7.1 The Karnaugh Map

With the introduction of SOP and POS functions in Chapter 6, we learned how to convert a truth table to a boolean expression, and if necessary, a digital circuit. Recall that in the SOP form of a boolean expression, each row with an output of one corresponded to a product. The OR of all of the products produced an expression that satisfied the truth table, but not necessarily one that was reduced to its simplest form. For example, the truth table below has four rows, each of which corresponds to a one output.

A	B	C	X	
0	0	0	1	A = 0, B = 0, and C = 0
0	0	1	1	A = 0, B = 0, and C = 1
0	1	0	0	
0	1	1	0	
1	0	0	1	A = 1, B = 0, and C = 0
1	0	1	0	
1	1	0	1	A = 1, B = 1, and C = 0
1	1	1	0	

The resulting boolean expression will produce the correct output satisfying the truth table, but it can be simplified.

$$\overline{A}\cdot\overline{B}\cdot\overline{C} + \overline{A}\cdot\overline{B}\cdot C + A\cdot\overline{B}\cdot\overline{C} + A\cdot B\cdot\overline{C}$$

$$\overline{A}\cdot\overline{B}\cdot(\overline{C} + C) + A\cdot\overline{C}\cdot(\overline{B} + B)$$ Distributive Law

$$\overline{A}\cdot\overline{B}\cdot 1 + A\cdot\overline{C}\cdot 1$$ OR'ing anything with its inverse is 1

$$\overline{A}\cdot\overline{B} + A\cdot\overline{C}$$ AND'ing anything with 1 is itself

The application of the rule stating that OR'ing anything with its inverse results in a one (the third line in the above simplification) is the most common way to simplify an SOP expression. This chapter presents a graphical method to quickly pair up products where this rule can be applied in order to drop out as many terms as possible.

Karnaugh Maps are graphical representations of truth tables. They consist of a grid with one cell for each row of the truth table. The grid shown below in Figure 7-1 is the two-by-two Karnaugh map used to represent a truth table with two input variables.

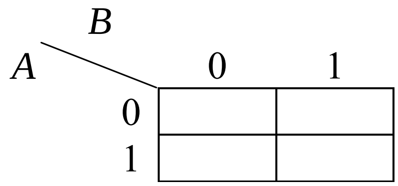


Figure 7-1 2-by-2 Karnaugh Map Used with Two Inputs

The intersection of each row and column corresponds to a unique set of input values. For example, the left column of the Karnaugh map in Figure 7-1 represents the outputs when the input B equals zero and the right column represents the outputs when the input B equals one. The top row represents the outputs when A equals zero and the bottom row represents the outputs when A equals one. Therefore, the left, top cell corresponds to A=0 and B=0, the right, top cell corresponds to A=0 and B=1, and so on. Figure 7-2 shows how the rows of a two-input truth table map to the cells of the Karnaugh map. The variables labeled S_n in the figure represent the binary output values.

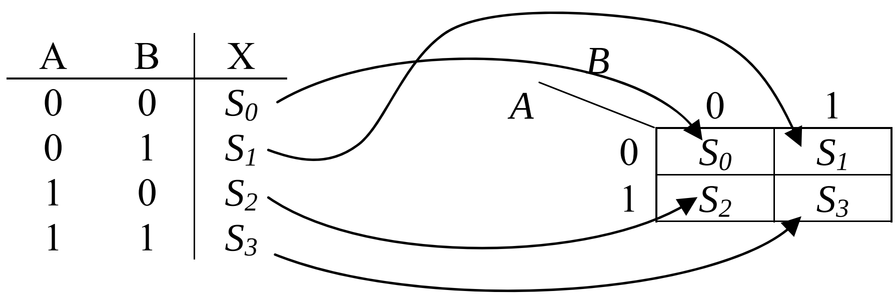


Figure 7-2 Mapping a 2-Input Truth Table to Its Karnaugh Map

The purpose of Karnaugh maps is to rearrange truth tables so that adjacent cells can be represented with a single product using the simplification described above where OR'ing anything with its inverse equals one. This requires adjacent cells to differ by exactly one of their input values thereby identifying the input that will drop out. When four rows or columns are needed as with a 3- or 4-input Karnaugh map, the

2-bit Gray code must be used to ensure that only one input differs between neighboring cells. Take for example the three-input Karnaugh map shown in Figure 7-3. The four rows are each identified with one of the potential values for A and B. This requires them to be numbered 00-01-11-10 in order to have only one input change from row to row.

