Optimal Polarity for Dual Reed-Muller Expressions

Khalid. Faraj
Computer Science
Wajdi Institute of Technology
Jerusalem, Mount of Olives. P.O.Box 19014
Palestine
http://www.wit.edu.ps

A.E.A. Almaini
School of Engineering
Napier University
Edinburgh
Scotland
www.napier.ac.uk

Abstract In this paper we present two algorithms, which can be used in converting between product of sum (POS) and fixed polarity dual Reed_Muller (FPDRM) and find the optimal polarity for large number of variables. The first algorithm is used to compute the coefficients of FPDRM directly from the truth table of POS. This algorithm is also used to compute the coefficients of POS from FPDRM. The second algorithm will find the optimal polarity among the 2^n different polarities for large n-variable functions, without generating all of the polarity sets. This algorithm is based on separating the truth vector of POS and the use of sparse techniques, which will lead to the optimal polarity. Time efficiency and computing speed are thus achieved in this technique.

Keywords:-Product of Sum, Dual Reed_Muller, Maxterms,Boolean functions.

1. Introduction

The increasing complexity of chip designs and the continuous development of smaller size fabrication processes present new challenges to the existing tools. Future synthesis tools are required to handle millions of gates in a realistic time. Computer-Aided Design (CAD) tools became critical for design and verification of Very Large Scale Integrated (VLSI) digital circuits. Up to now, most of the research has focused on developing algorithms for AND/OR or NAND/NOR circuits. An alternative description of a Boolean function is Reed-Muller expansion [1, 2]. It employs modulo-2 arithmetic and is also unique and canonical for a given Boolean function. The application of XOR/AND and XNOR/OR gates has some advantages over other implementations. In practice, it is well known that many useful circuits such as arithmetic units and parity checkers are heavily XOR oriented and it is more economical to implement their modulo-2 expressions [3-6]. Some authors [7, 8] even conjecture that it is generally more economical to base logic design on modulo-2 expressions rather than conventional OR expressions. Recent progress in circuit technology makes the use of OR/XNOR gates feasible, especially with the development of the new technologies and the arrival of various programmable gate array (FPGA) devices. A major other characteristic of the XNOR logic is the numerous possible canonical representations of switching functions it provides.

There are several kinds of OR/ XNOR circuits. The FPDRM is one of the canonical OR/ XNOR expressions. FPDRMs are a generalization of Positive Polarity Reed-Muller expressions (PPDRM). A PPDRM is unique for a completely specified function, is an OR/ XNOR expressions with only un-complemented (positive) literals. Each variable in the FPDRM can appear either in un-complemented or complemented form but not both. For an n-variable completely specified Boolean function there are 2^n distinct FPDRMs. There are techniques for converting from POSs to PPDRM or FPDRM [9-11]. In this paper we present an efficient algorithm, which can be used to compute the coefficients of PPDRM or FPDRM directly from the truth table of POS, without the use of mapping techniques [9] and without the use of matrix operation [10]. This algorithm is also used to compute the coefficients of POSs from PPDRM or FPDRM. Time efficiency and computing speed are thus achieved in this technique.

2. Preliminaries

In this section, essential definitions and notations are presented, which are important for the understanding of the paper.

Definition 2.1 An n-variable Boolean function can be expressed as

$$f(x_{n-1},x_{n-2},...,x_0)=\prod_{i=0}^{2^n-1}(\mathrm{d}_i+M_i) \qquad (1)$$

Where '$\prod$' represents logical products (AND), the '+' is OR operation and i is a binary n-tuple $i = [i_0, i_1,\ldots, i_{n-1}]_2$, $[\mathrm{d}_0,\mathrm{d}_1,\ldots, \mathrm{d}_{2^n-1}]$ is the truth vector of the function f, $\mathrm{d}_i \in\{0,1\}$ [12], M_i is a sum term

$$M_i = \sum_{k=n-1}^{0} \overset{\bullet\, i_k}{x_k} = \overset{\bullet\, i_{n-1}}{x_{n-1}} + \overset{\bullet\, i_{n-2}}{x_{n-2}} + \ldots + \overset{\bullet\, i_0}{x_0}$$

and

$$\overset{\bullet\, i_k}{x_k} = \begin{cases} x_k & i_k = 0 \\ \bar{x}_k & i_k = 1 \end{cases}$$

Alternatively, any Boolean function can be represented by a FPDRM expression as:

$$f(x_{n-1},x_{n-2},\ldots,x_0) = \underset{i=0}{\overset{2^n-1}{\otimes}}(\mathrm{c}_i + S_i) \qquad (2)$$

