} \quad P_a + P_N = P_N$$

9.2.2 Finding SP Partitions

The process of finding all SP partitions has two steps.

Step 1: For each pair of states, find the smallest SP partition that puts those two states in the same block.

We must ask what is required to make a partition SP if these two states are in the same block, that is, what makes these two states equivalent. They must go to equivalent states for each input. We must then follow through, determining what groupings are forced.

We will use the state table of Table 9.3, repeated here without the output columns (since that has no relevance to finding SP partitions) as Table 9.7.

Table 9.7 A state table for finding SP partitions.

q	$q^\star$ $x = 0$	$x = 1$
A	C	A
B	D	B
C	A	B
D	B	A

For A to be equivalent to B, C must be equivalent to D. We continue by checking what conditions are required to make C equivalent to D. In this example, the only requirement is that A be equivalent to B. Thus, we have our first SP partition

$$(AB) \rightarrow (CD) \qquad \rightarrow\rightarrow (AB)(CD) = P_1$$

where the right arrow ($\rightarrow$) is used to indicate requires, and the double arrow indicates the smallest SP partition that results. Sometimes, we find no new conditions and other times the conditions force all of the states into one block, producing P_N.

The next step is

$$(AC) \rightarrow (AB) \rightarrow (CD) \rightarrow (ABCD) = P_N$$

(Since C must be with A and B must be with A, then A, B, and C must all be together. But then D must be with C, resulting in P_N.) The balance of step 1 produces

$$\begin{aligned}
&(AD) \rightarrow (BC) && \rightarrow\rightarrow (AD)(BC) = P_2\\
&(BC) \rightarrow (AD) && \rightarrow\rightarrow (AD)(BC) = P_2\\
&(BD) \rightarrow (AB) && \rightarrow\rightarrow P_N\\
&(CD) \rightarrow (AB) && \rightarrow\rightarrow (AB)(CD) = P_1
\end{aligned}$$

Step 2: Find the sum of all of the SP partitions found in step 1 and, if new ones are found, repeat step 2 on these new ones.

In this process, we do not need to find the sum of another partition with any two-block partition since that always results in either the two-block partition or P_N. Also, if one partition is greater than another, its sum is always the greater partition. We can omit those additions, too.

For the first example, there are no sums to compute, since the only two unique nontrivial (that is, other than P_0 and P_N) SP partitions formed by step 1 are both two-block.

EXAMPLE 9.8

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	D	1
B	C	D	0
C	B	D	1
D	C	A	1

Step 1 produces five SP partitions.

$(AB) \rightarrow \surd$[*]	$\rightarrow \rightarrow P_1 = (AB)(C)(D)$
$(AC) \rightarrow (BC)$, $(BC) \rightarrow$ ok	$\rightarrow \rightarrow P_2 = (ABC)(D)$
$(AD) \rightarrow \surd$	$\rightarrow \rightarrow P_3 = (AD)(B)(C)$
$(BC) \rightarrow \surd$	$\rightarrow \rightarrow P_4 = (A)(BC)(D)$
$(BD) \rightarrow (AD) \rightarrow (ABD)$	$\rightarrow \rightarrow P_5 = (ABD)(C)$
$(CD) \rightarrow (BC), (AD)$	$\rightarrow \rightarrow P_N$

Step 2 really only requires three sums, although we will show all 10 below:

$P_1 + \mathbf{P_2} = (ABC)(D)$	$\rightarrow \rightarrow P_2$	not needed
$P_1 + P_3 = (ABD)(C)$	$\rightarrow \rightarrow P_5$	
$P_1 + P_4 = (ABC)(D)$	$\rightarrow \rightarrow P_2$	
$P_1 + \mathbf{P_5} = (ABD)(C)$	$\rightarrow \rightarrow P_5$	not needed
$\mathbf{P_2} + P_3$	$\rightarrow \rightarrow P_N$	not needed
$\mathbf{P_2} + P_4 = (ABC)(D)$	$\rightarrow \rightarrow P_2$	not needed
$\mathbf{P_2} + \mathbf{P_5}$	$\rightarrow \rightarrow P_N$	not needed
$P_3 + P_4 = (AD)(BC)$	$\rightarrow \rightarrow P_6 = (AD)(BC)$	
$P_3 + \mathbf{P_5} = (ABD)(C)$	$\rightarrow \rightarrow P_5$	not needed
$P_4 + \mathbf{P_5}$	$\rightarrow \rightarrow P_N$	not needed

Those partitions shown in bold are two-block and thus never produce anything new. Only one new SP partition is found by step 2.

EXAMPLE 9.9

	$q^\star$		
q	$x = 0$	$x = 1$	z
A	C	D	0
B	D	A	0
C	E	D	0
D	B	A	1
E	C	D	1

Step 1 of the process produces five SP partitions, as follows:

$(AB) \rightarrow (CD)(AD) \rightarrow (ACD) \rightarrow (BCE)$	$\rightarrow \rightarrow P_N$
$(AC) \rightarrow (CE)$	$\rightarrow \rightarrow (ACE)(B)(D) = P_1$
$(AD) \rightarrow (BC) \rightarrow (DE)$	$\rightarrow \rightarrow (ADE)(BC) = P_2$
$(AE) \rightarrow \surd$	$\rightarrow \rightarrow (AE)(B)(C)(D) = P_3$
$(BC) \rightarrow (ADE)$	$\rightarrow \rightarrow P_2$
$(BD) \rightarrow \surd$	$\rightarrow \rightarrow (A)(BD)(C)(E) = P_4$
$(BE) \rightarrow (ACD) \rightarrow (BCE)$	$\rightarrow \rightarrow P_N$
$(CD) \rightarrow (BE)(AD) \rightarrow (BC)$	$\rightarrow \rightarrow P_N$
$(CE) \rightarrow \surd$	$\rightarrow \rightarrow (A)(B)(CE)(D) = P_5$
$(DE) \rightarrow (BC)(AD) \rightarrow (ADE)$	$\rightarrow \rightarrow P_2$

*Here is the first example of a pairing that requires no other states to be combined. It results in a partition where these two states are in one block and all others are by themselves.

For step 2, we add each of the pairs of partitions found in step 1, except that we do not need to add P_2 to anything (since it is two-block) and P_1 need not be added to P_3 or P_5 (since it is greater than each of them).

$$P_1 + P_4 = (ACE)(BD) \quad = P_6$$
$$P_3 + P_4 = (AE)(BD)(C) \quad = P_7$$
$$P_3 + P_5 = (ACE)(B)(D) \quad = P_1$$
$$P_4 + P_5 = (A)(BD)(CE) \quad = P_8$$

We now add pairs of these new partitions (with the same exceptions as above); there is only one sum (which does not produce anything new):

$$P_7 + P_8 = (ACE)(BD) \quad = P_6$$

If there were new partitions of more than two blocks, they must also be added.

For this example, there are eight nontrivial SP partitions, of which two are two-block and none are output-consistent. We will return to this state table in the next sections when we discuss state reduction and how to make good state assignments.

[SP 2, EX 2]

9.3 STATE REDUCTION USING PARTITIONS

Any partition that is both output consistent and SP can be used to reduce the system to one with one state for each block of that partition. Just as there is always a unique largest SP partition (P_N), there is always a unique largest output consistent SP partition. That is the one with the fewest blocks and thus corresponds to the reduced system with the fewest number of states.*

For the state table of Example 9.8, repeated here as Table 9.8, the only SP partition that is output consistent is $P_3 = (AD)(B)(C)$; thus, this state table can be reduced to one with three states (one for each block of P_3). We will call the combined state A (rather than A-D); the reduced table is shown in Table 9.9.

Table 9.8 A reducible state table.

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	D	1
B	C	D	0
C	B	D	1
D	C	A	1

Table 9.9 Reduced state table.