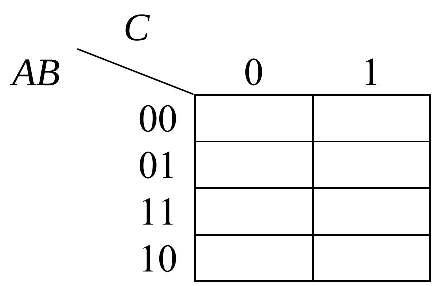


Figure 7-3 Three-Input Karnaugh Map

If we were to use the normal convention for binary counting to number the four rows, they would be numbered 00-01-10-11. In this case, moving from the second to the third row would result in A changing from 0 to 1 and B changing from 1 to 0. This means *two* inputs would change with a vertical movement between two cells and we would lose the simplification benefit we get using Karnaugh maps.

Figure 7-4 shows a four-input Karnaugh map. Notice that the Gray code had to be used for both the rows and the columns.

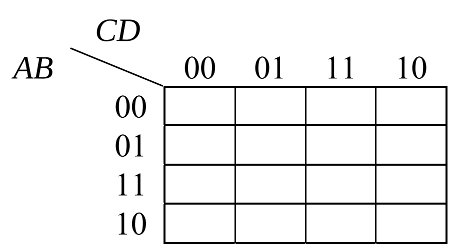


Figure 7-4 Four-Input Karnaugh Map

Note that mapping the outputs from a larger truth table to a Karnaugh map is no different than it was for the two-by-two map except that there are more cells.

We are limited to four input variables when it comes to using Karnaugh maps on paper. Remember that the purpose of a Karnaugh map is to rearrange the truth table so that adjacent cells can be combined allowing for a term to drop out. In other words, the key to the effectiveness of a Karnaugh map is that each cell represents the output for a specific pattern of ones and zeros at the input, and that to move to an adjacent cell, one and only one of those inputs can change.

Take for instance the Karnaugh map in Figure 7-4. The cell in the third column of the second row represents the condition where A=0, B=1, C=1, and D=1. Moving to the cell immediately to the left will change only C; moving right will change D; moving up changes B; and moving down changes A. Therefore, there is an adjacent cell that represents a change in any of the four input variables.

If we were to add a fifth variable, not only would we need to double the number of cells in our map, we would also need to make sure that there were five directions to move adjacently out of every cell in the map. This is impossible to do and remain in two dimensions. A second layer of sixteen cells would have to be added on top of the four-input Karnaugh map to give us a fifth direction, i.e., perpendicular to the page. Although this can be done with a computer, we will not be addressing maps with more than four input variables here.

Example

Convert the three-input truth table below to its corresponding Karnaugh map.

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Solution

The three-input Karnaugh map uses the two-by-four grid shown in Figure 7-3. It doesn't matter which row is used to begin the transfer. Typically, we begin with the top row.

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

AB \ *C*	0	1
00	0	
01		
11		
10		

A few more of the transfers are shown in the map below.

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

AB \ *C*	0	1
00	0	1
01	0	
11	1	
10	1	

The final map is shown below.

AB \ *C*	0	1
00	0	1
01	0	1
11	1	1
10	1	0

7.2 Using Karnaugh Maps

Each cell represents a boolean product just as a row in a truth table does. Figure 7-5 identifies the products for each cell containing a one from the previous example.