Where '$\otimes$' is XNOR operator, $[\mathrm{c}_{2^n-1}, \mathrm{c}_{2^n-2},\ldots,\mathrm{c}_0]$ is the truth vector of the function f, $\mathrm{c}_i \in\{0,1\}$, $i = [i_0, i_1,\ldots, i_{n-1}]_2$, S_i represents a Sum term as

$$\mathrm{S}_i = \sum_{k=n-1}^{0} \overset{\sim i_k}{x_k} = \overset{\sim i_{n-1}}{x_{n-1}} + \overset{\sim i_{n-2}}{x_{n-2}} + \ldots + \overset{\sim i_0}{x_0}\,, \qquad (3)$$

and

$$\overset{\sim i_k}{x_k} = \begin{cases} 0 & i_k = 0 \\ x_k & i_k = 1 \end{cases}$$

Definition 2.2 Polarity vector $(\mathrm{p}_{n-1},\mathrm{p}_{n-2},\ldots,\mathrm{p}_0)$ for a FPDRM of an n-variable Boolean function is a binary vector with n elements, where $\mathrm{p}_i = 0$ indicates the variable x_i in an un-complemented form (x_i), while $\mathrm{p}_i = 1$ indicates the variable x_i in the complemented form x_i.

Property 1 For an n-variable Boolean function, there are 2^n FPDRM expansions corresponding to 2^n different polarity numbers. Each of such expansions is a canonical representation of a completely specified Boolean function.

Maxterms can be identified by expanding a Kronecker sum of n basis vectors of the form [0 x_i] for '0' polarity and [0 x_i] for '1' polarity .

The FPDRM can be deduced by substituting the coefficient vector **c** in equation (4) for a zero polarity. Thus, for $n = 2$ and $\mathrm{P} = 0$

$[0\ \ x_1]*[0\ \ x_0] = [0{+}0\ \ 0{+}\ x_0\ \ x_1{+}0\ \ x_1{+}\ x_0]$

$$f(x_{n-1},x_{n-2},\ldots,x_0) = \{[0\ \ x_{n-1}]*[0\ \ x_{n-2}]**[0\ \ x_0]\} ✪\ \mathbf{c} \qquad (4)$$

Where '✪' represents matrix multiplication based on OR and XNOR [9-11].

3 Conversion Algorithms from POS to FPDRM and vise versa

To compute ci coefficients from di coefficients, the following principles and derivation are developed. Equation (1) can be represented as

$$f(x_{n-1},x_{n-2},\ldots,x_0) = (\mathrm{d}_0 + x_{n-1}+ x_{n-2} + \ldots\ x_0)\cdot(\mathrm{d}_1 + x_{n-1}+ x_{n-2} + \ldots\ \bar{x}_0)\cdot(\mathrm{d}_2 + x_{n-1}+ x_{n-2} + \ldots\ \bar{x}_1 + x_0)\cdot\ldots\cdot(\mathrm{d}_{2^n-1} + \bar{x}_{n-1}+ \bar{x}_{n-2} + \ldots\ \bar{x}_0) \qquad (5)$$

In equation (1) if all Maxterms are ANDed for each different combination of the inputs the result will be '0' and if all Maxterms are XNORed for each different combination of the inputs variables the result will be also a '0', because for each combination of the inputs one of the Maxtrems will be '0' and the rest will be '1'. Hence equation (1) can be written as in Equation (6) by replacing each AND gate by XNOR gate.

$$f(x_{n-1},x_{n-2},\ldots,x_0) = \underset{i=0}{\overset{2^n-1}{\otimes}}(\mathrm{d}_i + M_i) \qquad (6)$$

$$f(x_{n-1},x_{n-2},\ldots,x_0) = (\mathrm{d}_0 + x_{n-1}+ x_{n-2} + \ldots\ x_0) \otimes (\mathrm{d}_1 + x_{n-1}+ x_{n-2} + \ldots\ \bar{x}_0) \otimes (\mathrm{d}_2 + x_{n-1}+ x_{n-2} + \ldots\ \bar{x}_1 + x_0) \otimes\ldots\otimes (\mathrm{d}_{2^n-1} + \bar{x}_{n-1}+ \bar{x}_{n-2} + \ldots\ \bar{x}_0) \qquad (7)$$

Equation (7) can be described in terms of a coefficient truth vector. The coefficient vector for an n-variable Boolean function can be represented as

$$\mathbf{T} = [\mathrm{d}_0,\mathrm{d}_1,\ldots, \mathrm{d}_{2^n-1}] \qquad (8)$$

The elements of the truth vector (**T**) are placed in the order of decimal equivalent binary coding of the sum terms.

Examining equation (7), half of the sum terms include variable x_i in true form and the second half include variable x_i in complemented form. Therefore, each truth vector (**T**) for any Boolean function in POS form can be separated into two rows for each variable x_i and the result is stored in the separation matrix $\mathbf{T}(x_i)$. The first row of the separation matrix $\mathbf{T}(x_i)$ contains Maxterms with variable x_i in un-complemented form, while the second row of **T**(x_i) contains Maxterms with variable x_i in complemented [13]. The elements in the truth vector and the separation matrix **T**(x_i) are arranged into groups of four bits for convenient. The following example illustrates the separation process.

