7

ADVANCED SIMPLIFYING METHODS

What to Expect

Boolean expressions are used to describe logic circuits and by simplifying those expressions the circuits can also be simplified. This chapter introduces both the *Quine-McCluskey* technique and the *Karma Automated Simplification Tool*. These are methods that are useful for simplifying complex Boolean expressions that include five or more input variables. This chapter includes the following topics.

- Simplifying a complex Boolean expression using the Quine-McCluskey method
- Creating the implicants and prime implicants of a Boolean expression
- Combining prime implicants to create a simplified Boolean expression
- Simplifying a complex Boolean expression using the KARMA automated tool

7.1 QUINE-MCCLUSKEY SIMPLIFICATION METHOD

7.1.1 *Introduction*

This method was developed by W.V. Quine and Edward J. McCluskey and is sometimes called the method of prime implicants.

When a Boolean equation involves five or more variables it becomes very difficult to solve using standard algebra techniques or Karnaugh maps; however, the Quine-McCluskey algorithm can be used to solve these types of Boolean equations.

The Quine-McCluskey method is based upon a simple Boolean algebra principle: if two expressions differ by only a single variable and its complement then those two expressions can be combined:

$$ABC + ABC' = AB \tag{7.1}$$

The Quine-McCluskey method looks for expressions that differ by only a single variable and combines them. Then it looks at the combined expressions to find those that differ by a single variable and

combines them. The process continues until there are no expressions remaining to be combined.

7.1.2 *Example One*

7.1.2.1 *Step 1: Create the Implicants*

Equation 7.2 is the Sigma representation of a Boolean equation.

$$\int(A, B, C, D) = \sum(0, 1, 2, 5, 6, 7, 9, 10, 11, 14) \tag{7.2}$$

Truth Table 7.1 shows the input variables for the *True* minterm values.

Minterm	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
14	1	1	1	0

Table 7.1: Quine-McCluskey Ex 1: Minterm Table

To simplify this equation, the minterms that evaluate to *True* (as listed above) are first placed in a minterm table so that they form sections that are easy to combine. Each section contains only the minterms that have the same number of ones. Thus, the first section contains all minterms with zero ones, the second section contains the minterms with one one, and so forth. Truth Table 7.2 shows the minterms rearranged appropriately.

Number of 1's	Minterm	Binary
0	0	0000
1	1	0001
	2	0010
2	5	0101
	6	0110
	9	1001
	10	1010
3	7	0111
	11	1011
	14	1110

Table 7.2: Quine-McCluskey Ex 1: Rearranged Table

Start combining minterms with other minterms to create Size Two Implicants (called that since each implicant combines two minterms), but only those terms that vary by a single binary digit can be combined. When two minterms are combined, the binary digit that is different between the minterms is replaced by a dash, indicating that the digit does not matter. For example, 0000 and 0001 can be combined to form 000—. The table is modified to add a Size Two Implicant column that indicates all of the combined terms. Note that every minterm must be compared to every other minterm so all possible implicants are formed. This is easier than it sounds, though, since terms in section one must be compared only with section two, then those in section two are compared with section three, and so forth, since each section differs from the next by a single binary digit. The Size Two Implicant column contains the combined binary form along with the numbers of the minterms used to create that implicant. It is also important to mark all minterms that are used to create the Size Two Implicants since allowance must be made for any not combined. Therefore, in the following table, as a minterm is used it is also struck through. Table 7.3 shows the Size Two Implicants that were found.

1's	Mntrm	Bin	Size 2
0	~~0~~	0000	000- (0,1)
1	~~1~~	0001	00-0 (0,2)
	~~2~~	0010	0-01 (1,5)
2	~~5~~	0101	-001 (1,9)
	~~6~~	0110	0-10 (2,6)
	~~9~~	1001	-010 (2,10)
	~~10~~	1010	01-1 (5,7)
3	~~7~~	0111	011- (6,7)
	~~11~~	1011	-110 (6,14)
	~~14~~	1110	10-1 (9,11)
			101- (10,11)
			1-10 (10,14)

Table 7.3: Quine-McCluskey Ex 1: Size 2 Implicants

All of the Size Two Implicants can now be combined to form Size Four Implicants (those that combine a total of four minterms). Again, it is essential to only combine those with only a single binary digit difference. For this step, the dash can be considered the same as a single binary digit, as long as it is in the same place for both implicants. Thus, -010 and -110 can be combined to $--10$, but -010 and $0-00$ cannot be combined since the dash is in different places in those numbers. It helps to match up the dashes first and then look at the binary digits. Again, as the various size-two implicants are used they are marked; but notice that a single size-four implicant actually combines four size-two implicants. Table 7.4 shows the Size Four Implicants.

