Unit 19 Reed-Muller Canonical Form

- A function is in the Reed-Muller canonical form when it is expressed as an eXclusive-OR of the products of uncomplemented variables.
- Conversion from Sum-of-Products to Reed-Muller form is done by replacing the complemented terms in the SOP using $\overline{A} = 1 \oplus A$

A point in three dimensional space can be specified in the Cartesian coordinate system, (x, y, z). In some circumstances, it may be better to specify the point in other coordinate systems such as the spherical coordinate system, (r, θ, ϕ), or the cylindrical coordinate system, (r, θ, z). It is still the same point but just specified relative to a different frame of reference. The characteristics of the axes of each of these systems is that the axes are orthogonal, that is a movement along one axis does not change the coordinates on the other axes. There are also various formulae for converting from one coordinate system to another.

In a similar way it is possible to specify the coordinates of a logical function or state in a Boolean AND/OR form. The same logical function can also be specified in other logic coordinate systems. One such system is the AND/XOR system and this system has advantages for certain purposes. There are also algorithms for converting from the AND/OR system to the AND/XOR system. In this Unit and Unit 20 we will examine the advantages of the AND/XOR system and the associated conversion and logic minimization algorithms. It must be pointed out that theoretical work is continuing on the properties of this system at the present time so that these units only cover some of the results and that there may be significant developments in the future which are not mentioned here.

In performing Boolean algebraic operations using the eXclusive-OR operator, $\oplus$, the following rules apply:

$$\begin{aligned}
0 \oplus 0 &= 0 \\
1 \oplus 1 &= 0 \\
0 \oplus 1 &= 1 \\
A \oplus A &= 0 \\
A \oplus \overline{A} &= 1 \\
1 \oplus A &= \overline{A} \\
A + B &= A \oplus B \oplus AB = A \oplus \overline{A}B = B \oplus \overline{B}A \\
A(B \oplus C) &= AB \oplus AC
\end{aligned}$$

We have seen in earlier units how a Boolean function can be expressed as a sum of products using the AND/OR logic gate array. The function can then be described very succinctly using the minterm formulation. For instance, if we have a Boolean Sum-of-Products form of the function:

$$f(ABCD) = \overline{A}B\overline{C}D + \overline{A}BCD + A\overline{B}C\overline{D} + ABCD$$

then this can be put into the minterm form as:

$$f(ABCD) = \Sigma m(5, 7, 10, 15)$$

The difficulty about this SOP form is that each of the variables A, B, C, D can appear in both the complemented form and also in the uncomplemented form so that there are effectively twice the number of primary input variables in the expression.

If we remember that only one of the AND terms can be true at any time, that is the minterms are mutually exclusive, then only one of the 2^n rows of the truth table can hold at one time. Therefore it is possible to rewrite this function using eXclusive-OR instead of OR operators to obtain:

$$f(ABCD) = \overline{A}B\overline{C}D \oplus \overline{A}BCD \oplus A\overline{B}C\overline{D} \oplus ABCD$$

In our example we still have a possible $2 \times n = 8$ distinct inputs but the number can be reduced by using the identity:

$$\overline{A} = 1 \oplus A$$

so that all of the complemented terms can be replaced by uncomplemented terms to give:

$$
\begin{aligned}
f(ABCD) &= \overline{A}B\overline{C}D \oplus \overline{A}BCD \oplus A\overline{B}C\overline{D} \oplus ABCD \\
&= (1 \oplus A)B(1 \oplus C)D \\
&\quad \oplus (1 \oplus A)BCD \\
&\quad \oplus A(1 \oplus B)C(1 \oplus D) \\
&\quad \oplus ABCD \\
&= BD \oplus BCD \oplus ABD \oplus ABCD \\
&\quad \oplus BCD \oplus ABCD \\
&\quad \oplus AC \oplus ACD \oplus ABC \oplus ABCD \\
&\quad \oplus ABCD \\
&= AC \oplus BD \oplus ABC \oplus ABD \oplus ACD \oplus BCD \oplus BCD \\
&\quad \oplus ABCD \oplus ABCD \oplus ABCD \oplus ABCD \\
&= AC \oplus BD \oplus ABC \oplus ABD \oplus ACD \\
&\quad \oplus (1 \oplus 1)BCD \oplus (1 \oplus 1 \oplus 1 \oplus 1)ABCD \\
&= AC \oplus BD \oplus ABC \oplus ABD \oplus ACD \oplus 0
\end{aligned}
$$

This AND/XOR form of the function is now said to be in **Reed-Muller canonical form**. Note that while the function is in either the AND/OR Boolean form or the equivalent minterm list form each of the n inputs appears in each of the AND terms of the expression. However, when the function is put into the Reed-Muller form not all of the inputs appear explicitly in all of the terms of the Reed-Muller canonical form and also none of the terms are in the complemented form ($\overline{A}$).