Example 1
Construct the truth vector **T** for a 4-variable function $f(x_3,x_2,x_1,x_0) = \prod M(0,4,6,7,11,15)$ and use the truth vector **T** to generate the separation matrix for each variable x_i.
The truth vector **T** has 2^n elements. Each Maxterms correspond to '0's in the truth vector **T**. Hence **T** is presented as follows:
T = [0111 0100 1110 1110]
To generate the first matrix T(x_3), the truth vector **T** is separated around variable x_3 int two equal parts. The first part corresponds to un-complemented part, while the second part to the complemented part. This is can be done according to the following formula:

$$\text{Number of Divisions} = (2^n/2^{n-i})$$

Where n is the number of variables and i is the number for variable x_i.
Therefore, n = 4 and i = 3.
Hence,
The un- complemented for x_3 is:

[0111 0100]

And for the complemented is:

[1110 1110]

Therefore,

To generate the first matrix T(x_3), the truth vector **T** is separated around variable x_3 which gives the following result

$$\mathrm{T}(x_3) = \begin{bmatrix} 0111 & 0100 \\ 1110 & 1110 \end{bmatrix} \begin{matrix} x_3 \\ \overline{x}_3 \end{matrix}$$

Similarly the separation matrices for x_2, x_1 and x_0 are as follows:

$$\mathrm{T}(x_2) = \begin{bmatrix} 0111 & 1110 \\ 0100 & 1110 \end{bmatrix} \begin{matrix} x_2 \\ \overline{x}_2 \end{matrix}$$

$$\mathrm{T}(x_1) = \begin{bmatrix} 0101 & 1111 \\ 1100 & 1010 \end{bmatrix} \begin{matrix} x_1 \\ \overline{x}_1 \end{matrix}$$

$$\mathrm{T}(x_0) = \begin{bmatrix} 0100 & 1111 \\ 1110 & 1010 \end{bmatrix} \begin{matrix} x_0 \\ \overline{x}_0 \end{matrix}$$

To replace any complemented variable $\overline{x}_i$ by un-complemented variable x_i in equation (7) the following identity $\overline{x}_i = (0 \otimes x_i)$ is used. The following result is obtained

$$\begin{aligned}(a + \overline{x}_i) \otimes (b + x_i) &= [a + (0 \otimes x_i)] \otimes (b + x_i)\\ &= [(a + 0) \otimes (a + x_i)] \otimes (b + x_i)\\ &= a \otimes [(a + x_i) \otimes (b + x_i)]\end{aligned}$$

However

$$[(a + x_i) \otimes (b + x_i)] = [(a \otimes b) + x_i]$$

This can be verified as follows

$$\overline{[(a + x_i) \otimes (b + x_i)]} = \overline{a}\,\overline{x}_i \oplus \overline{b}\,\overline{x}_i$$

$$= \overline{x}_i(\overline{a} \oplus \overline{b})$$

Where '⊕' is XOR operator.
Complementing the last expression, the following is obtained

$$\overline{\overline{x}_i\ (\overline{a} \oplus \overline{b})} = [(a \otimes b) + x_i]$$

Therefore,

$$[(a + x_i) \otimes (b + x_i)] = [(a \otimes b) + x_i]$$

Hence

$$(a + \overline{x}_i) \otimes (b + x_i) = a \otimes [(a \otimes b) + x_i] \qquad (9)$$

Examining equation (9), the coefficients of the un-complemented part of variable x_i takes a new form. The new coefficient is (a XNOR b), while the coefficient for the complemented part will remain the same. Similarly, to convert un-complemented form to complemented form the following principal is applied.
Each un-complemented variable x_i is replaced by $0 \otimes \overline{x}_i$.

$$\begin{aligned}(a + \overline{x}_i) \otimes (b + x_i) = &\ (a + \overline{x}_i) \otimes [(b + (0 \otimes \overline{x}_i)]\\ &= (a + \overline{x}_i) \otimes [(b + 0) \otimes (b + \overline{x}_i)]\\ &= [(a + \overline{x}_i) \otimes (b + \overline{x}_i)] \otimes b\end{aligned}$$

By taking the complement of the following expression

$$\overline{[(a + \overline{x}_i) \otimes (b + \overline{x}_i)]} = \overline{a}x_i \oplus \overline{b}x_i = x_i(\overline{a} \oplus \overline{b})$$

Taking the complement for the last expression will give the following result

$$\overline{x_i\ (\overline{a} \oplus \overline{b})} = \overline{x}_i + (a \otimes b)$$

Hence

$$[(a + \overline{x}_i) \otimes (b + \overline{x}_i)] = \overline{x}_i + (a \otimes b)$$

Therefore,

$$(a + \bar{x}_i) \otimes (b + x_i) = b \otimes [(a \otimes b) + \bar{x}_i] \qquad (10)$$

Inspecting equation (10), the coefficient of the true form stays as it is while the coefficient of the complemented form is replaced by (a XNOR b).