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	A	1
B	C	A	0
C	B	A	1

We do not need to recalculate all of the SP partitions (although for this small example, that would be very easy). Any SP partition of the original system that is greater than (>) the one used to reduce the system is still SP. For this example, only $P_5 \geq P_3$. Thus, we get one nontrivial SP partition for the reduced system, namely,

$$P_5^\star = (AB)(C)$$

where AD of the original P_5 has been replaced by the new state A.

The last state table of the previous section, Example 9.9, did not have any output consistent SP partitions. Thus, it can not be reduced.

*It is possible that P_N is output consistent; but that is a combinational system, where the output does not depend on the state.

We will now look at two state tables with the same next state section, but different output columns.

EXAMPLE 9.10

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	D	0
B	D	A	1
C	E	D	0
D	B	A	0
E	C	D	0

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	D	0
B	D	A	1
C	E	D	0
D	B	A	1
E	C	D	0

The set of SP partitions for these two is the same as those for Example 9.9, since the substitution property does not depend on the output. Repeating the complete list here, we have

$$P_1 = (ACE)(B)(D)$$
$$P_2 = (ADE)(BC)$$
$$P_3 = (AE)(B)(C)(D)$$
$$P_4 = (A)(BD)(C)(E)$$
$$P_5 = (A)(B)(CE)(D)$$
$$P_6 = (ACE)(BD)$$
$$P_7 = (AE)(BD)(C)$$
$$P_8 = (A)(BD)(CE)$$

In the first table, P_1, P_3, and P_5 are the only output consistent partitions. Since

$$P_1 = (ACE)(B)(D)$$

is greater than either of the others, we will use it to reduce the system to one with three states, as follows:

q	$q^\star$ $x = 0$	$x = 1$	z
A-C-E	A-C-E	D	0
B	D	A-C-E	1
D	B	A-C-E	0

(We labeled the combined state with a compound name; we could have just called it A.) Note that only $P_6 > P_1$; thus, the only SP partition of the reduced system is

$$P_6^\star = (A\text{-}C\text{-}E)(BD)$$

For the second state table, P_1, P_3, P_4, P_5, P_6, and P_8 are all output consistent. The largest is

$$P_6 = (ACE)(BD)$$

as can be seen from the chart below, where the smaller ones are on the left.

$$\begin{tikzcd}
(A\,E)\,(B)\,(C)\,(D) \arrow[dr, dash, "\leq"] & & \\
& (A\,C\,E)\,(B)\,(D) \arrow[dr, dash, "\leq"] & \\
(A)\,(B)\,(C\,E)\,(D) \arrow[ur, dash, "\leq"] \arrow[dr, dash, "\leq"'] & & (A\,C\,E)\,(B\,D) \\
& (A)\,(B\,D)\,(C\,E) \arrow[ur, dash, "\leq"'] & \\
(A)\,(B\,D)\,(C)\,(E) \arrow[ur, dash, "\leq"] & &
\end{tikzcd}$$

We can thus reduce the system to one with only two states, *A* (*ACE*) and *B* (*BD*), as shown below. This system requires only one flip flop.

	$q^\star$		
q	***x* = 0**	***x* = 1**	***z***
A	*A*	*B*	0
B	*B*	*A*	1

The computation of all of the SP partitions for a fairly large system can be quite time-consuming. If our interest is in reducing the system to one with the minimum number of states, we can do that immediately when we find an output consistent SP partition. Consider the following example.

EXAMPLE 9.11

	$q^\star$		
q	***x* = 0**	***x* = 1**	***z***
A	*B*	*E*	0
B	*D*	*A*	1
C	*G*	*A*	0
D	*F*	*G*	1
E	*B*	*C*	0
F	*D*	*G*	1
G	*D*	*E*	1

As we begin the process of finding SP partitions, we get

$$(AB) \to (BD)(AE) \to (DF)(AG)(CE) \quad \to\to P_N$$
$$(AC) \to (BG)(AE) \to (ACE) \quad \to\to (ACE)(BG)(D)(F)$$

This SP partition is also output consistent. Therefore, we could stop and reduce the system to one with four states (one for each block) and find the SP partitions of that smaller system.

	$q^\star$		
q	***x* = 0**	***x* = 1**	***z***
A	*B*	*A*	0
B	*D*	*A*	1
D	*F*	*B*	1
F	*D*	*B*	1

We can now find the SP partitions of this smaller system

$(AB) \to (BD) \to (DF)$	$\to \to P_N$
$(AD) \to (AB)$*(BF)	$\to \to P_N$
$(AF) \to (BD)(AB)$	$\to \to P_N$
$(BD) \to (DF)(AB)$	$\to \to P_N$
$(BF) \to (AB)$	$\to \to P_N$
$(DF) \to \surd$	$\to (A)(B)(DF)$

This system can be reduced further, to one with three states, since the SP partition is also output consistent. The smallest equivalent system is thus

q	$q^\star$ $x = 0$	$x = 1$	z
A	B	A	0
B	D	A	1
D	D	B	1

As a final example, consider the following state table, where five different output columns are shown. (This is a Moore system with an output that does not depend on the input; we will consider the different outputs as five different problems.)

EXAMPLE 9.12

q	$q^\star$ $x = 0$	$x = 1$	z_1	z_2	z_3	z_4	z_5
A	D	B	0	0	0	1	1
B	E	C	0	0	1	0	1
C	A	B	1	1	0	0	1
D	E	C	1	1	1	1	1
E	D	B	1	0	0	1	1

We will start by finding all of the SP partitions. That, of course, does not depend upon which output column is used.

$(AB) \to (BC)(DE) \to (AE)$	$\to \to P_N$
$(AC) \to (AD) \to (AE)\ (BC)$	$\to \to P_N$
$(AD) \to (DE)(BC)$	$\to \to (ADE)(BC) = P_1$
$(AE) \to \surd$	$\to \to (AE)(B)(C)(D) = P_2$
$(BC) \to (AE)$	$\to \to (AE)(BC)(D) = P_3$
$(BD) \to \surd$	$\to \to (A)(BD)(C)(E) = P_4$
$(BE) \to (BC)(DE) \to (AE)$	$\to \to P_N$
$(CD) \to (AE)(BC)$	$\to \to (AE)(BCD) = P_5$
$(CE) \to (AD) \to (DE)(BC)$	$\to \to P_N$
$(DE) \to (BC) \to (AE)$	$\to \to P_1$

*Since we found that the only SP partition that combines A and B is P_N, we can stop looking; this must also produce P_N.

Now, forming sums, we obtain only one new partition

$P_2 + P_4 = (AE)(BD)(C) = P_6$

Thus, there are six nontrivial SP partitions.

For the first output column, none of the SP partitions are output consistent. Thus, the state table cannot be reduced. (We will return to this example in the next section and determine a good state assignment.)

For the second output column, only P_2 is output consistent. Thus, this system can be reduced to one with four states (replacing A and E by a state called A).

	$q^\star$		
q	$x = 0$	$x = 1$	z_2
A	D	B	0
B	A	C	0
C	A	B	1
D	A	C	1

Since any SP partition that is greater than P_2 is an SP partition of the reduced table (with states A and E shown as one, just A), we can see that the SP partitions are

$P_1^\star = (AD)(BC)$
$P_3^\star = (A)(BC)(D)$
$P_5^\star = (A)(BCD)$
$P_6^\star = (A)(BD)(C)$

For the third output column, P_2, P_4, and P_6 are all output consistent. Since P_6 is the largest of these, it is used to reduce the system to one with only three states.

	$q^\star$		
q	$x = 0$	$x = 1$	z_3
A	B	B	0
B	A	C	0
C	A	B	1

The only nontrivial SP partition for this system is

$P_5^\star = (A)(BC)$

For the fourth output column, P_1 is output consistent, reducing the state table to one with only two states.

q	$q^\star$ $x = 0$	$x = 1$	z_4
A	A	B	0
B	A	B	1

Finally, for the last output column, there was no need to find the SP partitions; P_N is output consistent; the system is combinational. It does not depend on the state.