The equivalence of these two forms (the SOP form and the R-M form) for this example can be checked using a QuickBasic program such as:

```
100 INPUT a, b, c, d
a = -a: b = -b: c = -c: d = -d
REM logic 1 is stored as -1 in QuickBasic
z = ((NOT a) AND b AND (NOT c) AND d) XOR ((NOT a) AND b
     AND c AND d) XOR (a AND (NOT b) AND c AND (NOT d))
     XOR (a AND b AND c AND d)
p = (a AND c) XOR (b AND d) XOR (a AND b AND c) XOR
     (a AND b AND d) XOR (a AND c AND d) XOR 0
PRINT z, p
GOTO 100
```

The reverse problem, that of going from a Reed-Muller form to a Boolean AND/OR form which is suitable for specification in minterm form, is accomplished, in principle, by remembering that the minterm form is a canonical form in which all of the variables, whether complemented or uncomplemented, appear in each of the terms. Once the expression is in this canonical form, the XOR, $\oplus$, can be replaced by the OR, $+$, in the expression since the canonical terms are mutually exclusive. The basis of the process is that in general $X \oplus X = 0$ so that such duplicated terms can be inserted without altering the function. The term expansion process is best illustrated by the following example conversion:

$$\begin{aligned} f(AB) &= A \oplus B \\ &= A \oplus AB \oplus AB \oplus B \\ &= A(1 \oplus B) \oplus B(1 \oplus A) \\ &= A\overline{B} \oplus B\overline{A} \\ &= A\overline{B} \oplus \overline{A}B \\ &= A\overline{B} + \overline{A}B \\ &= \Sigma m(1,2) \end{aligned}$$

In order to generalize the conversion process from the Boolean AND/OR form to the Reed-Muller form we first assign coefficients a_k and c_k which take the values 0 or 1 and where the coefficients combine using mod(2) arithmetic rules, ie. $0+0=0$, $0+1=1$, $1+0=1$ and $1+1=0$. Then a function of one variable, A, can be written:

$$\begin{aligned} f(A) &= a_0\overline{A} + a_1A \\ &= a_0\overline{A} \oplus a_1A \\ &= a_0(1 \oplus A) \oplus a_1A \\ &= a_0 \oplus a_0A \oplus a_1A \\ &= a_0 \oplus (a_0 + a_1)A \\ &= c_0 \oplus c_1A \\ \text{where } c_0 &= a_0 \\ \text{and } c_1 &= a_0 + a_1 \\ \text{or in matrix form } \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \end{aligned}$$

For the two variable function:

$$\begin{aligned} f(AB) &= a_0\overline{AB} + a_1\overline{A}B + a_2A\overline{B} + a_3AB \\ &= c_0 \oplus c_1B \oplus c_2A \oplus c_3AB \\ \text{where} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} . \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} \\ \text{or} \qquad \mathbf{c} &= \mathbf{T.a} \end{aligned}$$

In general the $\mathbf{T_n}$ matrix is formed recursively from the $\mathbf{T_{n-1}}$ matrix by:

$$\mathbf{T_n} = \begin{pmatrix} \mathbf{T_{n-1}} & \mathbf{0} \\ \mathbf{T_{n-1}} & \mathbf{T_{n-1}} \end{pmatrix} \quad \text{and} \quad \mathbf{T_1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

It can be shown that this transformation matrix also serves as the reverse transformation matrix, that is, the matrix is self-inverse so that the reverse transformations are very straightforward:

$$\mathbf{c} = \mathbf{T.a} \quad \text{and} \quad \mathbf{a} = \mathbf{T.c}$$

However, be warned:

$$\mathbf{T_n^{-1}.T_n} = \mathbf{T_n.T_n} = \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix is true only when the arithmetic is performed using modulus(2) arithmetic.

One practical difficulty is that the $\mathbf{T_n}$ matrix for transformations on n input variables is of dimension $2^n \times 2^n$ which, for even quite small numbers of variables, may be greater than the capacity of many computer systems.