1's	Mntrm	Bin	Size 2	Size 4
0	~~0~~	0000	000- (0,1)	–10 (2,10,6,14)
1	~~1~~	0001	00-0 (0,2)	
	~~2~~	0010	0-01 (1,5)	
2	~~5~~	0101	-001 (1,9)	
	~~6~~	0110	~~0-10 (2,6)~~	
	~~9~~	1001	~~-010 (2,10)~~	
	~~10~~	1010	01-1 (5,7)	
3	~~7~~	0111	011- (6,7)	
	~~11~~	1011	~~-110 (6,14)~~	
	~~14~~	1110	10-1 (9,11)	
			101- (10,11)	
			~~1-10 (10,14)~~	

Table 7.4: Quine-McCluskey Ex 1: Size 4 Implicants

None of the terms can be combined any further. All of the minterms or implicants that are not marked are *Prime Implicants*. In the table above, for example, the Size Two Implicant 000– is a Prime Implicant. The Prime Implicants will be placed in a chart and further processed in the next step.

7.1.2.2 *Step 2: The Prime Implicant Table*

A *Prime Implicant Table* can now be constructed, as in Table 7.5. The prime implicants are listed down the left side of the table, the decimal equivalent of the minterms goes across the top, and the Boolean representation of the prime implicants is listed down the right side of the table.

	0	1	2	5	6	7	9	10	11	14	
$000-\ (0,1)$	X	X									$A'B'C'$
$00-0\ (0,2)$	X		X								$A'B'D'$
$0-01\ (1,5)$		X		X							$A'C'D$
$-001\ (1,9)$		X					X				$B'C'D$
$01-1\ (5,7)$				X		X					$A'BD$
$011-\ (6,7)$					X	X					$A'BC$
$10-1\ (9,11)$							X		X		$AB'D$
$101-\ (10,11)$								X	X		$AB'C$
$--10\ (2,10,6,14)$			X		X			X		X	CD'

Table 7.5: Quine-McCluskey Ex 1: Prime Implicants

An X marks the intersection where each minterm (on the top row) is used to form one of the prime implicants (in the left column). Thus, minterm 0 (or 0000) is used to form the prime implicant $000-(0,1)$ in row one and $00-0(0,2)$ in row two.

The Essential Prime Implicants can be found by looking for columns that contain only one X. The column for minterm 14 has only one X, in the last row, $--10(2,10,6,14)$; thus, it is an Essential Prime Implicant. That means that the term in the right column for the last row, CD', must appear in the final simplified equation. However, that term also covers the columns for 2, 6, and 10; so they can be removed from the table. The Prime Implicant table is then simplified to 7.6.

	0	1	5	7	9	11	
$000-(0,1)$	X	X					$A'B'C'$
$00-0(0,2)$	X						$A'B'D'$
$0-01(1,5)$		X	X				$A'C'D$
$-001(1,9)$		X			X		$B'C'D$
$01-1(5,7)$			X	X			$A'BD$
$011-(6,7)$				X			$A'BC$
$10-1(9,11)$					X	X	$AB'D$
$101-(10,11)$						X	$AB'C$

Table 7.6: Quine-McCluskey Ex 1: 1st Iteration

The various rows can now be combined in any order the designer desires. For example, if row $10-1(9,11)$, is selected as a required implicant in the solution, then minterms 9 and 11 are accounted for in the final equation, which means that all X marked in those columns can be removed. When that is done, then, rows $101-(10,11)$ and $10-1(9,11)$ no longer have any marks in the table, and they can be removed. Table 7.7 shows the last iteration of this solution.

	0	1	5	7	
$000-(0,1)$	X	X			$A'B'C'$
$00-0(0,2)$	X				$A'B'D'$
$0-01(1,5)$		X	X		$A'C'D$
$-001(1,9)$		X			$B'C'D$
$01-1(5,7)$			X	X	$A'BD$
$011-(6,7)$				X	$A'BC$

Table 7.7: Quine-McCluskey Ex 1: 2nd Iteration

The designer next decided to select $01-1(5,7)$, $A'BD$, as a required implicant. That will include minterms 5 and 7, and those columns may be removed along with rows $01-1(5,7)$, $A'BD$, and $011-(6,7)$, $A'BC$, as shown in Table 7.8.

	0	1	
$000-\ (0,1)$	X	X	$A'B'C'$
$00-0\ (0,2)$	X		$A'B'D'$
$0-01\ (1,5)$		X	$A'C'D$
$-001\ (1,9)$		X	$B'C'D$

Table 7.8: Quine-McCluskey Ex 1: 3rd Iteration

The last two minterms (0 and 1) can be covered by the implicant $000-(0,1)$, and that also eliminates the last three rows in the chart.

The original Boolean expression, then, has been simplified from ten minterms to Equation 7.3.

$$A'B'C' + A'BD + AB'D + CD' = Y \tag{7.3}$$

7.1.3 *Example Two*

7.1.3.1 *Step 1: Create the Implicants*

Given Equation 7.4, which is a Sigma representation of a Boolean equation.

$$\int(A,B,C,D,E,F) = \sum(0,1,8,9,12,13,14,15,32,33,37,39,48,56) \tag{7.4}$$

Truth Table 7.9 shows the *True* minterm values.