$z = 1$

[SP 3, EX 3]

9.4 CHOOSING A STATE ASSIGNMENT

In this section, we will look at a strategy for making a "good" state assignment. We will first find all of the SP partitions, then reduce the system if possible, and finally make a state assignment to solve the problem. We have three levels of work, depending upon how important it is to reduce the cost of the combinational logic. If an absolute minimum is required, we must try all possible sets of two-block partitions for which the product is P_0. For three or four states, there are only three such assignments, and it is fairly easy to do that. For five states, however, that number goes up to 140, and this method is not practical. (It rises to 420 for six states, to 840 for seven or eight states, and to over 10 million for nine states.) If we can use two-block SP partitions for one or more of the variables, that is almost always preferable (as long as we do not increase the number of variables). We can then try to group states that are in the same block of multiblock SP partitions or to use partitions that correspond to one or more of the output columns for the other variables. This will usually lead to a pretty good solution. Last, and least likely to produce good results, we could choose an assignment at random, say using 000 for A, 001 for B, and so forth. Sometimes, that will lead to a good solution. But more often, it will result in a more costly system.

To illustrate this, we will consider the example in Table 9.10. The SP partitions are

Table 9.10 State assignment example.

q	$q^\star$ $x = 0$	$x = 1$	z
A	B	C	0
B	A	D	1
C	A	D	0
D	B	C	1

$$P_1 = (AB)(CD)$$
$$P_2 = (AD)(B)(C)$$
$$P_3 = (A)(BC)(D)$$
$$P_2 + P_3 = P_4 = (AD)(BC)$$

There are no output consistent SP partitions. There are two SP partitions that could be used for state assignment, namely, P_1 and P_4. That would produce the assignment of Table 9.11 and the D flip flop input equations shown.

Table 9.11 State assignment.

q	q_1	q_2
A	0	0
B	0	1
C	1	1
D	1	0

$$z = q_1'q_2 + q_1q_2'$$
$$D_1 = x$$
$$D_2 = q_2'$$

Since both flip flops were assigned according to an SP partition, the input equations are very simple.

If we repeat the design, using the output consistent partition for q_2, we get the state assignment of Table 9.12 and the equations shown beside it.

Table 9.12 State assignment.

q	q_1	q_2
A	0	0
B	0	1
C	1	0
D	1	1

$$z = q_2'$$
$$D_1 = x$$
$$D_2 = x'q_1'q_2' + x'q_1q_2 + xq_1'q_2 + xq_1q_2'$$

Notice that D_1 is unchanged. Since it was assigned according to the same SP partition as before, its behavior does not depend on the rest of the assignment. Also, z becomes simple, since q_2 is assigned according to an output consistent partition. This is an extreme case; D_2 is particularly complex. If, on the other hand, we assigned q_1 according to the output consistent partition and q_2 according to P_4 (as in the first example), we would get the assignment of Table 9.13 and the equations shown below.

Table 9.13 State assignment.

q	q_1	q_2
A	0	0
B	1	1
C	0	1
D	1	0

$$z = q_1$$
$$D_1 = x'q_2' + xq_2$$
$$D_2 = q_2'$$

Now, D_1 is more complex, although the total cost of combinational logic is the same as for the first assignment. Costs do not vary as much in two flip flop circuits as they do in larger ones.

We will illustrate the procedure with two of the output columns from Example 9.12.

EXAMPLE 9.13

We will first consider the table of Example 9.12 with output column z_2.

q	$q^\star$ $x = 0$	$x = 1$	z_2
A	D	B	0
B	E	C	0
C	A	B	1
D	E	C	1
E	D	B	0

The first step is to see if the system can be reduced. The SP partition, $(AE)(B)(C)(D)$ is output consistent, and thus this system can be reduced to one with four states, as shown below.

q	$q^\star$ $x = 0$	$x = 1$	z_2
A	D	B	0
B	A	C	0
C	A	B	1
D	A	C	1

The SP partitions for this are (as we found earlier)

$P_1^\star = (AD)(BC) \qquad P_5^\star = (A)(BCD)$

$P_3^\star = (A)(BC)(D) \qquad P_6^\star = (A)(BD)(C)$

The best assignment seems to be the one that uses $P_1^\star$ and the output consistent partition ($P_{OC} = (AB)(CD)$). That produces

$z = q_2$

$D_1 = x$

$D_2 = q_1'q_2' + xq_2$

If, instead, we used output column z_1, there could be no reduction, and three flip flops would be needed. However, there are 2 two-block SP partitions, in addition to the output consistent one, that can be used for the state assignment.

$P_1 = (ADE)(BC)$

$P_5 = (AE)(BCD)$

$P_{OC} = (AB)(CDE)$

This produces the state assignment

q	q_1	q_2	q_3
A	0	0	0
B	1	1	0
C	1	1	1
D	0	1	1
E	0	0	1

and the equations

$z = q_3$

$D_1 = x$

$D_2 = x + q_2'$

$D_3 = \{x'q_1' \text{ or } x'q_2'\} + q_1'q_2 + \{q_2q_3' \text{ or } q_1q_3'\}$

This requires only five gates plus the NOT gate for x'.

EXAMPLE 9.14

	$q^\star$		z	
q	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A	D	C	0	1
B	F	C	0	0
C	E	A	0	0
D	A	C	1	0
E	C	B	1	0
F	B	C	1	1

The nontrivial SP partitions are

$P_1 = (AB)(C)(DF)(E)$ $P_6 = (AB)(CE)(DF)$

$P_2 = (ABC)(DEF)$ $P_7 = (ABDF)(C)(E)$

$P_3 = (AD)(B)(C)(E)(F)$ $P_8 = (ABDF)(CE)$

$P_4 = (AF)(BD)(C)(E)$ $P_9 = (AD)(BF)(C)(E)$

$P_5 = (A)(BF)(C)(D)(E)$

As can be seen, none of these are output consistent; thus the table cannot be reduced.

For the first two variables, we will use the 2 two-block SP partitions, P_2 and P_8. The product of these are

$P_1 = (AB)(C)(DF)(E)$

For the third variable, we need a partition that separates A from B and D from F. There are many that will do that; we chose

$P_9 = (AF)(BCDE)$

because that corresponds to the second output column and will simplify somewhat the expression for z.

First, we will construct next block tables for q_1 and q_2.

	$q_1^\star$	
q_1	$x = 0$	$x = 1$
0	1	0
1	0	0

	$q_2^\star$	
q_2	$x = 0$	$x = 1$
0	0	1
1	1	0

This produces

$D_1 = x'q_1'$ $D_2 = xq_2' + x'q_2$

For q_3 and z, we will need the state assignment and truth table:

q	q_1	q_2	q_3
A	0	0	1
B	0	0	0
C	0	1	0
D	1	0	0
E	1	1	0
F	1	0	1
	P_2	P_8	P_9

	x	q_1	q_2	q_3	$q_3^\star$	z
B	0	0	0	0	1	0
A	0	0	0	1	0	0
C	0	0	1	0	0	0
—	0	0	1	1	X	X
D	0	1	0	0	1	1
F	0	1	0	1	0	1
E	0	1	1	0	0	1
—	0	1	1	1	X	X
B	1	0	0	0	0	0
A	1	0	0	1	0	1
C	1	0	1	0	1	0
—	1	0	1	1	X	X
D	1	1	0	0	0	0
F	1	1	0	1	0	1
E	1	1	1	0	0	0
—	1	1	1	1	X	X

(We do not need columns for $q_1^\star$ and $q_2^\star$, since we already computed the inputs for those flip flops from the next block table.) The resulting maps are

$q_2\,q_3$ \ $x\,q_1$	00	01	11	10
00	1	1		
01				
11	X	X	X	X
10				1

$q_3^\star$

$q_2\,q_3$ \ $x\,q_1$	00	01	11	10
00		1		
01		1	1	1
11	X	X	X	X
10		1		

z

From this, we can find

$$D_3 = x'q_2'q_3' + xq_1'q_2 \qquad z = x'q_1 + xq_3$$

The advantage of using SP partitions is even more dramatic with *JK* flip flops (since *J* and *K* do not depend on the state of that flip flop). Thus, for this example,

$$J_1 = x' \qquad K_1 = 1$$
$$J_2 = x \qquad K_2 = x$$
$$J_3 = x'q_2' + xq_1'q_2 \qquad K_3 = 1$$

If, instead, we used the state assignment

q	q_1	q_2	q_3
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0
F	1	0	1

we would obtain the equations

$$\begin{aligned} z &= x'q_1 + q_1q_3 + xq_1'q_2'q_3' + x'q_2q_3 \\ D_1 &= x'q_1'q_2q_3 + x'q_2q_3' \\ D_2 &= xq_3 + x'q_2'q_3' + q_1'q_2'q_3' \\ D_3 &= x'q_1'q_2' + xq_1q_3' + x'q_1q_3 \end{aligned}$$

These equations are much more complex than the previous solution.