Let us now look at how an expression such as:

$$A \oplus B \oplus C \oplus D$$

can be implemented using XOR gates. The most significant feature of an XOR gate is that there can only be two inputs. This is very different from an AND or OR gate which may have many inputs. It is therefore not possible to implement a multi input XOR function with a single gate. Instead a hierarchical structure is necessary. There are two possible structures which are illustrated in Figure 19.1.

The structures are either a binary division type tree structure or a cascade structure but when the truth table for these gate structures are prepared and the Karnaugh map is drawn up, a similar Karnaugh map structure, as shown below, is obtained for both types of structure.

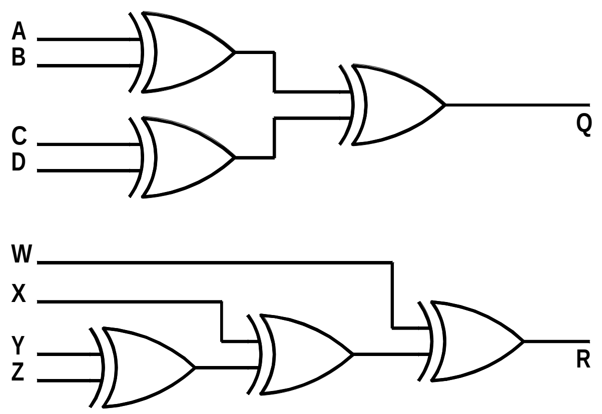

Figure 19.1: Binary tree structure and cascade structure.

	$\overline{A}.\overline{B}$	$\overline{A}.B$	$A.B$	$A.\overline{B}$
$\overline{C}.\overline{D}$	0	1	0	1
$\overline{C}.D$	1	0	1	0
$C.D$	0	1	0	1
$C.\overline{D}$	1	0	1	0

Inspection of this Karnaugh map or of the similar Karnaugh map which is obtained for the lower logic gate circuit in Figure 19.1 will show that the output from the circuit is a logic 1 when there are an odd number of 1's present at the inputs and the output is a logic 0 when there are an even number of 1's present at the inputs. The multi input XOR gate structure can therefore be considered to be a parity generating circuit.

If we want to obtain a circuit which will implement an AND/XOR function of variables x_j for $0 \leq j \leq n-1$ such as:

$$f(x_{n-1}...x_0) = 1 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1x_4 \oplus x_3x_4 \oplus x_1x_2x_3 \oplus x_2x_3x_4$$

then this is most simply implemented as a circuit employing a bus structure as shown in Figure 19.2 in which $x_0 = 1$ and the other AND gate combinations are picked off the bus and then combined into the sequential version of the XOR array. The advantage of this type of structure is that it can be fabricated as a standard integrated circuit and then customized by making or breaking the interrconnects between the input bus of x_1 to x_n inputs and the n inputs to each of the AND gates. As we will see in the unit on Programmable Logic Devices, this is a very cost effective way of obtaining Application Specific Integrated Circuits (ASICs).

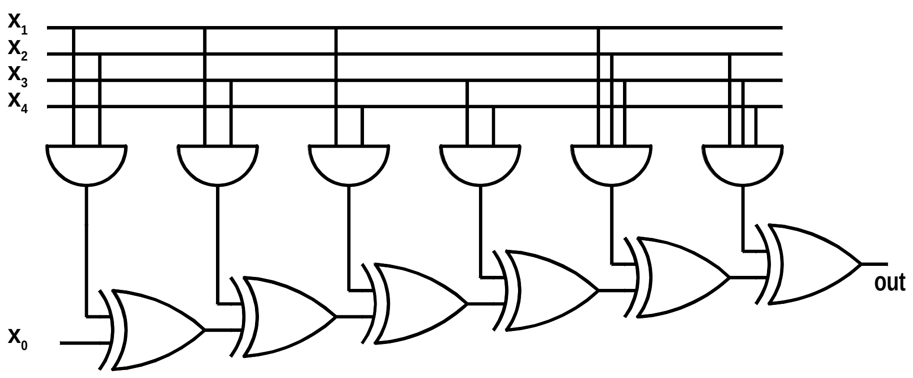

Figure 19.2: Circuit implementation of a Reed-Muller AND/XOR function.

19.1 References

Green, David (1986), *Modern Logic Design*, Addison-Wesley

Green DH. (1994), *Dual forms of Reed-Muller expansions*, IEE Proc.-Comput. Digit. Tech. 141, (3), 184-192.

Haomin Wu, Perkowski MA. Xiaoqiang Zeng, Nan Zhuang. (1996), *Generalized partially mixed polarity Reed-Muller expansion and its fast computation*, IEEE Trans. on Computers, 45, (9), 1084-1088.