Algorithm 1
A computer algorithm has been developed based on the previous theory as shown in the following steps.

Algorithm A: Converting from POS to FPDRM
Step 1: Store the coefficients of the POS in the truth vector **T**.
Step 2: Construct **T**(x_i) matrix from **T** vector for each variable x_i. The first row of **T**(x_i) matrix contains the coefficients of the Maxterms for variable x_i in un-complemented form. While the second row of **T**(x_i) contains the coefficients of the Maxterms with variable $\bar{x}_i$ in the complemented form.
Step 3: The elements in the first and second rows of **T**(x_i) matrix are group together using XNOR operation and the result is stored in vector **N**.
Step 4: If the required polarity for x_i variable is '0' then replace the contents of each true variable x_i in the truth vector **T** by the contents of vector **N**.
Step 5: If the required polarity for x_i variable is '1' then replace the contents of each complemented part of the $\bar{x}_i$ variable in the truth vector **T** by the contents of the un-complemented part of the x_i variable and store the result **N** in place of un-complemented part of x_i variable in **T**.
Step 6: Repeat the previous steps for the rest of the variables by using the new truth vector from step '5' or '6' depending on the polarity.
Step 7: The zero elements stored in the last **T** vector are the coefficients for that particular polarity of the FPDRM.

Algorithm B: Converting from FPDRM to POS
To find the POS's coefficients from the FPDRM's coefficients, step '5' is changed to the following step:
If the required polarity for x_i variable is '1' then replace the contents of each un-complemented part of the x_i variable in the truth vector **T** by the contents of the complemented part of the x_i variable and store the in **T**.
The following examples will illustrate Algorithm 1.a and Algorithm 1.b.

Example 2
Convert a 4-variable function $f\ (x_3, x_2, x_1, x_0) = \prod M(0,4,7,11,15)$ from the POS to the fixed polarity DRM by using polarity **p** = 7 = (0111).
Store the coefficients of Maxterms in the truth vector **T**.
T = [0111 0110 1110 1110]
Separate **T** vector around variable x_3 to obtain **T**(x_3) matrix and XNOR each element in the first row with the elements in the second row.

$$\mathrm{T}(x_3) = \underline{\begin{bmatrix} 0111\ 0110 \\ 1110\ 1110 \end{bmatrix}} \begin{matrix} x_3 \\ \bar{x}_3 \end{matrix} \ \text{XNOR}$$
$$0110\ 0111$$

Since the polarity is '0' for variable x_3, replace the un-complemented part of x_3 variable in **T** by the **N** vector results. The new **T** vector is [0110 0111 1110 1110].
Separate the new vector **T** around variable x_2 to obtain **T**(x_2) matrix as follows:

$$\mathrm{T}(x_2) = \underline{\begin{bmatrix} 0110 & 1110 \\ 0111 & 1110 \end{bmatrix}} \begin{matrix} x_2 \\ \bar{x}_2 \end{matrix} \ (x_2 \text{ XNOR } \bar{x}_2)$$
$$\mathrm{N} = [1110 \quad 1111]$$

Since the polarity is '1' for variable x_2 apply step '5', the new truth vector is
T = [1110 0110 1111 1110].
Similarly for variable x_1 and x_0

$$\mathrm{T}(x_1) = \underline{\begin{bmatrix} 1101 & 1111 \\ 1010 & 1110 \end{bmatrix}} \begin{matrix} x_1 \\ \bar{x}_1 \end{matrix} \ (x_1 \text{ XNOR } \bar{x}_1)$$
$$\mathrm{N} = [1000 \quad 1110]$$

T = [1011 0001 1111 1011]

$$\mathrm{T}(x_0) = \underline{\begin{bmatrix} 1100 & 1111 \\ 0101 & 1101 \end{bmatrix}} \begin{matrix} x_0 \\ \bar{x}_0 \end{matrix} \ (x_0 \text{ XNOR } \bar{x}_0)$$
$$\mathrm{N} = [0110 \quad 1101]$$

The final **T** vector is
T = [0111 1000 1111 0111]={0,5,6,7,12}
The sum terms in this canonical can be generated by using the basis vector

$$[0\ x_3]{+}{+}[0\ \bar{x}_2]{+}{+}[0\ \bar{x}_1]{+}{+}[0\ \bar{x}_0] =$$
$$[0\ \bar{x}_o\ \ \bar{x}_1\ \ (\bar{x}_1+\bar{x}_0)\ \bar{x}_2\ \ (\bar{x}_2+\bar{x}_o)$$
$$(\bar{x}_2+\bar{x}_1)\ \ (\bar{x}_2+\bar{x}_1+\bar{x}_o)\ (x_3+\bar{x}_0)\ \ (x_3+\bar{x}_1)$$
$$(x_3+\bar{x}_1+\bar{x}_0)\ \ (x_3+\bar{x}_2)\ \ (x_3+\bar{x}_2+\bar{x}_0)$$
$$(x_3+\bar{x}_2+\bar{x}_1)\ \ (x_3+\bar{x}_2+\bar{x}_1+\bar{x}_0)]$$

The FPDRM can be generated using by substituting the coefficient vector **c** in the following general equation.