This shows that an SOP expression can be derived from a Karnaugh map just as it would be from a truth table.

$$X = (\overline{A} \cdot \overline{B} \cdot C) + (\overline{A} \cdot B \cdot C) + (A \cdot B \cdot \overline{C}) + (A \cdot B \cdot C) + (A \cdot \overline{B} \cdot \overline{C})$$

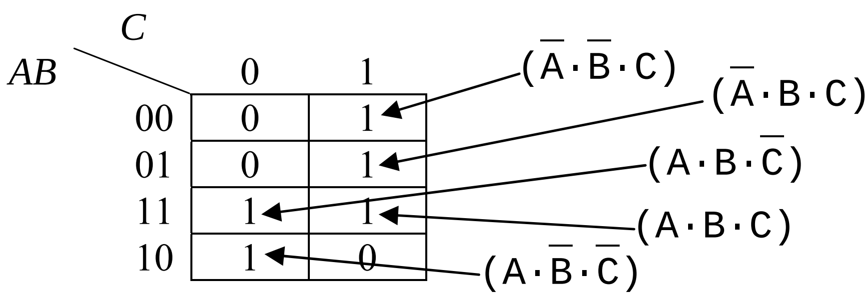


Figure 7-5 Identifying the Products in a Karnaugh Map

If this was all that Karnaugh maps could be used for, they wouldn't be of any use to us. Notice, however, that adjacent cells, either horizontally or vertically adjacent, differ by only one inversion. For example, the top right cell and the cell below it are identical except that B is inverted in the top cell and not inverted in the bottom cell. This implies that we can combine these two products into a single, simpler product depending only on A and C.

$$(\overline{A}\cdot\overline{B}\cdot C)+(\overline{A}\cdot B\cdot C) = \overline{A}\cdot C\cdot(B + \overline{B}) = \overline{A}\cdot C$$

The third row in the Karnaugh map in Figure 7-5 has another adjacent pair of products.

$$(A\cdot B\cdot\overline{C})+(A\cdot B\cdot C) = A\cdot B\cdot(\overline{C} + C) = B\cdot C$$

Karnaugh maps can go even further though. If a pair of adjacent cells containing ones are adjacent to a second pair of adjacent cells containing ones, then all four cells can be represented with a single product with two variables dropping out. For example, the four adjacent cells in Figure 7-6 reduce to a single term with only one of the three original variables left.

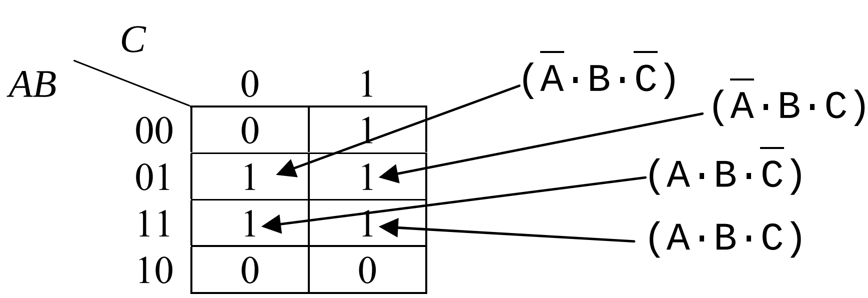


Figure 7-6 Karnaugh Map with Four Adjacent Cells Containing '1'

By applying the rules of boolean algebra, we can see how the products represented by these four cells reduce to a single product with only one variable.

$$X = \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C + A \cdot B \cdot \overline{C} + A \cdot B \cdot C$$

$$X = \overline{A} \cdot B \cdot (\overline{C} + C) + A \cdot B \cdot (\overline{C} + C)$$

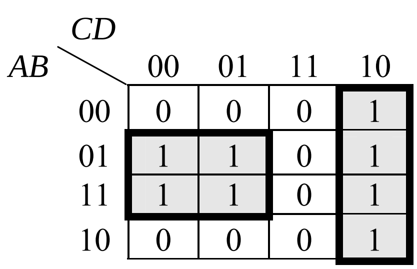
$$X = \overline{A} \cdot B + A \cdot B$$

$$X = B \cdot (\overline{A} + A)$$

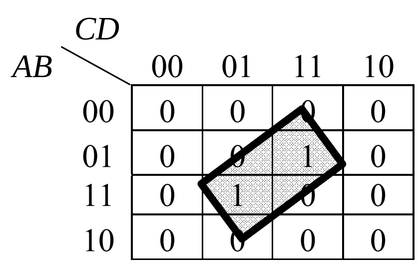
$$X = B$$

So the key to effectively using Karnaugh maps is to find the largest group of adjacent cells containing ones. The larger the group, the fewer products and inputs will be needed to create the boolean expression that produces the truth table. In order for a group of cells containing ones to be considered adjacent, they must follow some rules.