Minterm	A	B	C	D	E	F
0	0	0	0	0	0	0
1	0	0	0	0	0	1
8	0	0	1	0	0	0
9	0	0	1	0	0	1
12	0	0	1	1	0	0
13	0	0	1	1	0	1
14	0	0	1	1	1	0
15	0	0	1	1	1	1
32	1	0	0	0	0	0
33	1	0	0	0	0	1
37	1	0	0	1	0	1
39	1	0	0	1	1	1
48	1	1	0	0	0	0
56	1	1	1	0	0	0

Table 7.9: Quine-McCluskey Ex 2: Minterm Table

To simplify this equation, the minterms that evaluate to *True* are placed in a minterm table so that they form sections that are easy to combine. Each section contains only the minterms that have the same number of ones. Thus, the first section contains all minterms with zero ones, the second section contains the minterms with one one, and so forth. Table 7.10 shows the rearranged truth table.

Number of 1's	Minterm	Binary
0	0	000000
1	1	000001
	8	001000
	32	100000
2	9	001001
	12	001100
	33	100001
	48	110000
3	13	001101
	14	001110
	37	100101
	56	111000
4	15	001111
	39	100111

Table 7.10: Quine-McCluskey Ex 2: Rearranged Table

Start combining minterms with other minterms to create Size Two Implicants, as in Table 7.11.

1's	Mntrm	Bin	Size 2
0	~~0~~	000000	00000- (0,1)
1	~~1~~	000001	-000000 (0,32)
	~~8~~	001000	00-000 (0,8)
	~~32~~	100000	-00001 (1,33)
2	~~9~~	001001	00-001 (1,9)
	~~12~~	001100	10000- (32,33)
	~~33~~	100001	1-0000 (32,48)
	~~48~~	110000	00100- (8,9)
3	~~13~~	001101	001-00 (8,12)
	~~14~~	001110	100-01 (33,37)
	~~37~~	100101	001-01 (9,13)
	~~56~~	111000	00110- (12,13)
4	~~15~~	001111	0011-0 (12,14)
	~~39~~	100111	11-000 (48,56)
			1001-1 (37,39)
			0011-1 (13,15)
			00111- (14,15)

Table 7.11: Quine-McCluskey Ex 2: Size Two Implicants

All of the Size Two Implicants can now be combined to form Size Four Implicants, as in Table 7.12.

1's	Mntrm	Bin	Size 2	Size 4
0	~~0~~	000000	~~00000- (0,1)~~	-0000- (0,1,32,33)
1	~~1~~	000001	~~-00000 (0,32)~~	00-00- (0,1,8,9)
	~~8~~	001000	~~00-000 (0,8)~~	001-0- (8,9,12,13)
	~~32~~	100000	~~-00001 (1,33)~~	0011– (12,13,14,15)
2	~~9~~	001001	~~00-001 (1,9)~~	
	~~12~~	001100	~~10000- (32,33)~~	
	~~33~~	100001	1-0000 (32,48)	
	~~48~~	110000	~~00100- (8,9)~~	
3	~~13~~	001101	~~001-00 (8,12)~~	
	~~14~~	001110	100-01 (33,37)	
	~~37~~	100101	~~001-01 (9,13)~~	
	~~56~~	111000	~~00110- (12,13)~~	
4	~~15~~	001111	~~0011-0 (12,14)~~	
	~~39~~	100111	11-000 (48,56)	
			1001-1 (37,39)	
			~~0011-1 (13,15)~~	
			~~00111- (14,15)~~	

Table 7.12: Quine-McCluskey Ex 2: Size 4 Implicants

None of the terms can be combined any further. All of the minterms or implicants that are not struck through are *Prime Implicants*. In the table above, for example, $1-0000$ is a Prime Implicant. The Prime Implicants are next placed in a table and further processed.

7.1.3.2 *Step 2: The Prime Implicant Table*

A *Prime Implicant Table* can now be constructed, as in Table 7.13. The prime implicants are listed down the left side of the table, the decimal equivalent of the minterms goes across the top, and the Boolean representation of the prime implicants is listed down the right side of the table.

	0	1	8	9	12	13	14	15	32	33	37	39	48	56	
11 – 000 (48, 56)													X	X	ABD′D′F′
00 – 00 – (0, 1, 8, 9)	X	X	X	X											A′B′D′E′
1001 – 1 (37, 39)											X	X			AB′C′DF
1 – 0000 (32, 48)									X				X		AC′D′E′F′
0011 – – (12, 13, 14, 15)					X	X	X	X							A′B′CD
–0000 – (0, 1, 32, 33)	X	X							X	X					B′C′D′E′
001 – 0 – (8, 9, 12, 13)			X	X	X	X									A′B′CE′
100 – 01 (33, 37)										X	X				AB′C′E′F

Table 7.13: Quine-McCluskey Ex 2: Prime Implicants

In the above table, there are four columns that contain only one *X*: 14, 15, 39, and 56. The rows that intersect the columns at that mark are *Essential Prime Inplicants*, and their Boolean Expressions must appear in the final equation. Therefore, the final equation will contain, at a minimum: A′B′CD (row 5, covers minterms 14 and 15), AB′C′DF (row 3, covers minterm 39), and ABD′E′F′ (row 1, covers minterm 56). Since those expressions are in the final equation, the rows that contain those expressions can be removed from the chart in order to make further analysis less confusing.