$f(x_{n-1},x_{n-2},\ldots,x_0) = \{[0\ \ x_{n-1}] ++ [0\ \ x_{n-2}] ++ \ldots ++ [0\ \ x_0]\ \} ^{o}\ \mathbf{c}$ (11)

Hence

$$f\ (x_3, x_2, x_1, x_0) = 0 \otimes (\bar{x}_2 + \bar{x}_0) \otimes (\bar{x}_2 + \bar{x}_1)$$
$$\otimes (\bar{x}_2 + \bar{x}_1 + \bar{x}_0) \otimes (x_3 + \bar{x}_2 + \bar{x}_0)$$

Example 3

Convert a 4-variable function $f\ (x_3,x_2,x_1,x_0)\ = \otimes$ (0,5,6,7,12) from FPDRM form to POS form by using polarity **p** = 7 = (0111).

Store the truth vector in **T** matrix.

T = [0111 1000 1111 0111]

Separate **T** vector around variable x_3 to obtain T(x_3) matrix.

$$\mathrm{T}(x_3) = \begin{array}{c|c} \begin{bmatrix} 0111 & 1000 \\ 1111 & 0111 \end{bmatrix} & \begin{array}{c} x_3 \\ \bar{x}_3 \end{array} \end{array} \quad (x_3 \text{ XNOR } \bar{x}_3)$$

$$\mathrm{N} = [0111\quad 0000]$$

Since polarity is '0' for variable x_3, replace the un-complemented part of variable x_3 in **T** by the **N** vector results. The new truth vector is

T = [0111 0000 1111 0111].

Separate the new vector **T** around variable x_2 to obtain T(x_2) matrix as follows

$$\mathrm{T}(x_2) = \begin{array}{c|c} \begin{bmatrix} 0111 & 1111 \\ 0000 & 0111 \end{bmatrix} & \begin{array}{c} x_2 \\ \bar{x}_2 \end{array} \end{array} \quad (x_2 \text{ XNOR } \bar{x}_2)$$

$$\mathrm{N} = [1000\quad 0111]$$

Since polarity is '1' for variable x_2 apply Algorithm B, the new truth vector is

T= [0000 1000 0111 0111]

Similarly for variable x_1 and x_0

$$\mathrm{T}(x_1) = \begin{array}{c|c} \begin{bmatrix} 0010 & 0101 \\ 0000 & 1111 \end{bmatrix} & \begin{array}{c} x_1 \\ \bar{x}_1 \end{array} \end{array} \quad (x_1 \text{ XNOR } \bar{x}_1)$$

$$\mathrm{N} = [1101\quad 0101]$$

T= [0011 0001 1101 1101]

$$\mathrm{T}(x_0) = \begin{array}{c|c} \begin{bmatrix} 0100 & 1010 \\ 0101 & 1111 \end{bmatrix} & \begin{array}{c} x_0 \\ \bar{x}_0 \end{array} \end{array} \quad (x_0 \text{ XNOR } \bar{x}_0)$$

$$\mathrm{N} = [1110\quad 1010]$$

T= [0111 0110 1110 1110]

Therefore, the POS's coefficients are (0,4,7,11,15).

$f\,(x_3,x_2,x_1,x_0) = \prod \mathrm{M}(0,4,7,11,15)$.

4. Optimization of the Fixed Polarity DRM forms

In the optimization of the FPDRM functions with different polarities are usually calculated directly from POS expressions [10, 11]. A new algorithm is presented in this paper to find the optimal polarity directly from the truth vector of the zero polarity. This technique is aimed at large number of variables, where time is very crucial. It usually requires a long time to convert from POS to DRM for each polarity and then search for the best polarity among the 2^n polarities. The new algorithm introduced in this section will achieve maximum efficiency in respect of time for large number of variables and does not require a large memory. The time required to find a 'good' polarity, is almost equal to the time required for converting a single polarity as giving in algorithm B. This algorithm doesn't search each polarity to convert from POS to FPDRM and it doesn't use matrix technique to convert from POS to DRM for each polarity. Thus the algorithm is fast with respect to time and efficient in terms of memory storage.

