Exact minimization of Dual Reed-Muller expansions

K. Faraj               A.E.A. Almaini
Computer Science                School of Engineering
Wajdi Institute of Technology                    Napier University
Jerusalem, Mount of Olives. P.O.Box 19014           Edinburgh
Palestine              Scotland

http://www.wit.edu.ps  www.napier.ac.uk

Abstract:- This paper presents two algorithms; the first one is used to convert between Product of Sums (POS) and Positive Polarity Dual Reed-Muller (PPDRM) forms. The second algorithm is used to compute all the coefficients of the Fixed Polarity Dual Reed-Muller (FPDRM) with polarity p from any polarity q. This technique is used to find the best polarity of FPDRM among the $2^n$ fixed polarities. The algorithm is based on the dual property and the Gray code strategy. Therefore, there is no need to start from POS form to find FPDRM coefficients for all the polarities. The proposed methods are efficient in terms of memory size and CPU time as shown in the experimental results.

Key-Words: - POS, FPDRM, Reed-Muller

1 Introduction

The realization of Boolean functions using OR/XNOR gates is very attractive. In many applications, Boolean functions based on OR/XNOR operations can have fewer gates and fewer connections, resulting in less power consumption and less area of VLSI circuits. Recent progress in circuit technology makes the use of OR/XNOR gates feasible. XNOR, XOR circuits have applications in arithmetic telecommunication error and detection,. In certain applications, circuits which use XNOR/XOR gates, have advantages over circuits which use other kind of gates. Some of these advantages are improved testability and reduced number of transistors [1-4]. FPDRM is one of the canonical OR/XNOR expressions [5, 6] and they are a generalization of PPDRM expressions. 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 POS to PPDRM or FPDRM [5-7].

Many algorithms have been published for converting a Boolean function from sum of product (SOP) to a Fixed Polarity Reed-Muller (FPRM) format [8-12]. But little has been done by converting POS to FPDRM and finding the best polarity. An efficient technique is presented in this paper to convert from POS to FPDRM. Using simple equations, this can be used directly to compute FPDRM coefficients without using transformation matrices.

In the optimization of FPDM functions, polynomials with different polarities are usually calculated directly from POS expressions [5, 6]. A new algorithm is presented in this paper to generate all the polarity sets from any polarity set q for a single output Boolean function. Time efficiency and computing speed are achieved in this technique because the information in finding FPDRM expansion of one polarity is utilized by others. Two-fixed polarities can be derived from each other without the need to go back to the original Boolean function, if the two polarities are dual.

This algorithm is fast, it requires a short CPU time to find the optimal polarity and it doesn't require a large memory.

2 Definitions and representation of fixed polarity Dual Reed-Muller

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

$$f(x_{n-1}, x_{n-2}, \ldots, x_0) = \bigwedge_{i=0}^{2^n-1} (d_i + M_i)$$

Where $\bigwedge$ represents logical products (AND), the '+' is OR operation and $i$ is a binary $n$-tuple $i = (i_0, i_1, \ldots, i_{n-1})_2$, $[d_0, d_1, \ldots, d_{2^n-1}]$ is the truth vector of the function $f$, $d_i \in \{0,1\}$ [13], $M_i$ is a sum term

$$M_i = \bigoplus_{k=0}^{n-1} x_k = x_{n-1} + x_{n-2} + \ldots + x_0$$
and \( \cdot \mathbf{i} = \begin{bmatrix} x_k \\ i_k \end{bmatrix} \) with \( i_k = 0 \) or \( i_k = 1 \).