Also, because the rows with Essential Prime Implicants are contained in the final equation, other minterms marked by those rows are covered and need no further consideration. For example, minterm 48 is covered by row one (used for minterm 56), so column 48 can be removed from the table. In a similar fashion, columns 12, 13, and 37 are covered by other minterms, so they can be removed from the table. Table 7.14 shows the next iteration of this process.

	0	1	8	9	32	33	
00 – 00 – (0, 1, 8, 9)	X	X	X	X			A′B′D′E′
1 – 0000 (32, 48)					X		AC′D′E′F′
–0000 – (0, 1, 32, 33)	X	X			X	X	B′C′D′E′
001 – 0 – (8, 9, 12, 13)			X	X			A′B′CE′
100 – 01 (33, 37)						X	AB′C′E′F

Table 7.14: Quine-McCluskey Ex 2: 1st Iteration

The circuit designer can select the next term to include in the final equation from any of the five rows still remaining in the chart; however, the first term (00 – 00–, or A′B′D′E′) would eliminate four columns, so that would be a logical next choice. When that term is selected for the final equation, then row one, 00 – 00–, can be removed from the chart; and columns 0, 1, 8, and 9 can be removed since those minterms are covered.

The minterms marked for row $001-0-(8,9,12,13)$ are also covered, so this row can be removed. Table 7.15 shows the next iteration.

	32	33	
$1-0000\ (32,48)$	X		$AC'D'E'F'$
$-0000-\ (0,1,32,33)$	X	X	$B'C'D'E'$
$100-01\ (33,37)$		X	$AB'C'E'F$

Table 7.15: Quine-McCluskey Ex 2: 2nd Iteration

For the next simplification, row $-0000-$ is selected since that would also cover the minterms that are marked for all remaining rows. Thus, the expression $B'C'D'E'$ will become part of the final equation.

When the analysis is completed, the original equation (7.4), which contained 14 minterms, is simplified into Equation 7.5, which contains only five terms.

$$ABD'E'F' + A'B'D'E' + AB'C'DF + A'B'CD + B'C'D'E' = Y \tag{7.5}$$

7.1.4 *Summary*

While the Quine–McCluskey method is useful for large Boolean expressions containing multiple inputs, it is also tedious and prone to error when done by hand. Also, there are some Boolean expressions (called "Cyclic" and "Semi-Cyclic" Primes) that do not reduce using this method. Finally, both Karnaugh maps and Quine-McCluskey methods become very complex when more than one output is required of a circuit. Fortunately, many automated tools are available to simplify Boolean expressions using advanced mathematical techniques.

7.1.5 *Practice Problems*

The following problems are presented as practice for using the Quine-McClusky method to simplify a Boolean expression. Note: designers can select different Prime Implicants so the simplified expression could vary from what is presented below.

1	Expression	$\int(A, B, C, D) = \sum(0, 1, 2, 5, 6, 7, 9, 10, 11, 14)$
	Simplified	$A'B'C' + A'BD + AB'D + CD'$
2	Exression	$\int(A, B, C, D) = \sum(0, 1, 2, 3, 6, 7, 8, 9, 14, 15)$
	Simplified	$A'C + BC + B'C'$
3	Exression	$\int(A, B, C, D) = \sum(1, 5, 7, 8, 9, 10, 11, 13, 15)$
	Simplified	$C'D + AB' + BD$
3	Exression	$\int(A, B, C, D, E) = \sum(0, 4, 8, 9, 10, 11, 12, 13, 14, 15, 16, 20, 24, 28)$
	Simplified	$A'B + D'E'$

Table 7.16: Quine-McCluskey Practice Problems

7.2 AUTOMATED TOOLS

7.2.1 *KARMA*

There are numerous automated tools available to aid in simplifying complex Boolean equations. Many of the tools are quite expensive and intended for professionals working full time in large companies; but others are inexpensive, or even free of charge, and are more than adequate for student use. One free tool, a Java application named KARnaugh MAp simplifier (KARMA), can be downloaded from http://bit.ly/2mcXVp9 and installed on a local computer, though the website also has a free online version that can be used without installation. KARMA helps to simplify complex Boolean expressions using both Karnaugh Maps and Quine-McCluskey methods. KARMA is a good program with numerous benefits.

When first started, KARMA looks like Figure 7.1.