Algorithm 4.1

Step 1: Algorithm A is used to obtain zero polarity, the coefficients are stored in vector **T**. Let $\mathbf{P}_{\min}$ equals the polarity number zero for the zero polarity which is zero.

Step 2: Count the number of zero terms in vector **T** and denote it by (TNZ).

Step 3: Construct T(x_i) matrix from vector **T** for each variable x_i. The first row of T(x_i) matrix contains the coefficients of the Maxterms for variable x_i in un-complemented form. The second row of T(x_i) contains the coefficients of the Maxterms with variable $\bar{x}_i$ in the complemented form.

Step 4: The elements in the first and second rows of matrix T(x_i) are grouped together using XNOR operation and the result is stored in vector **N**.

Step 5: Count the number of zeros of the un-complemented part from T(x_i) and **N** . Add the two numbers together and denote it by NZ(x_i).

Step 6: Repeat steps 4 and 5 for all the variables that have not been converted to polarity 1.

Step 7: To determine the variable (x_i) that has to be converted from '0' polarity to '1' polarity. Select the variable with the least number of zeros NZ(x_i) from **Step 6**. This should be less than or equal to the total number of zeros from step 2, TNZ.

Step 8: Replace the contents of complemented part of T(x_i) by the contents of vector **N**. This will generate a new **T** vector. $\mathbf{P}_{\min}$ is set to the new polarity number.

Step 9: Use the new **T** vector from step 8 and repeat the same procedure from step 2 for the variables that have not been converted.

Step 10: If the total number of zeros NZ(x_i) for each variable x_i from step 6 is greater than TNZ from step 2, then stop and there will be no more variables to convert from '0' polarity to '1' polarity.

Step 11: The zero elements stored in the last **T** vector are the coefficients for that particular polarity of the FPDRM.

Example 4

Find an optimal polarity for a 5-variable function

$$f(x_4, x_3, x_2, x_1, x_0) = \prod (1,3,4,5,7,10,11,12).$$

Step 1: use Algorithm (A) to convert from POS to zero polarity DRM and set

$\mathbf{P}_{\min} = 0$.

The result is as follows: **T** = [1010 1100 0010 0110 1111 1111 1111 1111]

Hence the coefficients for PPDRM with polarity **p** = 0 are: **c** = {1, 3, 6, 7, 8, 9, 11, 12, 15}

Step 2: Count the number of zero terms in **T** vector and set TNZ = 9.

Step 3: Separate **T** vector around variable x_4 into un-complemented and complemented parts, and store the result into vector **N** as follows:

$$\mathrm{T}(x_4) = \begin{bmatrix} 1010 & 1100 & 0010 & 0110 \\ 1111 & 1111 & 1111 & 1111 \end{bmatrix} \begin{matrix} x_4 \\ \bar{x}_4 \end{matrix} (x_4 \text{ XNOR } \bar{x}_4)$$
$$\mathrm{N} = [1010 \quad 1100 \quad 0010 \quad 0110]$$

Step 4: Count the number of zeros of the un-complemented part from T(x_4) and **N** vector and add the two numbers together: NZ (x_4) = 18

By repeating steps 3 to 6, the following results are obtained for the following variables x_3, x_2, x_1 and x_0

$$\mathrm{T}(x_3) = \begin{bmatrix} 1010 & 1100 & 1111 & 1111 \\ 0010 & 0110 & 1111 & 1111 \end{bmatrix} \begin{matrix} x_3 \\ \bar{x}_3 \end{matrix} (x_3 \text{ XNOR } \bar{x}_3)$$
$$\mathrm{N} = [0111 \quad 0101 \quad 1111 \quad 1111]$$

NZ (x_3) = 7

$$\mathrm{T}(x_2) = \begin{bmatrix} 1010 & 0010 & 1111 & 1111 \\ 1100 & 0110 & 1111 & 1111 \end{bmatrix} \begin{matrix} x_2 \\ \bar{x}_2 \end{matrix} (x_2 \text{ XNOR } \bar{x}_2)$$
$$\mathrm{N} = [1001 \quad 1011 \quad 1111 \quad 1111]$$

NZ (x_2) = 8

$$\mathrm{T}(x_1) = \begin{bmatrix} 1011 & 0001 & 1111 & 1111 \\ 1000 & 1010 & 1111 & 1111 \end{bmatrix} \begin{matrix} x_1 \\ \bar{x}_1 \end{matrix} (x_1 \text{ XNOR } \bar{x}_1)$$
$$\mathrm{N} = [1100 \quad 0100 \quad 1111 \quad 1111]$$

NZ (x_1) = 9

$$\mathrm{T}(x_0) = \begin{bmatrix} 1110 & 0101 & 1111 & 1111 \\ 0010 & 0010 & 1111 & 1111 \end{bmatrix} \begin{matrix} x_0 \\ \bar{x}_0 \end{matrix} (x_0 \text{ XNOR } \bar{x}_0)$$
$$\mathrm{N} = [0011 \quad 1000 \quad 1111 \quad 1111]$$

NZ (x_0) = 8

Since NZ (x_3) has the minimum number of zeros, replace the contents of complemented part of **T** matrix by the result of XNOR operation hence the new T vector is

T = [1010 1100 0111 0101 1111 1111 1111 1111].