EXAMPLE 9.15

q	$q^\star$ $x = 0$	$x = 1$	z
A	B	C	0
B	D	C	1
C	A	E	0
D	A	C	0
E	A	C	1

The SP partitions are

$$\begin{aligned} P_1 &= (ABD)(C)(E) & P_4 &= (A)(B)(CE)(D) \\ P_2 &= (ABDE)(C) & P_5 &= (A)(B)(C)(DE) \\ P_3 &= (A)(B)(CDE) & P_6 &= (ABD)(CE) \end{aligned}$$

None of these is output consistent; therefore, the system cannot be reduced. Although there are 2 two-block partitions, we cannot use both of them, since their product is P_1, which has three states in the same block. One more two-block partition cannot separate these three states. We can use P_6 and the output consistent partition,

$$P_7 = (ACD)(BE)$$

for two of the variables. Their product is

$$(AD)(B)(C)(E)$$

We now need to choose one more partition to separate A and D. From the list of SP partitions, P_3 is attractive. It groups C, D, and E. We could use either

$$P_8 = (AB)(CDE) \quad \text{or} \quad P_9 = (A)(BCDE)$$

The two state assignments are

q	q_1	q_2	q_3
A	0	0	0
B	0	1	0
C	1	0	1
D	0	0	1
E	1	1	1

q	q_1	q_2	q_3
A	0	0	0
B	0	1	1
C	1	0	1
D	0	0	1
E	1	1	1

The resulting sets of equations for the first assignment are

$$\begin{array}{ll} J_1 = x & K_1 = x' \\ J_2 = x'q_3' + xq_1 & K_2 = 1 \\ J_3 = x + q_2 & K_3 = x' \\ z = q_2 & \end{array}$$

For the second assignment, J_1, K_1, J_2, K_2, and z are unchanged; the others become

$$J_3 = 1 \qquad K_3 = x'q_2' + x'q_1$$

If, instead, we use the first five combinations for the five states, the equations become

$$\begin{array}{ll} J_1 = q_2q_3' & K_1 = 1 \\ J_2 = x + q_3 & K_2 = x' + q_3' \\ J_3 = x'q_1'q_2' & K_3 = x + q_2 \\ z = q_1 + q_2'q_3 & \end{array}$$

The cost of this combinational logic is about double that of the first solution.

The choice of state assignment is more of an art than a science. Surely, we want to use two-block SP partitions when possible. But when we run out of those, we use the output consistent partition and the groupings suggested by other SP partitions (if there are any).

This approach does not guarantee a minimum solution. The only way to do that is to try *all* possible sets of partitions. (In some unusual circumstances, it may even be possible to find a less costly solution with an extra flip flop or without reducing the number of states to a minimum.)

[SP 4; EX 4, 5, 6, 7]

9.5 SOLVED PROBLEMS

1. Reduce each of the following systems to ones with the minimum number of states using the tabular method.

a.

	$q^\star$		
q	$x = 0$	$x = 1$	z
A	C	B	0
B	D	A	1
C	A	B	0
D	B	B	1

b.

	$q^\star$		
q	$x = 0$	$x = 1$	z
A	C	D	1
B	C	C	1
C	E	B	0
D	E	A	0
E	A	B	1

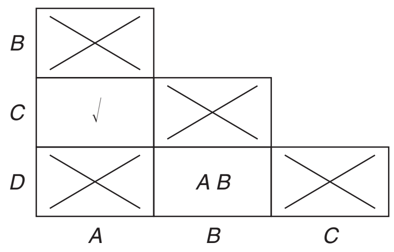
c.

q	$q^\star$ $x = 0$	$x = 1$	z
A	E	B	0
B	D	A	1
C	F	B	0
D	E	B	1
E	D	C	1
F	D	A	1

d.

q	$q^\star$ $x = 0$	$x = 1$	z
A	D	G	0
B	C	E	1
C	B	G	0
D	A	B	1
E	F	E	0
F	G	B	1
G	F	A	0

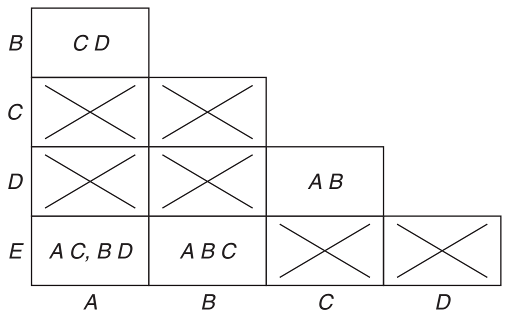
a. We will first construct the chart



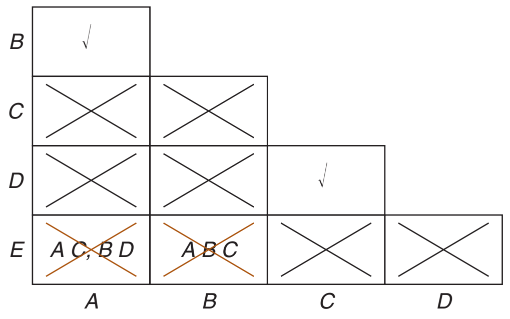
Since the *AB* box already has an X in it, the only equivalent states are *A* and *C*. The state table can be reduced to one with three states, as follows:

q	$q^\star$ $x = 0$	$x = 1$	z
A-C	A-C	B	0
B	D	A-C	1
D	B	B	1

b. The chart for this table is



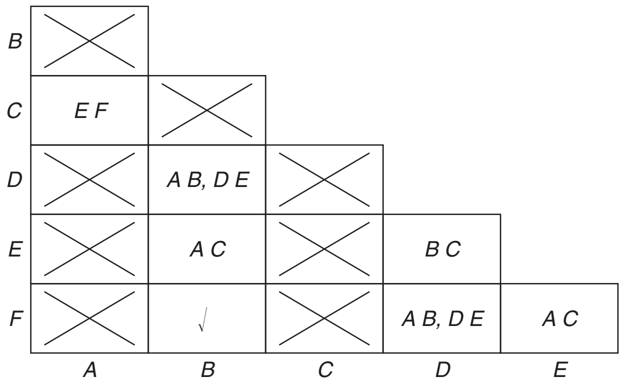
Since for *A* to be together with *B* requires *C* to be grouped with *D*, and the *CD* grouping requires *A* and *B* to be together, both of those can be checked off. *A* and *C* cannot be in the same block of a partition; thus, the other two are crossed off, resulting in