Figure 7.1: KARMA Start Screen

The right side of the screen contains a row of tools available in KARMA and the main part of the screen is a canvas where most of the work is done. The following tools are available:

- **Logic2Logic**. Converts between two different logical representations of data; for example, a Truth Table can be converted to Boolean expressions.
- **Logic Equivalence**. Compares two functions and determines if they are equivalent; for example, a truth table can be compared with a SOP expression to see if they are the same.

- **Logic Probability**. Calculates the probability of any one outcome for a given Boolean expression.
- **Karnaugh Map**. Analyzes a Karnaugh map and returns the Minimized Expression.
- **KM Teaching Mode**. Provides drill and practice with Karnaugh maps; for example, finding adjacent minterms on a 6-variable map.
- **SOP and POS**. Finds the SOP and POS expressions for a given function.
- **Exclusive-OR**. Uses XOR gates to simplify an expression.
- **Multiplexer-Based**. Realizes a function using multiplexers.
- **Factorization**. Factors Boolean expressions.
- **About**. Information about Karma.

For this lesson, only the Karnaugh Map analyzer will be used, and the initial screen for that function is illustrated in Figure 7.2.

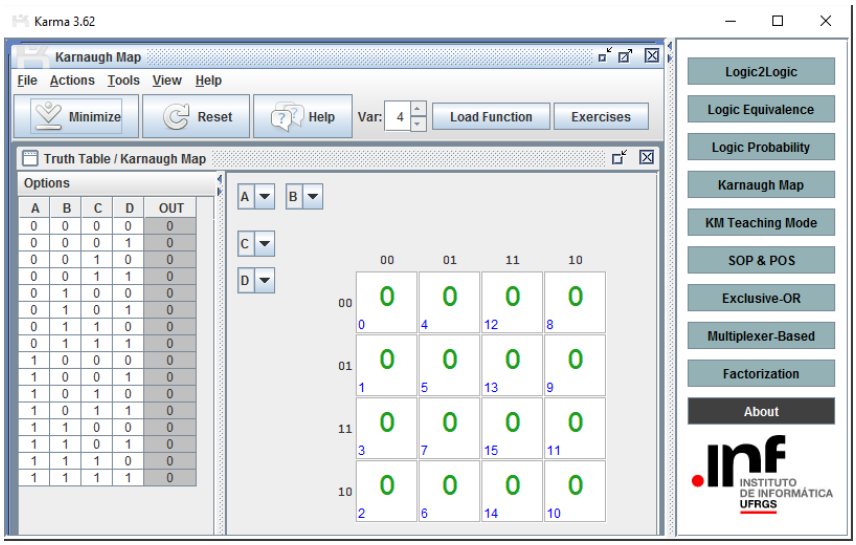

Figure 7.2: Karnaugh Map Screen

7.2.1.1 *Data Entry*

When using KARMA, the first step is to input some sort of information about the circuit to be analyzed. That information can be entered in several different formats, but the most common for this class would be either a truth table or a Boolean expression.

To enter the initial data, click the Load Function button at the top of the canvas.

By default, the Load Function screen opens with a blank screen. In the lower left corner of the Load Function window, the Source Format for the input data can be selected. There is a template available for each of the different source formats; and that template can be used to help with data entry. The best way to work with KARMA is to click the "Templates" button and select the data format being used. In Figure 7.3, the "Expression 1" template was selected:

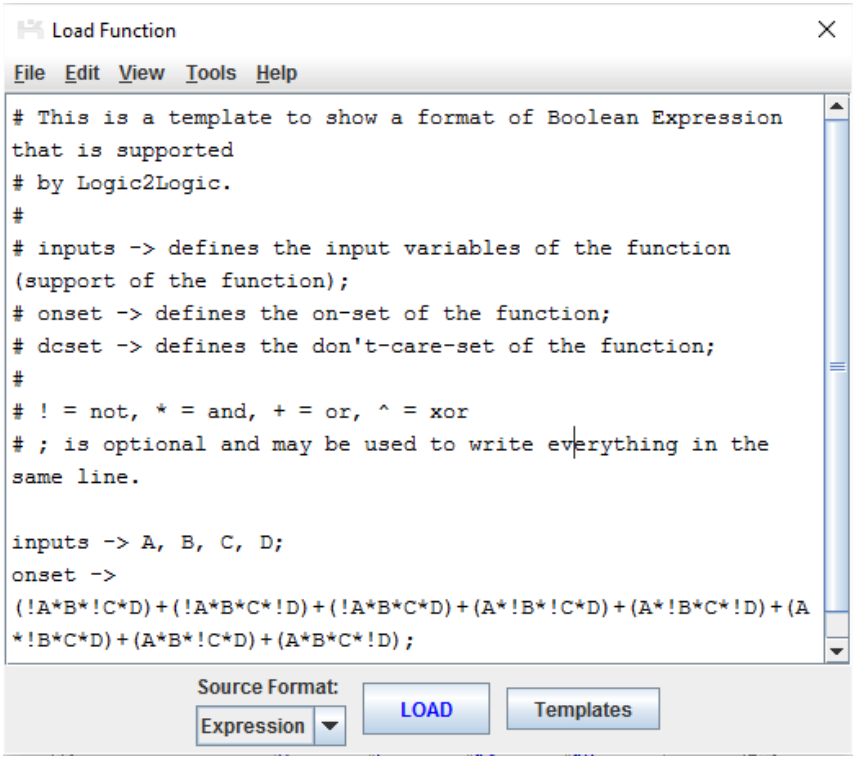

Figure 7.3: The Expression 1 Template

The designer would replace the "inputs" and "onset" lines with information for the circuit being simplified. Once the source data are entered into this window, click the "Load" button at the bottom of the window to load the data into KARMA.