The total number of zeros for the new vector TNZ = 7, and the polarity number for this vector is $\mathbf{P}_{\min}$ = {01000}= 8.

Similarly as in the previous part, separate vector **T** around each variable but not x_3 because it has been changed to $\bar{x}_3$.

$$\mathrm{T}(x_4) = \begin{bmatrix} 1010 & 1100 & 0111 & 0101 \\ 1111 & 1111 & 1111 & 1111 \end{bmatrix} \begin{matrix} x_4 \\ \bar{x}_4 \end{matrix} (x_4 \text{ XNOR } \bar{x}_4)$$
$$\mathrm{N} = [1010 \quad 1100 \quad 0111 \quad 0101]$$

NZ (x_4) = 14

$$\mathrm{T}(x_2) = \begin{bmatrix} 1010 & 0111 & 1111 & 1111 \\ 1100 & 0101 & 1111 & 1111 \end{bmatrix} \begin{matrix} x_2 \\ \bar{x}_2 \end{matrix} (x_2 \text{ XNOR } \bar{x}_2)$$
$$\mathrm{N} = [1001 \quad 1101 \quad 1111 \quad 1111]$$

NZ (x_2) = 6

$$\mathrm{T}(x_1) = \begin{bmatrix} 1011 & 0101 & 1111 & 1111 \\ 1000 & 1101 & 1111 & 1111 \end{bmatrix} \begin{matrix} x_1 \\ \bar{x}_1 \end{matrix} (x_1 \text{ XNOR } \bar{x}_1)$$
$$\mathrm{N} = [1100 \quad 0111 \quad 1111 \quad 1111]$$

NZ (x_1) = 6

$$\mathrm{T}(x_0) = \begin{bmatrix} 1110 & 0100 & 1111 & 1111 \\ 0010 & 1111 & 1111 & 1111 \end{bmatrix} \begin{matrix} x_0 \\ \bar{x}_0 \end{matrix} (x_0 \text{ XNOR } \bar{x}_0)$$
$$\mathrm{N} = [0011 \quad 0100 \quad 1111 \quad 1111]$$

NZ (x_0) = 9

Since the total number of zeros for variable x_2 is less than TNZ from the last operation, therefore x_2 will be

converted to x_2. Hence the new **T** vector is: **T** = [1010 1001 0111 1101 1111 1111 1111 1111].
The total number of zeros for the new vector TNZ = 6 and P_{min} = {01100} = 12.
Repeat the same procedure for x_4, x_1 and x_0.

$$\mathrm{T}(x_4)=\begin{bmatrix}1010 & 1001 & 0111 & 1101\\ 1111 & 1111 & 1111 & 1111\end{bmatrix}\begin{matrix}x_4\\ \bar{x}_4\end{matrix}\,(x_4 \text{ XNOR } \bar{x}_4)$$

$$\mathrm{N}=\begin{bmatrix}1010 & 1001 & 0111 & 1101\end{bmatrix}$$

$$\mathrm{NZ}\ (x_4) = 12$$

$$\mathrm{T}(x_1)=\begin{bmatrix}1010 & 0111 & 1111 & 1111\\ 1001 & 1101 & 1111 & 1111\end{bmatrix}\begin{matrix}x_1\\ \bar{x}_1\end{matrix}\,(x_1 \text{ XNOR } \bar{x}_1)$$

$$\mathrm{N}=\begin{bmatrix}1100 & 0101 & 1111 & 1111\end{bmatrix}$$

$$\mathrm{NZ}\ (x_1) = 7$$

$$\mathrm{T}(x_0)=\begin{bmatrix}1110 & 0110 & 1111 & 1111\\ 0001 & 1111 & 1111 & 1111\end{bmatrix}\begin{matrix}x_0\\ \bar{x}_0\end{matrix}\,(x_0 \text{ XNOR } \bar{x}_0)$$

$$\mathrm{N}=\begin{bmatrix}0000 & 0110 & 1111 & 1111\end{bmatrix}$$

$$\mathrm{NZ}\ (x_0) = 9$$

Since the total number of zeros from each operation is greater than TNZ from the last operation, there are no more variables to convert to the complemented form and the process is terminated at this point.
Therefore, the final **c** vector is given as follows:
c = [1010 1001 0111 1101 1111 1111 1111 1111]
The FPDRM terms that are needed for this form are {1,3,5,6,8,14}, and the best polarity is **P** = 12.
A program has been developed based on the previous theory and sparse technique. A matrix is called sparse if most of its elements are non-zero [14, 15]. Considerable saving in memory and computation time can be achieved by using sparse formats that store only the zeros in this case. Since most of the elements in the truth vector **T** are non-zeros where the normal size of **T** is 2^n, a sparse format will be a suitable solution to store the zero elements to avoid wasting memory. The following example illustrates this method.