- The grouping must be in the shape of a rectangle. There are no diagonal adjacencies allowed.

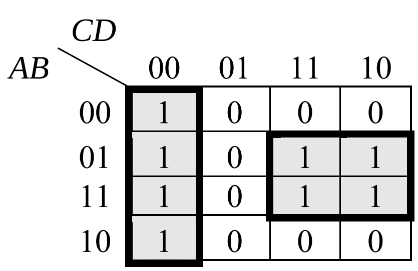


Right

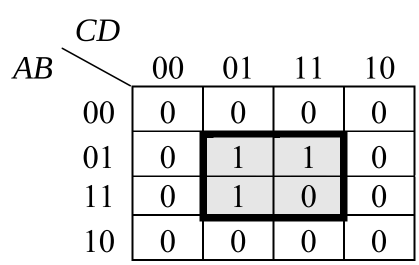


Wrong

- All cells in a rectangle must contain ones. No zeros are allowed.

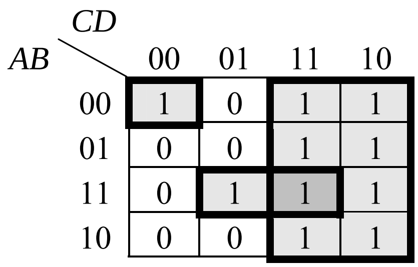


Right

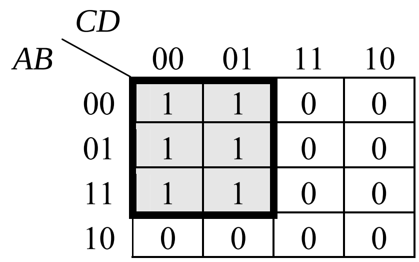


Wrong

- The number of cells in the grouping must equal a power of two, i.e., only groups of 1, 2, 4, 8, or 16 are allowed.

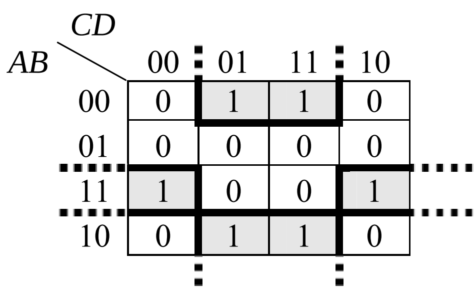


Right

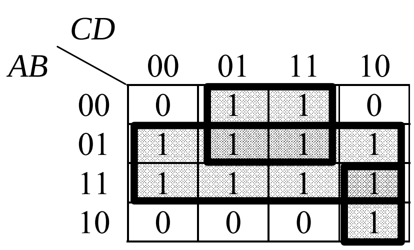


Wrong

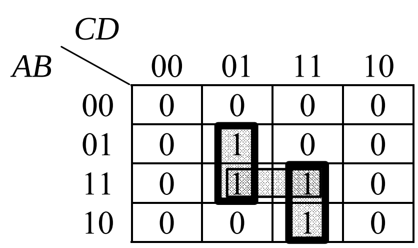
- Outside edges of Karnaugh maps are considered adjacent, so rectangles may wrap from left to right or from top to bottom.



- Cells may be contained in more than one rectangle, but every rectangle must have at least one cell unique to it. (In wrong example, the horizontal rectangle is an unnecessary duplicate.)