7.2.1.2 *Data Source Formats*

Karma works with input data in any of six different formats: Boolean Expression, Truth Table, Integer, Minterms, Berkeley Logic Interchange Format (*BLIF*), and Binary Decision Diagram (*BDD*). *BLIF* and *BDD* are programming tools that are beyond the scope of this lesson and will not be covered.

EXPRESSION Boolean expressions can be defined in Karma using the following format.

#Sample Expression

```
(!x1*!x2*!x4)+(!x1*x2*!x3)+(x1*!x4*!x5)+(x1*x3*x4)
```

Notes:

- Any line that starts with a hash tag ("#") is a comment and will be ignored by Karma.
- "Not" is indicated by a leading exclamation mark. Thus "!x1" is the same as X1'.
- All operations are explicit. In real-number algebra the phrase "AB" is understood to be "A*B". However, in KARMA, since variable names can be more than one character long, all operations must be explicitly stated. AND is indicated by an asterisk and OR is indicated by a plus sign.
- No space is left between operations.

TRUTH TABLE A truth table can be defined in KARMA using the following format.

```
#Sample Truth Table
inputs -> X, Y, Z
000 : 1
001 : 1
010 : 0
011 : 0
100 : 0
101 : 1
110 : 0
111 : 1
```

Notes:

- Any line that starts with a hash tag ("#") is a comment and will be ignored by KARMA.
- The various inputs are named before they are used. In the example, there are three inputs: X, Y, and Z.
- Each row in the truth table is shown, along with the output required. So, in the example above, an input of 000 should yield an output of 1.
- An output of "-" is permitted and means "don't care."

INTEGER In Karma, an integer can be used to define the outputs of the truth table, so it is "shorthand" for an entire truth table input. Following is the example of the "integer" type input.

```
#Sample Integer Input
inputs -> A, B, C, D
onset -> E81A base 16
```

Notes:

- Any line that starts with a hash mark ("#") is a comment and will be ignored by Karma.
- Input variables are defined first. In this example, there are four inputs: A, B, C, and D.
- The "onset" line indicates what combinations of inputs should yield a True on a truth table.

In the example, the number E81A is a hexadecimal number that is written like this in binary:

```
1110 1000 0001 1010
  E    8    1    A
```

The least significant bit of the binary number, 0 in this example, corresponds to the output of the first row in the truth table; thus, it is false. Each bit to the left of the least significant bit corresponds to the next row, counting from 0000 to 1111. Following is the truth table generated by the hexadecimal integer E81A.

Inputs				Output
A	B	C	D	Q
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Table 7.17: Truth Table for E81A Output

The "Output" column contains the binary integer 1110 1000 0001 1010 (or E81Ah) from bottom to top.

TERMS Data input can be defined by using the minterms for the Boolean expression. Following is an example minterm input.

```
#Sample Minterms
inputs -> A, B, C, D
onset -> 0, 1, 2, 3, 5, 10
```

Notes:

- Any line that starts with a hash mark ("#") is a comment and will be ignored by KARMA.
- The inputs, A, B, C, and D, are defined first.
- The "onset" line indicates the minterms that yield a "true" output.
- This is similar to a SOP Sigma expression, and the digits in that expression could be directly entered on the onset line. For example, the onset line above would have been generated from this Sigma expression:

$$\int(A, B, C, D) = \sum(0, 1, 2, 3, 5, 10)$$

7.2.1.3 *Truth Table and Karnaugh Map Input*

While Karma will accept a number of different input methods, as described above, one of the easiest to use is the Truth Table and its related Karnaugh Map, and these are displayed by default when the *Karnaugh Map* function is selected. The value of any of the cells in the *Out* column in the Truth Table, or cells in the Karnaugh Map, and can be cycled through 0, 1, and "don't care" (indicated by a dash) on each click of the mouse in the cell. The Truth Table and Karnaugh Map are synchronized as cells are clicked. The number of input variables can be adjusted by changing the *Var* setting at the top of the screen. Also, the placement of those variables on the Karnaugh Map can be adjusted as desired.

7.2.1.4 *Solution*

To simplify the Karnaugh Map, click the Minimize button. A number of windows will pop up (illustrated in Figure 7.4), each showing the circuit simplification in a slightly different way. Note: this Boolean expression was entered to generate the following illustrations: $A'C + A'B + AB'C' + B'C'D'$.