Maxfield C. (1996), *Hug an XOR gate today: an introduction to Reed-Muller logic*, EDN (Electronic Design News). March 1, (www.ednmag.com).

Maxfield C. (1996), *A Reed-Muller extraction utility*, EDN (Electronic Design News). April 11, (www.ednmag.com).

Tran A (1987), *Graphical method for the conversion of minterms to Reed-Muller coefficients and the minimization of exclusive-OR switching functions* , IEE Proc. 134, E, (2) 93-99.

Tsai C C, Marek-Sadowska M (1996), *Generalized Reed-Muller forms as a tool to detect symmetries*, IEEE Trans. on Computers, 45, (1), 33-40.

19.2 Problems

19.1 Show, by drawing up the truth tables, that the following rules are correct:

$$\begin{aligned} A + B &= A \oplus B \oplus AB = A \oplus \overline{A}B = B \oplus \overline{B}A \\ A(B \oplus C) &= AB \oplus AC \end{aligned}$$

19.2 Explain why it is possible to replace the + operators in the Sum-of-Product expression by $\oplus$ eXclusive-OR operators. Justify your answer by worked truth table examples.

19.3 Obtain the truth tables for the circuits shown in Figure 19.1 (a) and (b). Draw the Karnaugh map which represents these circuits. What is the most significant feature of the Karnaugh maps?

19.4 Show, by explicit calculation using a truth table structure similar to that used in Example 5.2, that:

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) = C \oplus (B \oplus A)$$

19.5 What are the Reed-Muller terms which will give Karnaugh maps which contain the maximum number of 1's and which cannot be further simplified?

19.6 The propagation delay time for a logic gate integrated circuit is the delay between the application of a changed signal at the input and the appearance of the resulting changed signal at the output. If the delay for an XOR gate is 10 ns, calculate the propagation delay time for each of the two XOR arrays shown in Figure 19.4. Obtain formulae for the delays for the case of n inputs for each of the two configurations.

19.7 Show that the Reed-Muller form for the minterm function described by:

$$f(x_2, x_1, x_0) = \Sigma m(0)$$

is given by:

$$f(x_2, x_1, x_0) = 1 \oplus x_0 \oplus x_1 \oplus x_2 \oplus x_0x_1 \oplus x_0x_2 \oplus x_1x_2 \oplus x_0x_1x_2$$

19.8 Show that the Reed-Muller form for the minterm function described by:

$$f(x_2, x_1, x_0) = \Sigma m(0, 3, 5, 6)$$

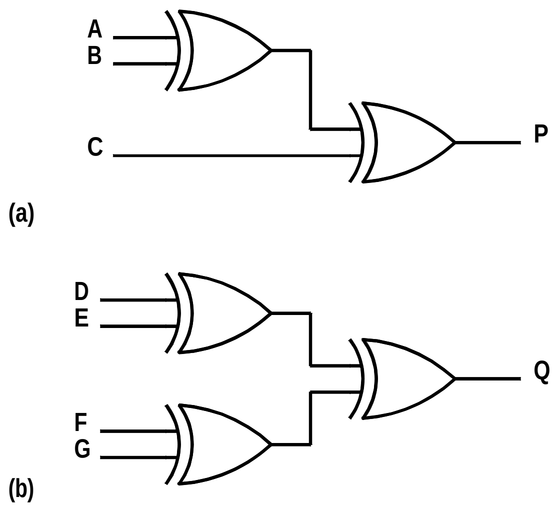

Figure 19.3: XOR gate array

is given by:

$$f(x_2, x_1, x_0) = 1 \oplus x_0 \oplus x_1 \oplus x_2$$

19.9 Convert the Reed-Muller form for an expression given by:

$$f(x_2, x_1, x_0) = x_0 \oplus x_2 \oplus x_1x_0 \oplus x_2x_1 \oplus x_2x_1x_0$$

to the minterm form. Warning — If you use the $\mathbf{T_3}$ matrix to solve this problem remember that the arithmetic is carried out modulus(2).

19.10 Calculate the terms on the extended Karnaugh map template on page 90 (Unit 15) which satisfy:

$$Q = A \oplus B \oplus C \oplus D \oplus E \oplus F$$

and describe the resulting pattern in words.

19.11 Is it necessary to first convert a Sum of Products expression into the Canonical SoP form before replacing the + operators by $\oplus$ operators. In other words, is the following expression valid?

$$A.\overline{B} + B.\overline{C} = A.\overline{B} \oplus B.\overline{C}$$