**Definition 2.2** Alternatively, any \( n \)-variable function can be expressed by the FPDRM expression as:

\[
f(x_{n-1}, x_{n-2}, \ldots, x_0) = \bigcirc \prod_{i=0}^{2^n-1}(c_i \oplus S_i)
\]

(2)

Where ‘\( \bigcirc \)’ is XNOR operator, \( [c_{2^n-1}, c_{2^n-2}, \ldots, c_0] \) is the truth vector of the function \( f \), \( c_i \in \{0, 1\}, i = (i_{n-1}, i_{n-2}, \ldots, i_0) \); \( S_i \) represents a Sum term.

\[
S_i = \sum_{k=0}^{n-1} x_k = x_{n-1} + x_{n-2} + \ldots + x_0,
\]

(3)

and \( x_k = \begin{cases} 0 & i_k = 0 \\ 1 & i_k = 1 \end{cases} \)

To convert from POS to FPDRM forms, requires the derivation of the coefficient vector \( \mathbf{c} \) from the truth vector \( \mathbf{d} \) using the transformation matrix as given in equation (4), provided the order of the components in \( \mathbf{d} \) are reversed. Where ‘\( \bigcirc \)’ represents matrix multiplication based on OR and XNOR [5-7]. The transformation matrix (TM) for a ‘0’ polarity is defined as:

\[
\mathbf{TM(0) = } \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}
\]

Using equation (4) the truth vector \( \mathbf{c} \) is obtained as follows:

\[
\mathbf{c} = \mathbf{d} \bigcirc \mathbf{TM(0)}, \mathbf{TM(1) = } \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}
\]

(5,6)

In general \( \mathbf{TM}_n \) is given as follows:

\[
\mathbf{TM}_n = \mathbf{TM}_1 \ast \mathbf{TM}_1 \ast \ldots \ast \mathbf{TM}_1 \text{ } n \text{ times.}
\]

(7)

Where ‘\( \ast \)’ is the Kronecker sum. Furthermore \( \mathbf{TM}_n^{-1} = \mathbf{TM}_n \) is a self-inverse matrix. Maxterms and sum terms are the counterparts for minterms and product terms in SOP form. In a FPDRM, each variable can appear in either un-complemented or complemented form. Therefore, for an \( n \)-variable function there are \( 2^n \) distinct FPDRM. Maxterms can be identified by expanding a Kronecker sum of \( n \) basis vectors of the form \( \{0 \} \) for ‘0’ polarity and \( \{0, \bar{x}_i\} \) for ‘1’ polarity. Thus, for \( n = 2 \) and \( P = 0 \):

\[
[0 \ 0] \ast [0 \ 0] = [0 \ 0,0 \ 0 \ 0 \ 0 \ 0 \ 0]
\]

The FPDRM can be deduced by substituting the coefficient vector \( \mathbf{c} \) in equation (8) for a zero polarity.

\[
f(x_{n-1}, x_{n-2}, \ldots, x_0) = [0 \ x_0] \ast [0 \ x_0] \ast \ldots \ast [0 \ x_0] \bigcirc \mathbf{c}
\]

(8)

Thus for \( n = 2 \) and \( p = 0 = 00 \) where \( p \) is the polarity number;

The transformation matrix for \( n = 2 \) is calculated as follow:

\[
\mathbf{TM(2) =} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}
\]

Using equation (4) the truth vector \( \mathbf{c} \) is obtained as follows:

\[
\mathbf{c} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}
\]

Using the following identities for XNOR

\[
0 \bigcirc 0 = 1, 0 \bigcirc 1 = 0, 1 \bigcirc 0 = 0, \text{ and } 1 \bigcirc 1 = 1.
\]

Hence:

\[
c_3 = (0 \bigcirc d_3) \bigcirc (1 \bigcirc d_2) \bigcirc (1 \bigcirc d_1) \bigcirc (1 \bigcirc d_0) = d_3 \bigcirc 1 \bigcirc 1 = d_3 \bigcirc 1
\]

\[
c_2 = (0 \bigcirc d_3) \bigcirc (0 \bigcirc d_2) \bigcirc (1 \bigcirc d_1) \bigcirc (1 \bigcirc d_0) = d_3 \bigcirc d_2 \bigcirc 1 \bigcirc d_0 = d_2 \bigcirc d_1 \bigcirc 1
\]