The reduced state table is thus (where *A*-*B* has been called *A*, and *C*-*D* has been called *C*)

q	$q^\star$ $x = 0$	$x = 1$	z
A	*C*	*C*	1
C	*E*	*A*	0
E	*A*	*A*	1

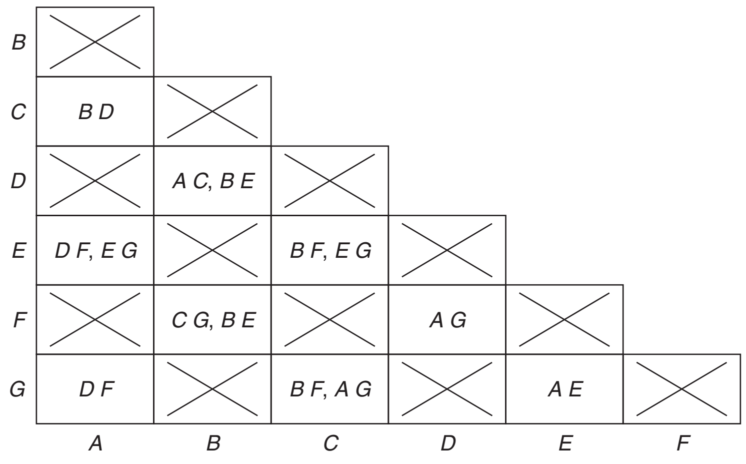
c. We get the following chart:



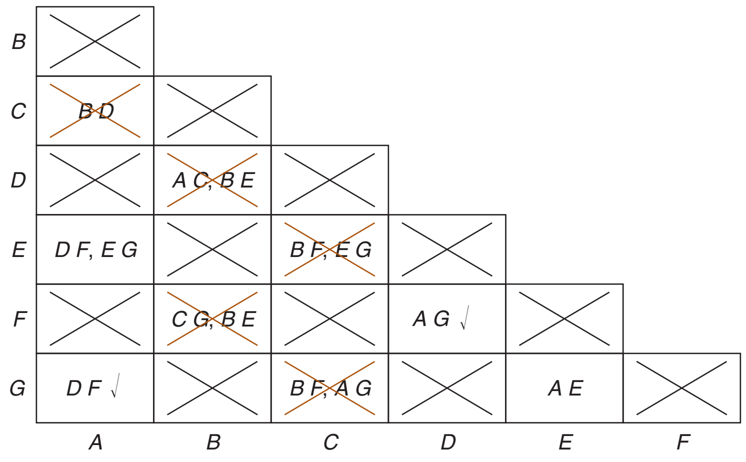
B and *F* are already grouped; *AC* and *EF* also group. That produces a group with *B*, *E*, and *F*, reducing the table to one with only three states (since *AB* and *BC* cannot be grouped):

q	$q^\star$ $x = 0$	$q^\star$ $x = 1$	z
A	B	B	0
B	D	A	1
D	B	B	1

d. The chart becomes



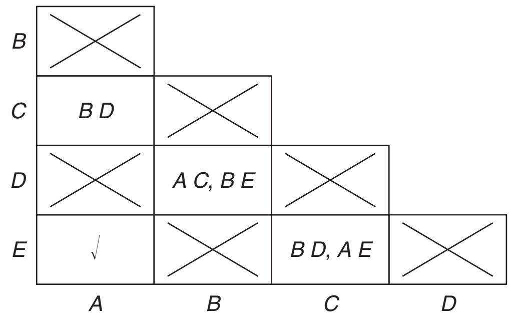
We can cross out a few squares and obtain



On the first pass, we were able to cross out *BE*, which then allowed us to cross out *BD* and *BF*. Since those pairs could not be equivalent, we were then able to eliminate *AC*, *CE*, and *EG*. At this point, we note that for *A* and *G* to be equivalent, *D* and *F* must be equivalent and for *D* and *F* to be equivalent, *A* and *G* must be equivalent. We can thus reduce the number of states by two, reduce the state table and repeat the process, as follows:

	$q^\star$		
q	$x = 0$	$x = 1$	z
A	*D*	*A*	0
B	*C*	*E*	1
C	*B*	*A*	0
D	*A*	*B*	1
E	*D*	*E*	0

The new smaller chart is thus



Only *A* and *E* can be equivalent, since *B* and *E* are not equivalent, making *B* and *D* not equivalent. Thus, we can reduce this further to four states, namely,

	$q^\star$		
q	$x = 0$	$x = 1$	z
A	*D*	*A*	0
B	*C*	*A*	1
C	*B*	*A*	0
D	*A*	*B*	1

Of course, we could have determined this from the original chart, where we have replaced the crossed out squares with X's and the two equivalences we had previously determined with checks.

B	X					
C	X	X				
D	X	X	X			
E	D F, G E	X	X	X		
F	X	X	X	√	X	
G	√	X	X	X	A E	X
	A	B	C	D	E	F

From this table, we can see that *G* and *E* are equivalent if *A* and *E* are equivalent, grouping *A*, *G*, and *E*. That would directly produce the same table with four states as above.

2. For the same state tables as in Solved Problem 1, find all of the nontrivial SP partitions. (Of course, the output columns are not used.)

a. $(AB) \rightarrow (CD)$	$\rightarrow \rightarrow (AB)(CD) = P_1$
(AC) √	$\rightarrow \rightarrow (AC)(B)(D) = P_2$
$(AD) \rightarrow (BC) \rightarrow (AB)$	$\rightarrow \rightarrow (ABCD) = P_N$
$(BC) \rightarrow (ABD)$	$\rightarrow \rightarrow P_N$
$(BD) \rightarrow (AB)$	$\rightarrow \rightarrow P_N$
$(CD) \rightarrow (AB)$	$\rightarrow \rightarrow (AB)(CD) = P_1$

b. $(AB) \rightarrow (CD)$	$\rightarrow \rightarrow (AB)(CD)(E) = P_1$
$(AC) \rightarrow (CE)(BD)$	$\rightarrow \rightarrow (ACE)(BD) = P_2$
$(AD) \rightarrow (CE) \rightarrow (AE)$	$\rightarrow \rightarrow P_N$
$(AE) \rightarrow (AC)(BD)$	$\rightarrow \rightarrow P_2$
$(BC) \rightarrow (CE)(ABCE)$	$\rightarrow \rightarrow P_N$
$(BD) \rightarrow (ACE)$	$\rightarrow \rightarrow P_2$
$(BE) \rightarrow (ABC)$	$\rightarrow \rightarrow P_N$
$(CD) \rightarrow (AB)$	$\rightarrow \rightarrow P_1$
$(CE) \rightarrow (AE) \rightarrow (BD)$	$\rightarrow \rightarrow P_2$
$(DE) \rightarrow (ABE)$	$\rightarrow \rightarrow P_N$

c. $(AB) \rightarrow (DE) \rightarrow (BC) \rightarrow (DF)$	$\rightarrow \rightarrow (ABC)(DEF) = P_1$
$(AC) \rightarrow (EF)$	$\rightarrow \rightarrow (AC)(B)(EF)(D) = P_2$
(AD) √	$\rightarrow \rightarrow (AD)(B)(C)(E)(F) = P_3$
$(AE) \rightarrow (DE)(BC)$	$\rightarrow \rightarrow P_N$

$(AF) \to (DE)(AB) \to (BC) \quad \to \to P_N$
$(BC) \to (DF)(AB)$
$\quad \to (ABC)(DEF) \quad \to \to P_1$
$(BD) \to (DE)(AB) \to (BC) \quad \to \to P_N$
$(BE) \to (AC) \to (EF) \quad \to \to (AC)(BEF)(D) = P_4$
$(BF) \to \surd \quad \to \to (A)(BF)(C)(D)(E) = P_5$
$(CD) \to (EF)(AC) \quad \to \to (ACD)(B)(EF) = P_6$
$(CF) \to (DF)(AB) \to (DE) \quad \to \to P_N$
$(DE) \to (BC) \to (DF)(AB) \quad \to \to P_1$
$(DF) \to (DE)(AB) \quad \to \to P_1$
$(EF) \to (AC) \quad \to \to P_2$