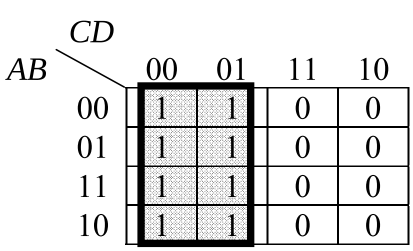


Right

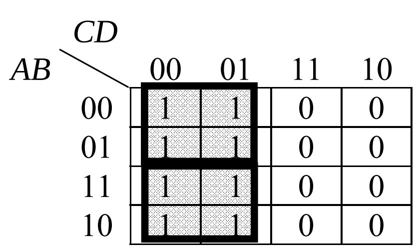


Wrong

- Every rectangle must be as large as possible.

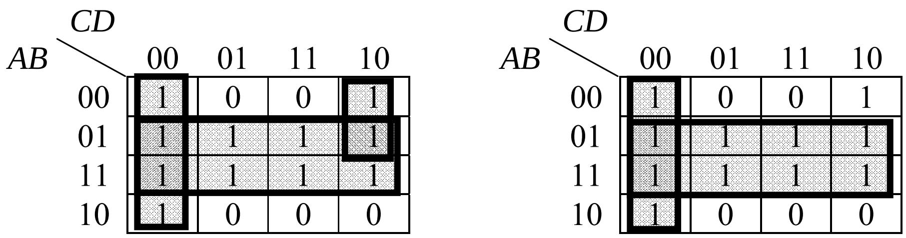


Right



Wrong

- Every 1 must be covered by at least one rectangle.



Right Wrong

The ultimate goal is to create the fewest number of valid rectangles while still covering every 1 in the Karnaugh map. Each rectangle represents a product, and the larger the rectangle, the fewer variables will be contained in that product.

For people new to Karnaugh maps, the easiest way to derive the product represented by a rectangle is to list the input values for all cells in the rectangle, and eliminate the ones that change. For example, the three-input Karnaugh map shown in Figure 7-7 has a four-cell rectangle with the following input values for each of its cells:

Top left cell:	A = 0, B = 1, and C = 0
Top right cell:	A = 0, B = 1, and C = 1
Bottom left cell:	A = 1, B = 1, and C = 0
Bottom right cell:	A = 1, B = 1, and C = 1

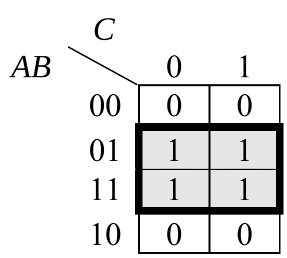


Figure 7-7 Sample Rectangle in a Three-Input Karnaugh Map

The inputs that are the same for *all* cells in the rectangle are the ones that will be used to represent the product. For this example, both A and C are 0 for some cells and 1 for others. That means that these inputs will drop out leaving only B which remains 1 for all four of the cells. Therefore, the product for this rectangle will equal 1 when B equals 1 giving us the same expression we got from simplifying the Figure 7-6 equation:

$$X = B$$

As for the benefits of simplification with Karnaugh maps, each time we are able to double the size of a rectangle, one input to the resulting product drops out. Rectangles containing only one cell will have all of the input variables represented in the final product. Two-cell rectangles will have one fewer input variables; four-cell rectangles will have two fewer input variables; eight-cell rectangles will have three fewer input variables; and so on.

Example

Determine the minimal SOP expression for the truth table below.

A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

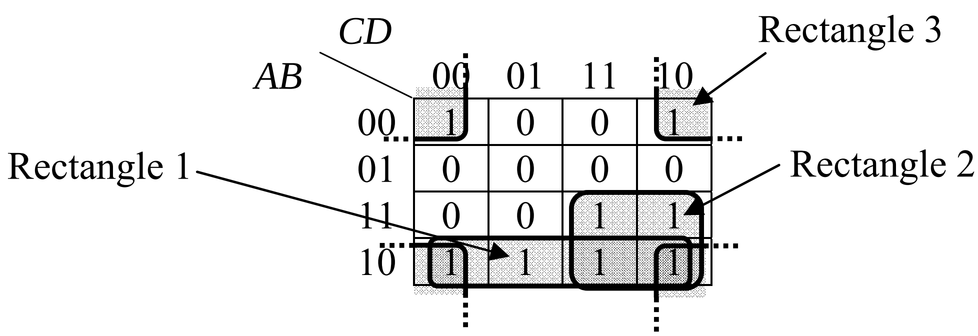
Solution

First, we need to convert the truth table to a Karnaugh map.