\[
c_1 = (0 \bigcirc d_3) \bigcirc (1 \bigcirc d_2) \bigcirc (0 \bigcirc d_1) \bigcirc (1 \bigcirc d_0) = d_3 \bigcirc d_2 \bigcirc d_1 \bigcirc 1
\]

\[
c_0 = (0 \bigcirc d_3) \bigcirc (0 \bigcirc d_2) \bigcirc (0 \bigcirc d_1) \bigcirc (0 \bigcirc d_0) = d_3 \bigcirc d_2 \bigcirc d_1 \bigcirc d_0
\]

Finally the FPDRM function is calculated using equation (8) as follow:

\[
f(x_1, x_0) = [0 \ x_0] \bigcirc \mathbf{TM(2) \ ast \ c}
\]

\[
= [0 \ x_1 \ x_0 \ x_1 + x_0]
\]

**Example 1**: For a three variable Boolean function \( f(x_2, x_1, x_0) = \bigcirc \{7, 4, 3, 1\} \), where the truth vector \( \mathbf{c} = [0, 1, 1, 0, 1, 0, 1, 1] \) for polarity \( p = 1 = [0, 0, 1] \) then \( f \) in FPDRM is

\[
f(x_2, x_1, x_0) = [0 \ x_2] \ast [0 \ x_1] \ast [0 \ x_0] \bigcirc \mathbf{c}
\]

Where

\[
[0 \ x_2] \ast [0 \ x_1] \ast [0 \ x_0] =
\begin{bmatrix}
0 \ x_0 \\
0 \ x_1 \\
0 \ x_2
\end{bmatrix}
\]

\[
= [0 \ x_0 \\
0 \ x_1 \\
0 \ x_2 + x_1 + x_0]
\]

Hence

\[
f(x_2, x_1, x_0) = 0 \bigcirc (x_1 + x_0 \bigcirc (x_2 \bigcirc (x_2 + x_1)
\]

**Example 2**: Dual polarity for polarity \( p = (00) \) are \( (01) \) and \( (10) \).
3 Conversions from POS to FPDRM

To derive FPDRM coefficients from POS coefficients using equation (4) would be very costly in terms of memory and CPU time. Equation (4) requires the construction and storing of the matrix TM, which has a size of $2^n$ by $2^n$ for n-variables. This is overcome by using the technique introduced in [7]. The following equations are used in [7] to convert between POS and FPDRM forms

$$c_i = c_{i_{n-1} i_{n-2} \ldots i_0} = \bigodot \prod_{l} d_{i_{n-1} i_{n-2} \ldots i_0} \quad (9)$$

$$l_k = \begin{cases} y & i_k = 0 \ 1 & i_k = 1 \end{cases} \quad (10)$$

$$d_j = d_{i_{n-1} i_{n-2} \ldots i_0} = \bigodot \prod_{l} c_{i_{n-1} i_{n-2} \ldots i_0} \quad (11)$$

Where $l = (l_{n-1} l_{n-2} \ldots l_0)$, ‘$y$’ is the notation for both 1 and 0 and $l_k$ in equation (11) is the same as in equation (10).

To facilitate the use equations (9-11) for large Boolean functions the bitwise relationship between the subscripts $d$ and $c$ in equation (10) is represented by using truth Table (1). The following equation is obtained using Karnaugh map of POS for the standard function $\psi_j = (\psi_{n-1} \psi_{n-2} \ldots \psi_0)$. A loop of 0-cells in a Karnaugh map is related to a sum term.

$$\psi_{j} = i_j \bar{l_j} \quad (12)$$

Where ‘$\cdot$’ is the normal AND operator and $\bar{l}$ represents the complement of the off-sets. When $\psi_{j}$ is equal to ‘0’, that is all of its binary bits are ‘0’, then equations (9) and (11) are satisfied.