At this point, we have found six SP partitions. We must now add each pair, since the sum of SP partitions is also SP. The only new partitions are

$$P_3 + P_4 = P_7 = (ACD)(BEF)$$
$$P_3 + P_5 = P_8 = (AD)(BF)(C)(E)$$

No new sums are formed from these two (since P_7 is two-block). (We will return to this example in later solved problems.)

d. $(AB) \to (CD)(EG) \to (ABEG)$
$\quad \to (CDF) \quad \to \to (ABEG)(CDF) = P_1$
$(AC) \to (BD) \to (BE) \to (CF) \to (BDG) \quad \to \to P_N$
$(AD) \to (BG) \to (CF)(AE) \quad \to \to P_N$
$(AE) \to (DF)(EG) \to (AEG) \quad \to \to (AEG)(B)(C)(DF) = P_2$
$(AF) \to (BDG) \to (ACF)(ABE) \quad \to \to P_N$
$(AG) \to (DF) \quad \to \to (AG)(B)(C)(DF)(E) = P_3$
$(BC) \to (GE) \to (AE) \to (DF) \quad \to \to (AEG)(BC)(DF) = P_4$
$(BD) \to (AC)(BE) \quad \to \to P_N$
$(BE) \to (CF) \to (BG) \to (AE) \quad \to \to P_1$
$(BF) \to (CG)(BE) \to (BF)(AG) \quad \to \to P_N$
$(BG) \to (CF)(AE) \to (DF)(EG) \quad \to \to P_1$
$(CD) \to (ABG) \quad \to \to P_1$
$(CE) \to (BF)(EG) \quad \to \to P_N$
$(CF) \to (BG) \to (AE) \to (DF)(EG) \quad \to \to P_1$
$(CG) \to (BF)(AG) \quad \to \to P_N$
$(DE) \to (AF)(BE) \quad \to \to P_N$
$(DF) \to (AG) \quad \to \to P_3$
$(DG) \to (ABF) \quad \to \to P_N$
$(EF) \to (FG)(BE) \to (AB) \quad \to \to P_N$

$(EG) \rightarrow (AE) \qquad \rightarrow \rightarrow P_2$

$(FG) \rightarrow (AB) \qquad \rightarrow \rightarrow P_N$

The sums produce nothing new. Thus, there are four nontrivial SP partitions.

3. a. Reduce the system of Solved Problem 2a to one with a minimum number of states if the output column is

	(i)	(ii)	(iii)
A	0	1	0
B	1	0	0
C	0	0	1
D	1	1	1

b. Reduce the system of Solved Problem 2c to one with a minimum number of states and find all of the SP partitions of the reduced system if the output column is

	(i)	(ii)	(iii)
A	0	1	0
B	1	0	0
C	0	1	1
D	0	0	1
E	0	0	1
F	1	0	0

a. (i) $P_2 = (AC)(B)(D)$ is the only output consistent SP partition. Thus, the system can be reduced to one with three states:

	$q^\star$		
q	$x = 0$	$x = 1$	z
A	*A*	*B*	0
B	*D*	*A*	1
D	*B*	*B*	1

(ii) There are no output consistent SP partitions; therefore, the system cannot be reduced for this output column.

(iii) $P_1 = (AB)(CD)$ is the only output consistent SP partition. Thus, we can reduce the system to one with just two states:

	$q^\star$		
q	$x = 0$	$x = 1$	z
A	*C*	*A*	0
C	*A*	*A*	1

b. (i) The output consistent SP partitions are

$P_3 = (AD)(B)(C)(E)(F)$
$P_5 = (A)(BF)(C)(D)(E)$
$P_8 = (AD)(BF)(C)(E)$

Clearly, P_8 is larger than either of the others; it can be used to reduce the system to one with only four states. None of the SP partitions is larger than P_8; therefore, the reduced system has no nontrivial SP partitions.

q	$q^\star$ $x = 0$	$x = 1$	z
A	E	B	0
B	A	A	1
C	B	B	0
E	A	C	0

(ii) The output consistent SP partitions are

$P_2 = (AC)(B)(D)(EF)$
$P_4 = (AC)(BEF)(D)$
$P_5 = (A)(BF)(C)(D)(E)$

P_4 is larger than either of the others and can be used to reduce this to a system with three states:

q	$q^\star$ $x = 0$	$x = 1$	z
A	B	B	1
B	D	A	0
D	B	B	0

Since $P_7 > P_4$, then $P_7^\star = (AD)(B)$ is SP.

(iii) The only output consistent SP partition is

$P_5 = (A)(BF)(C)(D)(E)$

Thus, the minimum system requires five states, namely,

q	$q^\star$ $x = 0$	$x = 1$	z
A	E	B	0
B	D	A	0
C	B	B	1
D	E	B	1
E	D	C	1

There are three SP partitions for this reduced system,

$$P_4^\star = (AC)(BE)(D)$$
$$P_7^\star = (ACD)(BE)$$
$$P_8^\star = (AD)(B)(C)(E)$$

4. Find good state assignments for each of the following state tables. (Each of the first four correspond to one of the state tables from Solved Problem 2.) Compute the input equations for either D or JK flip flops and the output equation.

a.

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	B	0
B	D	A	1
C	A	B	0
D	B	B	0

b.

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	B	1
B	D	A	1
C	A	B	0
D	B	B	1

c.

q	$q^\star$ $x = 0$	$x = 1$	z
A	E	B	0
B	D	A	0
C	F	B	1
D	E	B	1
E	D	C	1
F	D	A	1

d.

q	$q^\star$ $x = 0$	$x = 1$	z
A	E	B	0
B	D	A	0
C	F	B	1
D	E	B	1
E	D	C	0
F	D	A	1

e.

q	$q^\star$ $x = 0$	$x = 1$	z $x = 0$	$x = 1$
A	C	D	0	0
B	E	A	1	1
C	A	D	0	0
D	B	A	1	0
E	B	C	1	1

f.

q	$q^\star$ $x = 0$	$x = 1$	z $x = 0$	$x = 1$
A	C	D	0	1
B	E	A	1	1
C	A	D	0	0
D	B	A	1	0
E	B	C	1	1

a. $P_2 = (AC)(B)(D)$ is output consistent; therefore, this system can be reduced to one with three states, namely,

	$q^\star$		
q	$x = 0$	$x = 1$	z
A	A	B	0
B	D	A	1
D	B	B	0

Since the other SP partition is not larger than P_2, this system has no nontrivial SP partitions. There is not a good clue as to how to choose partitions for a state assignment, other than choosing the output consistent one to minimize the output logic. We can try both

i.

q	q_1	q_2
A	0	0
B	0	1
D	1	0

ii.

q	q_1	q_2
A	0	0
B	1	1
D	1	0

For assignment i, we get

$$D_1 = x'q_2 \quad D_2 = q_1 + xq_2' \quad z = q_2$$

and for assignment ii, we obtain

$$D_1 = x'q_1 + xq_2' \qquad D_2 = q_1q_2' + xq_2' \qquad z = q_2$$

The first assignment requires the least amount of logic. If we tried the third assignment, we would find that it needs about the same amount of logic as the second (but uses 2 three-input gates).