AB \ *CD*	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	1	1
10	1	1	1	1

Now that we have the Karnaugh map, it is time to create the rectangles. Two of them are easy to see: all four cells of the bottom row make one rectangle and the four cells that make up the lower right corner quadrant of the map make up a second rectangle.

Less obvious is the third rectangle that takes care of the two cells in the top row that contain ones. Remember that a rectangle can wrap from the left side of the map to the right side of the map. That means that these two cells are adjacent. What's less obvious is that this rectangle can wrap around from top to bottom too making it a four cell-rectangle.



It's okay to have the bottom right cell covered by three rectangles. The only requirement is that no rectangle can be fully covered by other rectangles and that no cell with a 1 be left uncovered.

Now let's figure out the products of each rectangle. Below is a list of the input values for rectangle 1, the one that makes up the bottom row of the map.

Rectangle 1:	A	B	C	D
	1	0	0	0
	1	0	0	1
	1	0	1	1
	1	0	1	0

A and B are the only inputs to remain constant for all four cells: A is always 1 and B is always 0. This means that for this rectangle, the product must output a one when A equals one and B equals zero, i.e., the *inverse* of B equals one. This gives us our first product.

Product for rectangle 1 = $A \cdot \overline{B}$

The product for rectangle 2 is found the same way. Below is a list of the values for A, B, C, and D for each cell of rectangle 2.

Rectangle 2:

A	B	C	D
1	1	1	1
1	1	1	0
1	0	1	1
1	0	1	0

In this rectangle, A and C are the only ones that remain constant across all of the cells. To make the corresponding product equal to one, they must both equal one.

Product for rectangle 2 = $A \cdot C$

Below is a list of the input values for each cell of rectangle 3.

Rectangle 3:

A	B	C	D
0	0	0	0
0	0	1	0
1	0	0	0
1	0	1	0

In rectangle 3, B and D are the only ones that remain constant across all of the cells. To make the corresponding product equal to one, they must both equal 0, i.e., their inverses must equal 1.

Product for rectangle 3 = $\overline{B} \cdot \overline{D}$

From these three products, we get our final SOP expression by OR'ing them together.

$$X = A \cdot \overline{B} + A \cdot C + \overline{B} \cdot \overline{D}$$

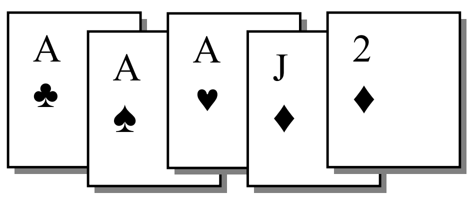
This is a significant simplification over what we would have gotten had we created a product for each row in the truth table. Since the four-input truth table had eight ones in it, the resulting SOP expression would have had eight products each with four input variables. This circuit would have taken eight four-input AND gates and one eight-input OR gate. The circuit from the expression derived from the

Karnaugh map, however, only requires three two-input AND gates and one three-input OR gate.

With practice, many Karnaugh map users can see the variables that will drop out of each rectangle without having to enumerate the input values for every cell. They usually do this by seeing where the rectangle spans variable changes for the rows and columns and drop out those variables. This skill is not necessary, however. Anyone can see which variables drop out by making a list of the bit patterns for each cell in the rectangle and identifying which input variables stay the same and which take on both one and zero as values.

7.3 "Don't Care" Conditions in a Karnaugh Map

Assume you've been invited to play some poker with some friends. The game is five card draw and jacks are wild. What does it mean that "jacks are wild"? It means that if you are dealt one or more jacks, then you can change them to whatever suit or rank you need in order to get the best possible hand of cards. Take for instance the following hand.



Three aces are pretty good, but since you can change the jack of diamonds to anything you want, you could make the hand better. Changing it to a two would give you a full house: three of a kind and a pair. Changing it to an ace, however, would give you an even better hand, one beatable by only a straight flush or five of a kind. (Note that five of a kind is only possible with wild cards.)

If a truth table contains a "don't care" element as its output for one of the rows, that "don't care" is transferred to corresponding cell of the Karnaugh map. The question is, do we include the "don't care" in a rectangle or not? Well, just like the poker hand, you do what best suits the situation.

For example, the four-input Karnaugh map shown in Figure 7-8 contains two "don't care" elements: one represented by the X in the far right cell of the second row and one in the lower left cell.

AB \ *CD*	00	01	11	10
00	1	0	0	0
01	1	0	0	X
11	1	0	1	1
10	X	0	0	0

Figure 7-8 Karnaugh Map with a "Don't Care" Elements

By changing the X in the lower left cell to a 1, we can make a larger rectangle, specifically one that covers the entire left column. If we didn't do this, we would need to use two smaller 2-cell rectangles to cover the ones in the left column.

If we changed the X in the second row to a one, however, it would force us to add another rectangle in order to cover it thereby making the final SOP expression more complex. Therefore, we will assume that that "don't care" represents a zero. Figure 7-9 shows the resulting rectangles.

AB \ *CD*	00	01	11	10
00	1	0	0	0
01	1	0	0	X
11	1	0	1	1
10	X	0	0	0

Figure 7-9 Karnaugh Map with a "Don't Care" Elements Assigned

The final circuit will have a one or a zero in that position depending on whether or not it was included in a rectangle. Later in this book, we will examine some cases where we will want to see if those values that were assigned to "don't cares" by being included or not included in a rectangle could cause a problem.

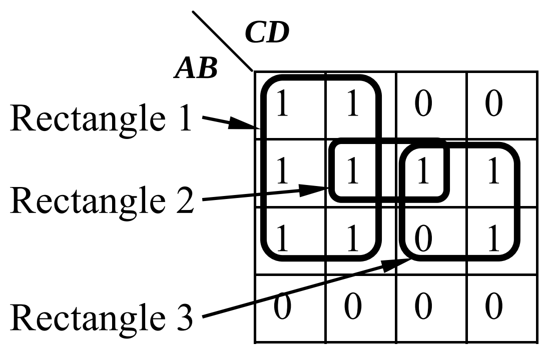
7.4 What's Next?

This chapter has shown how any truth table with up to four input variables can be quickly converted to its minimum SOP circuit expression. This means that if a designer can come up with a truth table, hardware can achieve it.

Chapter 8 presents some common digital circuits by starting with their general concept, and then taking the reader all of the way through to the realization of the hardware. This is done so that the reader can get a feel for the more abstract parts of circuit design, specifically, taking the leap from a system level concept or specification to the boolean expression that will fulfill the requirements.

Problems

1. How many cells does a 3-input Karnaugh map have?
2. What is the largest number of input variables a Karnaugh map can handle and still remain two-dimensional?
3. In a 4-variable Karnaugh map, how many input variables (A, B, C, or D) does a product have if its rectangle of 1's contains 4 cells? Your answer should be 0, 1, 2, 3, or 4.
4. Identify the problems with each of the three rectangles in the Karnaugh map below.



5. When a Karnaugh map has four rows or columns, they are numbered 00, 01, 11, 10 instead of 00, 01, 10, 11. Why?
6. Create Karnaugh maps for each of the truth tables below.

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

7. Derive the minimum SOP expressions for each of the Karnaugh maps below.

AB \ C	0	1
00	1	0
01	0	0
11	0	1
10	1	1

AB \ CD	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	1	1
10	1	0	0	1

AB \ C	0	1
00	0	1
01	1	1
11	1	1
10	0	1

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	0	X
11	1	X	1	1
10	X	0	0	0

8. Create a Karnaugh map that shows there can be more than one arrangement for the rectangles of ones in a Karnaugh map.