Table 1: Map of the standard function $\psi_{j}$

<table>
<thead>
<tr>
<th>$i_j$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Observation:

If the number of off-set coefficients $d_i$ in equation (9) for the corresponding $S_i$ coefficient is odd (odd number of zeros), then coefficient $c_i$ should be included for that DRM expansion. Otherwise it should not be included because $0 \odot 0 = 1$. Zero coefficients should be included for DRM expansion only.

Algorithm 1

This algorithm converts POS to PPDRM form.

Step 1: Store the complement of each term in the off-set.

Step 2: Use AND operation to AND each term from step ‘1’ with the decimal value of each coefficient $c_i$.

Step 3: Count the number of zeros of each term for each coefficient $c_i$ from step ‘2’, if the total number is odd then that coefficient $c_i$ should be included, otherwise it shouldn’t be included.

Step 4: Repeat steps two and three for the rest of the coefficients.

Example 3: For a three variable Boolean function $f(x_2x_1x_0) = \Pi M(0,4,6,7)$ find the expression of PPDRM.

Applying Algorithm 1

Step 1: Store the complement of each term in the off-set; this gives the following results.

$\bar{0}, \bar{4}, \bar{6}, \bar{7} = 7,3,1,0$

Step 2: The first coefficient of PPDRM ($c_0$) is calculated using equations (9) and (12) as follows:

$c_0 = \bar{0} \odot \bar{0} \odot \bar{4} \odot \bar{0} \odot \bar{6} \odot \bar{7} = \bar{0} \odot \bar{0} \odot \bar{3} \odot \bar{0} \odot \bar{0} \odot \bar{0} \odot \bar{0}$

Step 3: Count the number of zeros in the last expression, which is four. Since the number of zeros is even then this coefficient ($c_0$) should not be included for PPDRM. Hence $c_0 = 1$

Similarly for the rest of the coefficients are obtained as follows:

$c_1 = \bar{1} \odot \bar{0} \odot \bar{4} \odot \bar{1} \odot \bar{6} \odot \bar{1} \odot \bar{7} = \bar{7} \odot \bar{1} \odot \bar{3} \odot \bar{1} \odot \bar{0} \odot \bar{1} \odot \bar{0} \odot \bar{0} = 0$

$c_2 = 2 \odot \bar{0} \odot \bar{2} \odot \bar{4} \odot \bar{2} \odot \bar{6} \odot \bar{2} \odot \bar{7} = 2 \odot \bar{2} \odot \bar{3} \odot \bar{0} \odot \bar{2} \odot \bar{0} \odot \bar{0} = 1$

$c_3 = 0, c_4 = 0, c_5 = 1$, and $c_7 = 0$.

The truth vector $c$ for zero polarity is given as follows:

$c = [7,5,4,3,1]$.

The sum terms in this canonical form can be generated by using equation (8) to get the PPDRM form will lead to

$f(x_2x_1x_0) = (0) \odot (x_1) \odot (x_1 + x_0) \odot (x_2) \odot (x_2 + x_1)$

4 Exact Minimisation of FPDRM

In the optimisation of DRM functions, polynomials with different polarities are usually calculated directly from POS expressions [5-7]. A new algorithm is presented to generate all of the polarity
sets from any polarity set \( q \). Time efficiency is achieved in this technique because the information utilized in finding DRM expansion of one polarity is utilized by others. Two-fixed polarities can be derived from each other without the need to go back to the original Boolean function, if the two polarities are dual.

**Corollary 3.1** The DRM with a fixed polarity \( q_i \) can be derived from DRM with a fixed polarity \( p_j \), where \( p_j \) and \( q_i \) are dual polarities and \( j \) is the permuting bit.