---

b. This, of course, is the same next state behavior as part a, but the new output column is such that there are no output consistent SP partitions. We will implement it with *JK* flip flops. We do have one two-block SP partition,

$$P_1 = (AB)(CD)$$

The two-block output consistent partition is not useful, since its product with P_1 is not P_0. We will use for the second variable

$$P_3 = (AC)(BD)$$

which takes advantage of the other SP partition, $P_2 = (AC)(B)(D)$, by putting A and C in the same block. This results in

$$J_1 = x' \quad K_1 = 1 \quad J_2 = x \quad K_2 = xq_1' \quad z = q_1' + q_2$$

Each of the other solutions requires very little logic as well. That will normally be the case with only two flip flops.

c. There are no output consistent SP partitions; thus, this system cannot be reduced. There are 2 two-block SP partitions,

$$P_1 = (ABC)(DEF)$$
$$P_7 = (ACD)(BEF)$$

which will be used for the first two variables. The output consistent two-block partition is not useful, since its product with P_1 and P_7 is not P_0; states E and F would have the same assignment. We need a partition to separate A from C and E from F. P_3 indicates that A and D should be together; P_5 indicates that B and F should be together. One of the partitions that accomplishes these goals and still produces a product of P_0 with P_1 and P_7 is

$$P_9 = (ABDF)(CE)^*$$

The resulting state assignment is

q	q_1	q_2	q_3
A	0	0	0
B	0	1	0
C	0	0	1
D	1	0	0
E	1	1	1
F	1	1	0

The equations for q_1 and q_2 can be obtained from just the block tables; the equation for D_3 requires a 16-row truth table and those for z require an 8-row table. (The work is left as an exercise for the reader.)

$$D_1 = x' \quad D_2 = q_2' \quad z = q_1 + q_3$$
$$D_3 = x'q_2'q_3' + \{xq_1q_3 \text{ or } xq_2q_3\}$$

This solution requires four gates plus the NOT gate. If we used the straight binary assignment (A: 000, B: 001, . . .), we would need 13 gates.

d. The next state portion of the table (and thus the list of SP partitions) is the same as for part c. But, in this case, the product of the output consistent partition

$$P_{OC} = (ABE)(CDF)$$

*There are others; you should try them as an exercise to see if one of them produces a less costly solution.

with the other two is P_0, and we can use it for the third variable. The inputs to the first two flip flops would be the same as for part c; the inputs for the third flip flop and the output would be

$$D_3 = x'q_1'q_2 + x'q_2 + q_1'q_3' \quad z = q_3$$

This requires seven gates. As in part c, the straight binary assignment would be much more expensive (14 gates).

e. The SP partitions are found as follows (ignoring the output section, of course):

$$\begin{array}{ll}
(AB) \rightarrow (AD)(CE) \rightarrow (BC)(CD) & \rightarrow\rightarrow P_N \\
(AC) \rightarrow \surd & \rightarrow\rightarrow (AC)(B)(D)(E) = P_1 \\
(AD) \rightarrow (BC) \rightarrow (ADE) \rightarrow (AC) & \rightarrow\rightarrow P_N \\
(AE) \rightarrow (BCD) & \rightarrow\rightarrow P_N \\
(BC) \rightarrow (ADE) \rightarrow (AC) & \rightarrow\rightarrow P_N \\
(BD) \rightarrow (BE) \rightarrow (AC) & \rightarrow\rightarrow (AC)(BDE) = P_2 \\
(BE) \rightarrow (AC) & \rightarrow\rightarrow (AC)(BE)(D) = P_3 \\
(CD) \rightarrow (ABD) & \rightarrow\rightarrow P_N \\
(CE) \rightarrow (AB)(CD) & \rightarrow\rightarrow P_N \\
(DE) \rightarrow (AC) & \rightarrow\rightarrow (AC)(B)(DE) = P_4
\end{array}$$

Note that both

$$P_N \geq P_2 \geq P_3 \geq P_1 \geq P_0$$
$$P_N \geq P_2 \geq P_4 \geq P_1 \geq P_0$$

No additional SP partitions are found by taking the sum of these.

An inspection of the state table shows that P_1 and P_3 are output consistent; thus the system can be reduced (using the larger of these) to one with three states (combining A with C and B with E), as follows:

	$q^\star$		z	
q	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A	A	D	0	0
B	B	A	1	1
D	B	A	1	0

The only SP partition for this reduced table is

$$P_3^\star = (A)(BD)$$

It can be used for one variable, q_1, in the state assignment. For that variable, we just need a next block table, namely,

q	$q^\star$	
	$x = 0$	$x = 1$
0(A)	0	1
1(B-D)	1	0

Either directly from the state table or by converting it to a truth table or directly to maps, we can determine

$$J_1 = K_1 = x$$

The first row of $q^\star$ is the map for J and the second row is that for K'. (We can always do a block table for variables assigned using SP partitions.)

There is no clue from the next state portion as to which partition to choose for the other variable. However, if we use

$$P_4 = (AD)(B)$$

(which corresponds to the second column of the output section), we are assured of a fairly simple output equation, namely,

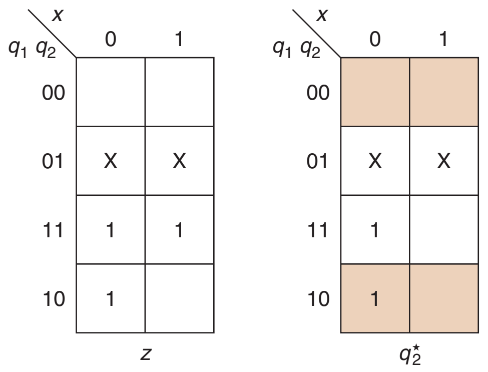
$$z = x'q_1 + xq_2$$

If we choose P_4 for q_2, then the state assignment and the truth table for the next value of q_2 and the output become

q	q_1	q_2
A	0	0
B	1	1
D	1	0

q	x	q_1	q_2	z	$q_2^\star$
A	0	0	0	0	0
—	0	0	1	X	X
D	0	1	0	1	1
B	0	1	1	1	1
A	1	0	0	0	0
—	1	0	1	X	X
D	1	1	0	0	0
B	1	1	1	1	0

The maps for z and q_2 (with the J portion for the quick method shaded) are



The resulting equations are

$$z = q_2 + x'q_1 \qquad J_2 = x'q_1 \qquad K_2 = x$$

Note that we have an even simpler version of z than expected.

If we retain the SP partition for assigning the first variable and use the other two-block partition, $P_5 = (AB)(D)$, for q_2, we get

$$z = x'q_2 + q_1q_2' \qquad J_2 = xq_1' \qquad K_2 = 1$$

There is not much difference. Finally, if we do not use the SP partition, but rather use the state assignment

q	q_1	q_2
A	0	0
B	0	1
D	1	0

we will get

$$z = q_2 + x'q_1 \qquad J_1 = xq_2' \qquad K_2 = 1 \qquad J_2 = x'q_1$$
$$K_2 = x$$

For this simple problem, the state assignment does not make a major difference.

f. The table of part f has one small change in the output section, so that there are no longer any output consistent SP partitions. We now need three variables, only one of which can use an SP partition. The only two-block SP partition is P_2, and we will use it to assign q_1. We will then use P_3 to help with the second variable; it keeps A and C together, and B and E together. We must group D with AC; otherwise, we would just repeat the same partition as before. Thus, we use

$$P_5 = (ACD)(BE)$$

The product of these two is P_3 and we must now find a two-block partition, the product of which with P_3 is P_0. There are several possibilities, such as

$$P_6 = (ABD)(CE)$$
$$P_7 = (AB)(CDE)$$
$$P_8 = (AE)(BCD)$$
$$P_9 = (ADE)(BC)$$

Any of these might lead to a good solution. Using P_2, P_5, and P_6, we have the following state assignment and design tables:

q	q_1	q_2	q_3
A	0	0	0
B	1	1	0
C	0	0	1
D	1	0	0
E	1	1	1

q_1	$q_1^\star$ $x = 0$	$x = 1$
0	0	1
1	1	0

	x	q_1	q_2	q_3	$q_2^\star$	$q_3^\star$	z
A	0	0	0	0	0	1	0
C	0	0	0	1	0	0	0
—	0	0	1	0	X	X	X
—	0	0	1	1	X	X	X
D	0	1	0	0	1	0	1
—	0	1	0	1	X	X	X
B	0	1	1	0	1	1	1
E	0	1	1	1	1	0	1
A	1	0	0	0	0	0	1
C	1	0	0	1	0	0	0
—	1	0	1	0	X	X	X
—	1	0	1	1	X	X	X
D	1	1	0	0	0	0	0
—	1	1	0	1	X	X	X
B	1	1	1	0	0	0	1
E	1	1	1	1	0	1	1

The block table is used to solve for q_1 and the truth table allows us to solve for the other two flip flop inputs and the system output. The resulting equations are

$$D_1 = xq_1' + x'q_1$$
$$D_2 = x'q_1$$
$$D_3 = x'q_1'q_3' + x'q_2q_3' + \{xq_1q_3 \text{ or } xq_2q_3\}$$
$$z = q_2 + x'q_1 + xq_1'q_3'$$

requiring nine gates (since $x'q_1$ need only be built once) (plus a NOT gate to form x').