Example 5
Let vector **A** = [0,4,7] and vector **B** = [2,4,6] then **A** XNOR **B** is given as follows:

A = [0 y y y 4 y y 7]
B = [y y 2 y 4 y 6 y]
N = [0 y 2 y y y 6 y]

Where 'y' presents a non-zero element '1' in the truth vector.

5. Experimental Results

In this section, experimental results are presented for the proposed algorithms. The proposed algorithms are implemented in C language and the programs are compiled using Borland C++ compiler. It is tested on a personal computer with Pentium 4 processor of 2.4 GHz CPU and 512 MB of RAM under Window operating system. The algorithms where applied to several MCNC benchmarks. Table (1) shows the results obtained from converting PPDRM coefficients into POSs coefficients. Table (2) shows the results obtained from converting POS coefficients into PPDRM coefficients. Where name denotes the name of circuit, *n* denotes the number of variables, init terms denotes the number of terms in POS form, PPDRM terms denotes the number of terms in PPDRM form and the CPU time is in seconds. For most of the circuits with *n* less 14 the CPU time is less than 0.6 seconds.

Table 1:
Conversion table from PPDRM to POS

Name	*n*	PPDRM Terms	POS Terms	Time (s)
Clip	9	92	480	0.001
Con1	7	9	88	0.001
Ex1010	10	480	142	0.010
Rd84	8	37	136	0.001
Table3	14	2528	1859	1.29
Table5	17	3359	28552	4.927

Table 2:
Conversion table from POS to PPDRM

Name	*n*	Init. POS terms	PPDRM terms	CPU Time
Con1	7	88	9	0.001
Rd84	8	136	37	0.001
Clip	9	480	92	0.001
Ex1010	10	142	480	0.001
F12†	12	1984	365	0.01
F13†	13	4152	127	0.04
F14†	14	16172	65	0.521
F15†	15	5792	3100	3.695
spla	16	5348	517	1.843
Table5	17	28552	3359	162.85

References

[1] Reed, I.S., 1954, A claa of multiple-error-correction codes and their decoding schem. IRE trans. Inform. Theory, IT-4, 38-49.

[2] Muller, D.E. Sept., 1954, pplication of Boolean algebra to switching circuit design for error detection. IEEE trans. Comput, EC-3, 6-12.

[3] Jeong, B.k., Sung, J.H., and Jong, K., 1997, New circuit for XOR and XNOR functions. International Journal of Electronics, **82**, 131-143.

[4] McCluskey, E., 1986, Logic Design Principles with Emphasis on Testable Semicustom Circuits. (Prentice Hall).

[5] Wang, J.M., Fang, S. C., and Feng, W.S., 1994, New efficient design for XOR and XNOR functions on the transistor level. IEEE Journal of solid-state Circuits, **29**, 780-786.

[6] Sasao, T., 1997 , Easily Testable Realizations for Generalized Reed-Muller Expressions. IEEE Transaction on computers, **46**, 709-716.

[7] Robinson, J.P. and Yeh, C.L. Aug., 1982, A method for modulo-2 minimization. IEEE Trans. Comput, C-31, 800-801

[8] Sassao, T and Besslich, P. Feb., 1990, On the complexity of Mod-2 sum PLA's. IEEE Trans. Comput. C-39, 262-266.

[9] Cheng, J., Chen, X., Faraj, K.M., and Almaini, A.E.A., 2003, Expansion of logical functions in the OR-coincidence system and the transform between it and maxterm. IEE Proc.-comput. Digit. Tech, **150**, 397-402.

[10] Green, D.H., 1994, Dual forms of Reed-Muller expansions. IEE Proc.-Comput. Digit. Tech, **141**, 184-192.

[11] Faraj, K., MacCallum, M., and Almaini, A.E.A., 2004, Fast computation of Conjunctive Canonical Reed-Muller functions. PREP Proceeding, University of Hertfordshire, 144.

[12] Almaini, A.E.A., 1994, Electronic logic systems, 3rded, (Prentice Hall).

[13] Habbib, M.K., 2002, A new approach to generate fixed-polarity Reed-Muller expansions for completely and incompletely specified functions. International Journal of Electronics, **89**, 845-876.

[14] Toledo, S., 'Improving the memory-system performance of sparse-matrix vector multiplication,' IBM J. RES. Develop, Vol. 41, No. 6, pp. 711-725, November 1997.

[15] Duff, I. S., Heroux, M. A., and Pozo, R., 'The Sparse BLAS, Technical Report, CERFACS tr/pa/01/24,' September 2001.