**Proof** From equation (2) with polarity \( p = p_j \), DRM is given as:

\[
 f(p_j) (x_{n-1}, x_{n-2}, \ldots, x_0) = \bigodot \prod_{j=0}^{n-1} (x_j^{p_j} + x_j^{\bar{p}_j}) \tag{13}
 \]

\[
 S_j^{p_j} = \sum_{k=0}^{\frac{n_k}{2}} x_k = x_{n-1} \pm x_{n-2} \pm \ldots \pm x_0 \tag{14}
 \]

\( p_j \), is any given polarity = \( (0,\ldots,2^n-1) \), \( S_j^{(p_j)} \) are the sum terms for the particular polarity \( p_j \).

In order to change polarity from \( p_j \) to \( q_j \), equation (14) can be expressed as

\[
 S_j^{q_j} = \sum_{k=0}^{\frac{n_k}{2}} (x_k)^j + (x_k)^\bar{j} \tag{15}
 \]

Each variable \( x_k \) in equation (15) will remain as it is, except variable \( x_j \) is replaced by the following identity \((x_j)^j = 0 \odot (x_j)^\bar{j}\). Therefore, equation (15) becomes the following equation for \( S_j^{q_j} \)

\[
 S_j^{q_j} = \sum_{k=0}^{\frac{n_k}{2}} \sum_{k=0}^{\frac{n_k}{2}} (x_k)^j + (x_k)^\bar{j} \tag{16}
 \]

By using the following property:

\[
 A+ (0 \odot B) + C = (A+C) + (0 \odot B)
 = (A+C+0) \odot (A+C+\bar{B})
 \]

Therefore,

\[
 S_j^{q_j} = \sum_{k=0}^{\frac{n_k}{2}} (x_k)^j + \sum_{k=0}^{\frac{n_k}{2}} (x_k)^\bar{j} \tag{17}
 \]

\[
 S_j^{q_j} = \sum_{k=0}^{\frac{n_k}{2}} (x_k)^j + (x_k)^\bar{j} \tag{18}
 \]

Substituting (17) in (13), we obtain the following expansion for \( q_j \) with \( p = q \).

\[
 f(q_j) = \bigodot \bigotimes_{k=0}^{\frac{n_k}{2}} (x_k)^j + (x_k)^\bar{j} \tag{19}
 \]

To convert polarity \( p_j \) to polarity \( q_j \), each sum in the FPDRM with polarity \( p_j \) is converted into a binary string. A zero is placed in the binary string if the variable is present and a one if the variable is absent. The new term is generated, by copying all the binary string except for bit \( j \). If bit \( j \) is zero change it to one. Duplicate terms are deleted according to the rule \( B \odot \bar{B} = 1 \). Based on this, the following algorithm is developed.

**Algorithm 2**

This algorithm converts between polarities and identifies the polarity with the least number of terms. The following steps shall be used to derive the coefficient set \( \psi_q \) from the dual set \( \phi_p \). The steps are repeated for the rest of the polarities till the best polarity is obtained.

**Step 1**: Use Algorithm 1 to calculate the coefficients for PPDRM function. Set \( \phi_{\min} \) = the number of offset coefficients for polarity \( p_j \).

**Step 2**: Determine the next polarity \( q_j \) in Gray code order, where polarities \( p_j \) and \( q_j \) are dual and differ in bit \( j \) only.

**Step 3**: Converts the sum terms for polarity \( p_j \) into a binary string. By replacing each variable by ‘0’ if the variable is present in the sum term or by ‘1’ if it is absent.

**Step 4**: For each term in polarity \( p_j \), generate a new term if bit \( j \) of the binary string is ‘0’. Replace bit \( j \) with ‘1’ and copy all others bits to generate the new term.

**Step 5**: Delete common pairs between original strings and newly generated strings because \( B \odot B = 1 \).

**Step 6**: Convert the affected strings are the product terms of the new polarity \( q_j \).

**Step 7**: Count the total number of zero coefficients \( \psi_{\min} \) for polarity \( q_j \).

If \( \psi_{\min} < \phi_{\min} \) then \( \phi_{\min} = \psi_{\min} \).