If, instead, we used the assignment (just the first five binary numbers)

q	q_1	q_2	q_3
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0

we get the equations

$$D_1 = x'q_2'q_3$$
$$D_2 = xq_3' + q_1'q_2'q_3$$
$$D_3 = x'q_1 + xq_1'q_3' + q_2q_3'$$
$$z = q_1 + x'q_3 + xq_2'$$

This requires 11 gates (plus the NOT for x'), significantly more logic than required for the other assignment. The other assignments with P_7, P_8, and P_9 are left as an exercise.

9.6 EXERCISES

1. Reduce each of the following systems to ones with the minimum number of states using the tabular method.

a.

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	B	0
B	D	A	0
C	A	B	1
D	B	B	1

b.

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	B	0
B	D	C	1
C	A	B	0
D	A	B	0

★c.

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	B	0
B	D	A	0
C	E	A	1
D	E	B	1
E	D	B	1

d. Same table as c, except that the output for state B is 1.

e.

q	$q^\star$		z
	$x = 0$	$x = 1$	
A	F	B	0
B	E	C	0
C	D	C	1
D	C	A	0
E	B	C	1
F	A	B	0

f.

q	$q^\star$		z
	$x = 0$	$x = 1$	
A	B	B	0
B	F	D	0
C	D	A	1
D	C	E	0
E	F	E	0
F	E	A	1

g.

q	$q^\star$		z
	$x = 0$	$x = 1$	
A	B	C	0
B	D	E	1
C	D	F	0
D	B	E	1
E	F	C	0
F	D	A	0

★h.

q	$q^\star$		z	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A	B	D	0	0
B	E	G	1	0
C	G	F	0	0
D	A	C	1	1
E	B	D	0	0
F	G	D	0	0
G	A	B	1	0

i.

q	$q^\star$		z
	$x = 0$	$x = 1$	
A	E	B	0
B	C	C	1
C	D	E	1
D	F	B	0
E	A	F	1
F	D	F	1

j.

q	$q^\star$		z
	$x = 0$	$x = 1$	
A	B	G	0
B	A	E	1
C	A	F	1
D	G	B	1
E	C	D	0
F	B	D	0
G	G	B	1

★k.

q	$q^\star$		z
	$x = 0$	$x = 1$	
A	G	B	0
B	E	C	0
C	E	B	1
D	A	B	0
E	F	D	0
F	E	D	1
G	A	B	1

l.

q	$q^\star$		z
	$x = 0$	$x = 1$	
A	G	B	0
B	E	C	1
C	E	B	1
D	A	B	1
E	F	D	1
F	E	D	1
G	A	B	0

2. For each of the state tables of Exercise 1, find all of the nontrivial SP partitions.

3. a. For state tables a and b of Exercise 2, reduce the system to one with a minimum number of states if the output column is

	(i)	(ii)	(iii)	(iv)
A	0	1	0	0
B	1	0	0	1
C	0	0	1	0
D	1	1	1	0

b. Reduce the system of Exercises 1e and 1g to ones with a minimum number of states if the output column is

	(i)	(ii)	(iii)	(iv)
A	0	1	0	1
B	1	0	0	0
C	0	1	0	0
D	1	1	1	0
E	0	0	1	0
F	1	1	1	0

★c. Reduce the system of Exercises 1k to one with a minimum number of states if the output column is

	(i)	(ii)	(iii)	(iv)
A	0	1	0	1
B	1	0	0	0
C	1	1	1	0
D	1	1	1	1
E	0	0	1	0
F	0	1	0	0
G	0	1	0	1

4. a. For Solved Problem 4c,

i. Find the D's and z for the straight binary assignment.

ii. Find D_3 and z using the two SP partitions for q_1 and q_2 and using $P_9 = (ABDE)(CF)$ for q_3.

iii. Find D_3 and z using the two SP partitions for q_1 and q_2 and using $P_{10} = (ADE)(BCF)$ for q_3.

b. Continue the example of Solved Problem 4f, using

$$P_2 = (AC)(BDE)$$
$$P_5 = (ACD)(BE)$$

and each of

i. $P_7 = (AB)(CDE)$

ii. $P_8 = (AE)(BCD)$

iii. $P_9 = (ADE)(BC)$

5. For each of the state tables shown below, find a good state assignment and design the system using *JK* flip flops. Compare that design with the state assignment that just uses the binary numbers in order for the states (that is, *A*: 000, *B*: 001, *C*: 010, . . .).*

a.

	$q^\star$		
q	$x = 0$	$x = 1$	z
A	*D*	*B*	1
B	*C*	*D*	1
C	*E*	*D*	1
D	*A*	*B*	0
E	*C*	*D*	0

b.

	$q^\star$		z	
q	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A	*D*	*G*	1	0
B	*C*	*E*	1	1
C	*B*	*G*	0	1
D	*A*	*B*	0	0
E	*F*	*E*	1	0
F	*G*	*B*	1	1
G	*F*	*A*	1	1

★c. Exercise 1e with output column
1
0
0
1
1
0

d. Exercise 1k.

6. For each of the following output columns, reduce the system if possible and find a good state assignment

	$q^\star$					
q	$x = 0$	$x = 1$	z_1	z_2	z_3	z_4
A	*E*	*B*	0	0	1	0
B	*C*	*D*	0	0	0	0
C	*E*	*F*	1	1	0	0
D	*E*	*A*	1	0	0	1
E	*C*	*F*	0	1	0	1
F	*C*	*D*	1	0	1	0

*Note that part b has the same next state portion as Solved Problem 2d.

7. Consider the following state table, where the next state is not specified. Complete the next state portion such that the system can be reduced to four states (not any smaller) and it is possible to get from any state to any other state with an appropriate input sequence.

	$q^\star$		z	
q	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A			0	0
B			0	1
C			1	1
D			0	0
E			1	1
F			1	0

9.7 CHAPTER 9 TEST (50 MINUTES)

1. Using the techniques of Section 7.1, reduce the following state table to one with the minimum number of states.

	$q^\star$		
q	$x = 0$	$x = 1$	z
A	C	B	0
B	D	A	0
C	E	A	0
D	E	B	0
E	D	B	1

2. For the state table of Problem 1,

a. For each of the following partitions, indicate whether or not it is SP and whether or not it is output consistent.

$P_1 = (ABCD)(E)$

$P_2 = (ABE)(CD)$

$P_3 = (AC)(BE)(D)$

$P_4 = (AB)(CD)(E)$

$P_5 = (AB)(CDE)$

$P_6 = (A)(B)(C)(D)(E)$

b. Using one of these partitions, reduce the system to the one with the smallest number of states, showing a new state table.

3. For the following state table, find all of the nontrivial SP partitions.

q	$q^\star$ $x = 0$	$x = 1$	z
A	C	B	
B	D	C	
C	A	B	
D	B	C	

4. For the following state table,

q	$q^\star$ $x = 0$	$x = 1$	z
A	D	B	1
B	F	D	0
C	A	D	1
D	E	D	0
E	C	B	1
F	D	C	0

The following are all of the SP partitions,

$P_1 = (AE)(CD)(B)(F)$
$P_2 = (AF)(BC)(D)(E)$
$P_3 = (AEF)(BCD)$

Make a "good" state assignment and show the output equation and the input equations for D flip flops.