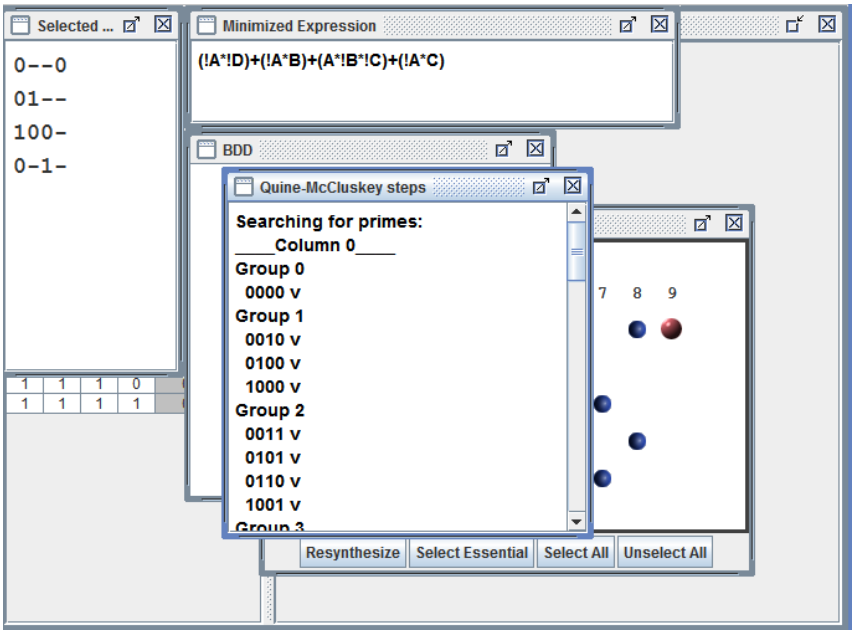

Figure 7.4: A KARMA Solution

BOOLEAN EXPRESSION The minimized Boolean expression is shown in Figure 7.5.

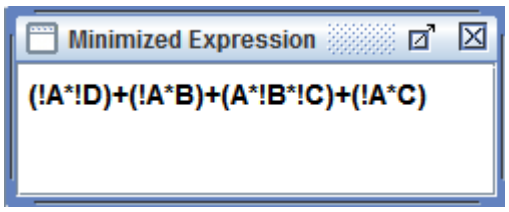

Figure 7.5: The Minimized Boolean Expression

In this solution, a NOT term is identified by a leading exclamation point; thus, the minimized expression is: $A'D' + AB'C' + A'C + A'B$.

BDDEIRO The *BDDeiro* window is a form of Binary Decision Diagram (*BDD*). The decision tree for the minimized expression is represented graphically in Figure 7.6.

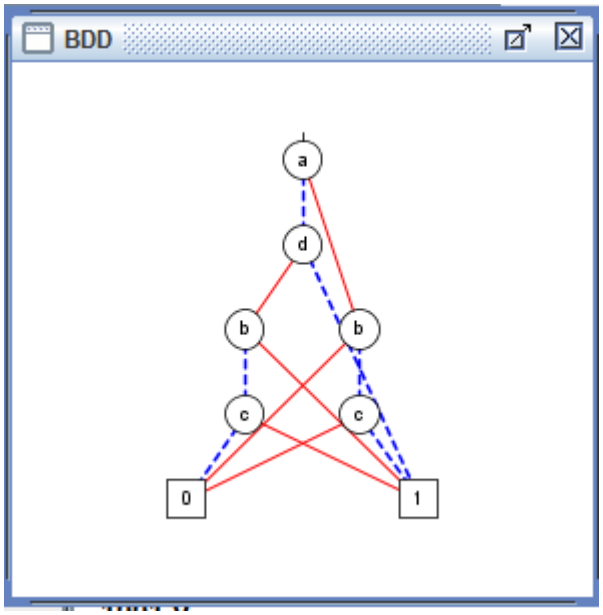

Figure 7.6: The BDDeiro Solution

The top node represents input *a*, which can either be true or false. If false, then follow the blue dotted line down to node *d*, which can also be either true or false. If *d* is false, follow the blue dotted line down to the *1* output. This would mean that one True output for this circuit is $A'D'$, which can be verified as one of the outputs in the minimized expression.

Starting at the top again, if *a* is true, then follow the solid red line down to *b*. It that is true, then follow the red solid line down to the *o* output. The expression AB is false and does not appear in the minimized solution. In a similar way, all four True outputs, and three false outputs, can be traced from the top to bottom of the diagram.

QUINE-MCCLUSKEY Karma includes complete Quine-McCluskey solution data. Several tables display the various implicants and show how they are derived.

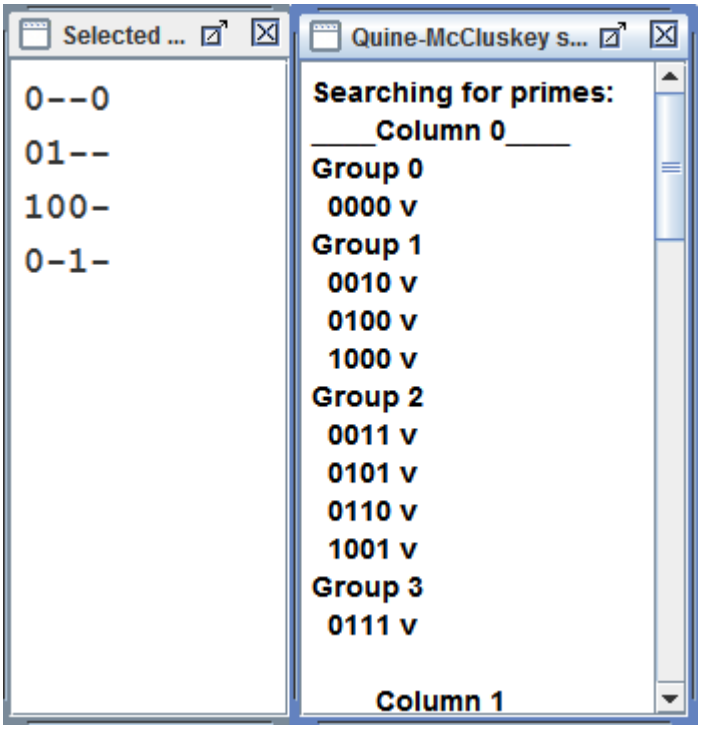

Figure 7.7: Quine-McCluskey Solution

Karma also displays the Covering Table for a Quine-McCluskey solution. Each of the minterms (down the left column) can be turned on or off by clicking on it. The smaller blue balls in the table indicate prime implicants and the larger red balls (if any) indicate essential prime implicants. Because this table is interactive, various different solutions can be attempted by clicking some of the colored markers to achieve the best possible simplification.

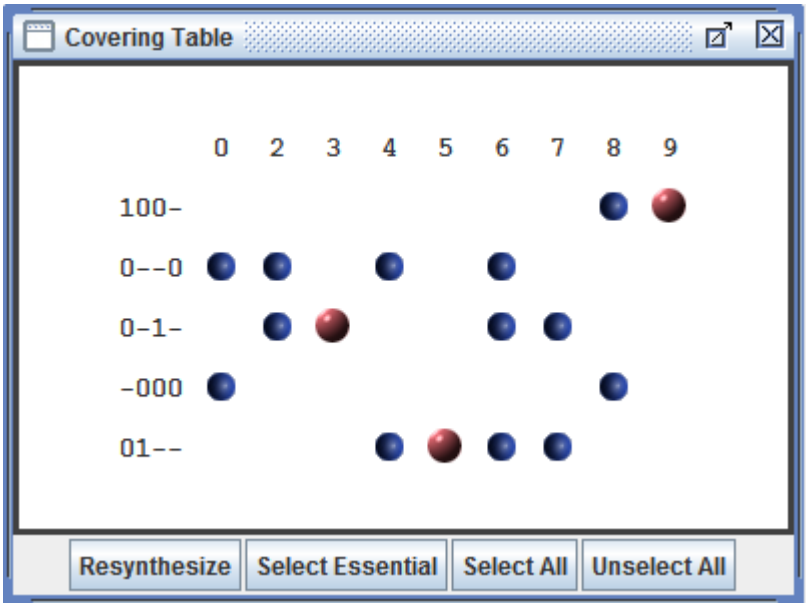

Figure 7.8: Selecting Implicants

7.2.1.5 *Practice Problems*

The following problems are presented as practice for usingKarma to simplify a Boolean expression. Note: designers can select different Prime Implicants so the simplified expression could vary from what is presented below.

1	Expression	$\int(A,B,C,D) = \sum(5,6,7,9,10,11,13,14)$
	Simplified	$BC'D + A'BC + ACD' + AB'D$
2	Exression	$A'BC'D + A'BCD' + A'BCD + AB'C'D + AB'CD' + AB'CD + AB'CD + ABC'D + ABCD'$
	Simplified	$BC'D + A'BC + ACD' + AB'D$
3	Exression	4-variable Karnaugh Map where cells 5,6,7,9,10 are True and 13,14 are Don't Care
	Simplified	$BC'D + AC'D + A'BC + ACD'$
3	Exression	$\int(A,B,C,D,E) = \sum(0,3,4,12,13,14,15,24,25,28,29,30)$
	Simplified	$ABD' + A'B'C'DE + BCE' + A'BC + A'B'D'E'$

Table 7.18: KARMA Practice Problems

7.2.2 *32x8*

One online site of interest is http://www.32x8.com/. This site permits visitors to set up a truth table with two to eight variables and then will create a simplified Boolean circuit for that truth table.