**Step 8**: Stop if all polarities have been checked. Otherwise go to step 2.

**Example 4**: Find the best polarity for function \( f(x_2,x_1,x_0) = \Pi M(0,4,6,7) \)

Step 1, the following results are obtained for PPDRM using algorithm 1.

\[
 f(x_2,x_1,x_0) = \odot \Pi (7,5,4,3,1)
 \]

In order to find all polarities for DRM expansion, a Gray code sequence is generated for \( n = 3 \), then Algorithm 2 is applied to find the best polarity with the least number of terms.

\[
 f(x_2,x_1,x_0) = \odot \Pi (7,5,4,3,1)
 \]

Count the number of coefficients from step 1, and set \( \phi_p = 5 \) since this the first step in the procedure set also \( \phi_{\min} = \phi_p \).

Step 2, since polarity ‘0’ = (000) and polarity ‘1’ = (001) are dual polarities. Hence DRM in polarity ‘1’
can be derived directly from equation (18) using Algorithm 2 as shown in Table 2 where the altered bit is at \( j \) equals 0.

**Table 2: Derivation of DRM for Polarity 1**

<table>
<thead>
<tr>
<th>Polarity 0 ( x_2x_1x_0 )</th>
<th>New terms</th>
<th>Polarity 1 ( x_2\overline{x}_1\overline{x}_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>111</td>
<td>111</td>
</tr>
<tr>
<td>(-101−)</td>
<td>100</td>
<td>(-101−)</td>
</tr>
<tr>
<td>100</td>
<td>(-101−)</td>
<td>011</td>
</tr>
<tr>
<td>011</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( f(x_2,x_1,\overline{x}_0) = \circ \prod (7,4,3,1) \)

\( \psi_{\text{min}} = 4 \), since \( \psi_{\text{min}} < \phi_{\text{min}} \) then record the corresponding polarity as \( P_{\text{min}} = q_j = 1 \) and \( \phi_{\text{min}} = 4 \).

Repeat step 2 to convert from polarity ‘1’ = (001) to ‘3’ = (011), with the altered bit \( j = 1 \) as shown in Table 3.

**Table 3: Derivation of DRM for Polarity 3**

<table>
<thead>
<tr>
<th>Polarity 1 ( x_2x_1\overline{x}_0 )</th>
<th>New terms</th>
<th>Polarity 3 ( x_2\overline{x}_1x_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>111</td>
<td>111</td>
</tr>
<tr>
<td>100</td>
<td>110</td>
<td>110</td>
</tr>
<tr>
<td>(-111−)</td>
<td>100</td>
<td>(-111−)</td>
</tr>
<tr>
<td>001</td>
<td></td>
<td>001</td>
</tr>
</tbody>
</table>

\( f(x_2,\overline{x}_1,\overline{x}_0) = \circ \prod (7,6,4,1) \)

\( \psi_{\text{min}} = 4 \), since \( \psi_{\text{min}} = \phi_{\text{min}} \) then go to the next polarity.

Repeat step 2 to convert from polarity ‘3’ = (011) to ‘2’ = (010), with the altered bit \( j = 0 \) as shown in Table 4.

**Table 4: Derivation of DRM for Polarity 2**

<table>
<thead>
<tr>
<th>Polarity 3 ( \overline{x}_2\overline{x}_1\overline{x}_0 )</th>
<th>New terms</th>
<th>Polarity 2 ( \overline{x}_2x_1x_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-111−)</td>
<td></td>
<td>110</td>
</tr>
<tr>
<td>110</td>
<td>(-111−)</td>
<td>101</td>
</tr>
<tr>
<td>100</td>
<td>101</td>
<td>100</td>
</tr>
<tr>
<td>001</td>
<td></td>
<td>001</td>
</tr>
</tbody>
</table>

\( f(x_2,\overline{x}_1,x_0) = \circ \prod (6,5,4,1) \)

\( \psi_{\text{min}} = 4 \), since \( \psi_{\text{min}} = \phi_{\text{min}} \) then go to the next polarity.

Repeat step 2 to convert from polarity ‘2’ = (010) to ‘6’ = (110), with the altered bit \( j = 2 \) as shown in Table 5.

**Table 5: Derivation of DRM for Polarity 6**

<table>
<thead>
<tr>
<th>Polarity 2 ( x_2\overline{x}_1.0 )</th>
<th>New terms</th>
<th>Polarity 6 ( \overline{x}_2x_1x_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td></td>
<td>110</td>
</tr>
<tr>
<td>(-101−)</td>
<td></td>
<td>(-101−)</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>001</td>
</tr>
<tr>
<td>001</td>
<td>(-101−)</td>
<td>001</td>
</tr>
</tbody>
</table>

\( f(\overline{x}_2,\overline{x}_1,x_0) = \circ \prod (6,4,1) \)

\( \psi_{\text{min}} = 3 \), since \( \psi_{\text{min}} < \phi_{\text{min}} \) then record the corresponding polarity as \( P_{\text{min}} = q_j = 6 \) and change \( \phi_{\text{min}} \) to 3.

Repeat step 2 to convert from polarity ‘6’ = (110) to ‘7’ = (111), with the altered bit \( j = 0 \) as shown in Table 6.

**Table 6: Derivation of DRM for Polarity 7**

<table>
<thead>
<tr>
<th>Polarity 6 ( \overline{x}_2x_1x_0 )</th>
<th>New terms</th>
<th>Polarity 7 ( \overline{x}_2\overline{x}_1\overline{x}_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td></td>
<td>111</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>110</td>
</tr>
<tr>
<td>001</td>
<td></td>
<td>101</td>
</tr>
<tr>
<td>(-101−)</td>
<td></td>
<td>001</td>
</tr>
</tbody>
</table>

\( f(\overline{x}_2,\overline{x}_1,\overline{x}_0) = \circ \prod (7,6,5,4,1) \)

\( \psi_{\text{min}} = 5 \), since \( \psi_{\text{min}} > \phi_{\text{min}} \) then go to the next polarity.

Repeat step 2 to convert from polarity ‘7’ = (111) to ‘5’ = (101), with the altered bit \( j = 1 \) as shown in Table 7.

**Table 7: Derivation of DRM for Polarity 5**

<table>
<thead>
<tr>
<th>Polarity 7 ( \overline{x}_2\overline{x}_1\overline{x}_0 )</th>
<th>New terms</th>
<th>Polarity 5 ( \overline{x}_2x_1x_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-111−)</td>
<td></td>
<td>101</td>
</tr>
<tr>
<td>(-110−)</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>101</td>
<td>(-111−)</td>
<td>011</td>
</tr>
<tr>
<td>100</td>
<td>(-110−)</td>
<td>001</td>
</tr>
<tr>
<td>001</td>
<td></td>
<td>011</td>
</tr>
</tbody>
</table>

\( f(\overline{x}_2,x_1,\overline{x}_0) = \circ \prod (5,4,3,1) \)

\( \psi_{\text{min}} = 4 \), since \( \psi_{\text{min}} > \phi_{\text{min}} \) then stop.

Therefore the best polarity for this function is \( p = 6 = (110) \) with 3 terms.
\[ f(\bar{x}_2, \bar{x}_1, x_0) = \bigodot \bigoplus (6,4,1) \]
The sum terms for this canonical form can be generated as follows:
\[
[0 \ x_2] \ast [0 \ x_1] \ast [0 \ x_0] = \\
[0 \ x_0 \ x_1 \bar{x}_1 + x_0 \ x_2 \bar{x}_2 + x_0 \bar{x}_2 + x_1 \bar{x}_2 + x_0] \\
(\bar{x}_2 +\bar{x}_1 + x_0)
\]
Using equation (8) the following equation is obtained for FPDRM form
\[
\text{where the truth vector } c = [1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1] \\
f(\bar{x}_2, \bar{x}_1, x_0) = x_0 \ast (\bar{x} + x_0) \ast (\bar{x}_2 + \bar{x}_1)
\]

### 5 Experimental Results

In this section, experimental results are presented using algorithm one. The proposed algorithm is implemented in C language and the programs compiled using Borland C++ compiler. The program was tested on a personal computer with Pentium 4 processor of 2.4 GHz CPU and 512 MB of RAM under Window operating system. The algorithm was applied to several MCNC benchmarks and some random functions. Table 8 shows the results obtained from converting POS coefficients into DRM coefficients. Where name denotes the name of circuit, \( n \) denotes the number of variables, Init terms denote the number of terms in POS form, DRM terms denote the number of terms in DRM form, the execution time (CPU Time (s)) is time required to calculate the coefficients of the Fixed Polarity Dual Reed-Muller form the coefficients of the Product of Sums and it is given in seconds. For most of the circuits with \( n \) less than 16 the CPU time is less than 1 seconds. The CPU time depends on the variable number \( n \) and the initial number of terms (number of off-set coefficients). For large Boolean functions, there are many coefficients and they should be accessed once. This algorithm can be improved by ordering the off-set coefficients in advance. This is can be achieved by introducing a multiple segment technique, then the execution time will be improved and it could handle large number of variables.

For incompletely specified Boolean functions, ‘don’t care’ are set to ‘1’. The experimental results obtained in Table 8 reflect the efficiency of the algorithm. To our knowledge there are no publications on this topic to compare with.

### 6 Conclusions

This paper presents an efficient method to find the best polarity of DRM expansions. A fast and simple algorithm has been developed to facilitate derivation of DRM expansions based on dual polarities without constructing \( T_{\text{M}_n} \) matrix. To compute each coefficient using \( T_{\text{M}_n} \) requires \( 2^n \) OR plus \( 2^n-1 \) XNOR operations. \( T_{\text{M}_n} \) can be very large for large values of \( n \) and may require a large memory since the size of \( T_{\text{M}_n} \) is \( 2^n \) by \( 2^n \). The proposed method is simple and doesn’t require storing \( T_{\text{M}_n} \) matrix, which saves memory and CPU time.

<table>
<thead>
<tr>
<th>Name</th>
<th>( n )</th>
<th>Init. terms in POS</th>
<th>DRM Terms</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Con1</td>
<td>7</td>
<td>88</td>
<td>9</td>
<td>0.000</td>
</tr>
<tr>
<td>Rd84</td>
<td>8</td>
<td>136</td>
<td>37</td>
<td>0.050</td>
</tr>
<tr>
<td>Apex4</td>
<td>9</td>
<td>534</td>
<td>181</td>
<td>0.000</td>
</tr>
<tr>
<td>Clip</td>
<td>9</td>
<td>480</td>
<td>92</td>
<td>0.000</td>
</tr>
<tr>
<td>Ex1010</td>
<td>10</td>
<td>142</td>
<td>480</td>
<td>0.010</td>
</tr>
<tr>
<td>F12†</td>
<td>12</td>
<td>6</td>
<td>255</td>
<td>0.010</td>
</tr>
<tr>
<td>F13†</td>
<td>13</td>
<td>5</td>
<td>220</td>
<td>0.000</td>
</tr>
<tr>
<td>Table3</td>
<td>14</td>
<td>1859</td>
<td>2528</td>
<td>0.101</td>
</tr>
<tr>
<td>spla</td>
<td>16</td>
<td>5348</td>
<td>517</td>
<td>0.931</td>
</tr>
<tr>
<td>Table5</td>
<td>17</td>
<td>28552</td>
<td>3359</td>
<td>9.845</td>
</tr>
</tbody>
</table>

†Randomly generated Boolean function where \( F_{12} = \prod M(0,500,1200,1900,2500,3000) \) and \( F_{13} = \prod M(500,1200,3500,4500,8000) \)

References:


