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

Where \([\prod]\) represents logical products (AND), the \(\{+\}\) is OR operation and \(i\) is a binary \(n\)-tuple \(i = \{i_0, i_1, \ldots, i_{n-1}\}\). \([d_0, d_1, \ldots, d_n]\) is the truth vector of the function \(f\), \(d_i \in \{0, 1\} \) \([12]\), \(M_i\) is a sum term

\[
M_i = \sum_{k=0}^n x_k = x_{n-1} + x_{n-2} + \cdots + x_0
\]

and
\[
x_k = \begin{cases} x_k & i_k = 0 \\ \bar{x}_k & i_k = 1 \end{cases}
\]

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

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

Where \(\otimes\) is XNOR operator, \([c_0, c_1, c_2, \ldots, c_0]\) is the truth vector of the function \(f\), \(c_i \in \{0, 1\}\), \(i = \{i_0, i_1, \ldots, i_{n-1}\}\), \(S_i\) represents a Sum term as

\[
S_i = \sum_{k=0}^n \bar{x}_k = x_{n-1} + x_{n-2} + \cdots + x_0
\]

and
\[
x_k = \begin{cases} 0 & i_k = 0 \\ \bar{x}_k & i_k = 1 \end{cases}
\]

**Definition 2.2** Polarity vector \([p_0, p_1, p_2, \ldots, p_n]\) for a FPDRM of an \(n\)-variable Boolean function is a binary vector with \(n\) elements, where \(p_i = 0\) indicates the variable \(x_i\) in an un-complemented form \((x_i)\), while \(p_i = 1\) indicates the variable \(x_i\) in the complemented form \((\bar{x}_i)\).

**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 \(e\) in equation (4) for a zero polarity. Thus, for \(n = 2\) and \(P = 0\)

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

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

Where \(\otimes\) represents matrix multiplication based on OR and XNOR \([9-11]\).

### 3 Conversion Algorithms from POS to FPDRM and vice versa

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

\[
f(x_{n-1}, x_{n-2}, \ldots, x_0) = (d_0 + x_{n-1} + x_{n-2} + \cdots + x_0) \otimes (d_1 + x_{n-1} + x_{n-2} + \cdots + x_0) \cdots \otimes (d_n + x_{n-1} + x_{n-2} + \cdots + x_0)
\]

(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 Maxterms 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.

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

(6)

\[
f(x_{n-1}, x_{n-2}, \ldots, x_0) = (d_0 + x_{n-1} + x_{n-2} + \cdots + x_0) \otimes (d_1 + x_{n-1} + x_{n-2} + \cdots + x_0) \cdots \otimes (d_n + x_{n-1} + x_{n-2} + \cdots + x_0)
\]

(7)

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

\[
T = [d_0, d_1, \ldots, d_n]
\]

(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 \(T(x_i)\). The first row of the separation matrix \(T(x_i)\) 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 $f(x_0,x_1,x_2,x_3) = \prod M(0,4,6,7,11,15)$ and use the truth vector $T$ to generate the separation matrix for each variable $x_i$.

The truth vector $T$ has 2$^4$ 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 $T(x_i)$, the truth vector $T$ is separated around variable $x_3$ into 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:

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

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 $T(x_i)$, the truth vector $T$ is separated around variable $x_3$ which gives the following result:

$T(x_i) = \begin{bmatrix} 0111 & 0100 \\ 1110 & 1110 \end{bmatrix} x_i$

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

$T(x_2) = \begin{bmatrix} 0111 & 1110 \\ 0100 & 1110 \end{bmatrix} x_2$

$T(x_1) = \begin{bmatrix} 0101 & 1111 \\ 1100 & 1010 \end{bmatrix} x_1$

$T(x_0) = \begin{bmatrix} 0100 & 1111 \\ 1110 & 1010 \end{bmatrix} x_0$

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

$$(a + \bar{x}_i) \otimes (b + x_i) = [a + (0 \otimes x_i)] \otimes (b + x_i)$$

$$= [(a + 0) \otimes (a + x_i)] \otimes (b + x_i)$$

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

However

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

This can be verified as follows:

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

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

Where ‘$\oplus$’ is XOR operator.

Complementing the last expression, the following is obtained:

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

Therefore,

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

Hence

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

Examining equation (9), the coefficients of the un-complemented part of variable $x_i$ takes a new form. The new coefficient is $(a \overline{\otimes} b)$, 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 $0 \otimes \overline{x}_i$.

$$(a + \bar{x}_i) \otimes (b + x_i) = (a + \bar{x}_i) \otimes [(b + 0) \otimes (b + \bar{x}_i)]$$

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

By taking the complement of the following expression

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

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

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

Hence

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

Therefore,
(a + x_i) \otimes (b + x_i) = b \otimes [(a \otimes b) + x_i] \quad (10)

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

**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 \(\overline{x_i}\) 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 \(N\). The new truth vector is \(T = [0111 \ 0111 \ 0111 \ 0111]\).

**Step 5**: If the required polarity for \(x_i\) variable is ‘1’ then replace the contents of each complemented part of \(x_i\) variable in \(T\) by the contents of the un-complemented part of \(x_i\) variable and store the result \(N\) in place of un-complemented part of \(x_i\) variable. The new truth vector is \(T = [1110 \ 0111 \ 1111 \ 1110]\).

**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.

\[
\begin{bmatrix}
0111 & 0110 \\
1110 & 1110 \\
\end{bmatrix}
\xNOR
\begin{bmatrix}
1110 & 1110 \\
0110 & 0111 \\
\end{bmatrix}
\]

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:

\[
\begin{bmatrix}
0110 & 1110 \\
0111 & 1110 \\
\end{bmatrix}
\xNOR
\begin{bmatrix}
1111 & 1111 \\
0110 & 0111 \\
\end{bmatrix}
\]

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 \(x_0\)

\[
\begin{bmatrix}
1101 & 1111 \\
1010 & 1110 \\
\end{bmatrix}
\xNOR
\begin{bmatrix}
1110 & 1111 \\
0111 & 1011 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1100 & 0110 \\
1000 & 1110 \\
\end{bmatrix}
\xNOR
\begin{bmatrix}
0111 & 1110 \\
0111 & 1011 \\
\end{bmatrix}
\]

The final \(T\) vector is \(T = [0111 \ 0110 \ 1111 \ 0111] = \{0,5,6,7,12\}\)

The sum terms in this canonical can be generated by using the basis vector

\[
\{x_3, x_2, x_1, x_0\}
\]
Separate the truth vector in the complemented part of variable \( B \), the new truth vector is:

\[
\begin{align*}
(\overline{x}_3 + \overline{x}_1) \& (\overline{x}_2 + \overline{x}_0) \& (x_3 + \overline{x}_1) \& (x_2 + \overline{x}_0) \\
(\overline{x}_3 + \overline{x}_1) \& (\overline{x}_2 + \overline{x}_0) \& (x_3 + \overline{x}_1) \& (x_2 + \overline{x}_0) \\
(\overline{x}_3 + \overline{x}_1) \& (\overline{x}_2 + \overline{x}_0) \\
(x_3 + \overline{x}_1) \& (x_2 + \overline{x}_0)
\end{align*}
\]

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

\[
f(x_{n1}, x_{n2}, \ldots, x_0) = [0 x_{n1}] ++ [0 x_{n2}] ++ \ldots ++ [0 x_0] \ 
\]

Hence

\[
f(x_3, x_2, x_1, x_0) = 0 \odot (\overline{x}_3 + \overline{x}_1) \otimes (\overline{x}_2 + \overline{x}_0) \\
\odot (\overline{x}_3 + \overline{x}_1 + \overline{x}_0) \odot (x_3 + \overline{x}_1 + \overline{x}_0)
\]

Example 3

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

Store the truth vector in \( T \) matrix.

\[
T = \begin{bmatrix} 0111 & 1000 & 1111 & 0111 \end{bmatrix}
\]

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

\[
T(x_3) = \begin{bmatrix} 0111 & 1000 & \overline{x}_3 \end{bmatrix}
\]

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

\[
T = \begin{bmatrix} 0111 & 0000 & 1111 & 0111 \end{bmatrix}
\]

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

\[
T(x_2) = \begin{bmatrix} 0111 & 1111 & \overline{x}_2 \end{bmatrix}
\]

Since polarity is ‘1’ for variable \( x_2 \) apply Algorithm B, the new truth vector is

\[
T = \begin{bmatrix} 0000 & 1000 & 0111 & 0111 \end{bmatrix}
\]

Similarly for variable \( x_1 \) and \( x_0 \)

\[
T(x_1) = \begin{bmatrix} 0010 & 0101 & \overline{x}_1 \end{bmatrix}
\]

\[
T = \begin{bmatrix} 0011 & 0001 & 1101 & 1101 \end{bmatrix}
\]

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

\[
f(x_3, x_2, x_1, x_0) = \bigoplus M(0, 4, 7, 11, 15).
\]

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 \( P_{\text{min}} \) 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 \( \overline{x}_i \) 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. \(P_{\min}\) 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 \(x\) 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 \(P_{\min} = 0\).

The result is as follows: \(T = \begin{bmatrix} 1010 & 1100 & 0010 & 0110 & 1111 & 1111 & 1111 & 1111 \end{bmatrix}\)

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

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:

\[
T(x_4) = \begin{bmatrix} 1010 & 1100 & 0010 & 0110 & 1111 & 1111 & 1111 & 1111 \end{bmatrix} x_i (x_i \text{ XNOR } \tau_i)
\]

\[
N = \begin{bmatrix} 1010 & 1100 & 0010 & 0110 & 1111 & 1111 & 1111 & 1111 \end{bmatrix}
\]

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 \(x_0\)

\[
T(x_3) = \begin{bmatrix} 1010 & 1100 & 1111 & 1111 \end{bmatrix} x_i (x_i \text{ XNOR } \tau_i)
\]

\[
N = \begin{bmatrix} 1010 & 1100 & 1111 & 1111 \end{bmatrix}
\]

\(NZ(x_3) = 14\)

\[
T(x_2) = \begin{bmatrix} 1011 & 0101 & 1111 & 1111 \end{bmatrix} x_i (x_i \text{ XNOR } \tau_i)
\]

\[
N = \begin{bmatrix} 1001 & 1101 & 1111 & 1111 \end{bmatrix}
\]

\(NZ(x_2) = 6\)

\[
T(x_1) = \begin{bmatrix} 1110 & 0100 & 1111 & 1111 \end{bmatrix} x_i (x_i \text{ XNOR } \tau_i)
\]

\[
N = \begin{bmatrix} 0011 & 0100 & 1111 & 1111 \end{bmatrix}
\]

\(NZ(x_1) = 6\)

\[
T(x_0) = \begin{bmatrix} 1110 & 0100 & 1111 & 1111 \end{bmatrix} x_i (x_i \text{ XNOR } \tau_i)
\]

\[
N = \begin{bmatrix} 0011 & 0100 & 1111 & 1111 \end{bmatrix}
\]

Since \(NZ(x_1)\) 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 \(T = \begin{bmatrix} 1010 & 1100 & 0010 & 0110 & 1111 & 1111 & 1111 & 1111 \end{bmatrix}\).

The total number of zeros for the new vector \(TNZ = 7\), and the polarity number for this vector is \(P_{\min} = \{01000\} = 8\).

Similarly as in the previous part, separate vector \(T\) around each variable but not \(x_3\) because it has been changed to \(\overline{x}_3\).
converted to \( x_3 \). Hence the new \( T \) vector is: \( T = [1010 1001 0111 1101 1111 1111 1111 1111] \).

The total number of zeros for the new vector \( \text{TNZ} = 6 \) and \( P_{\text{min}} = \{01100\} = 12 \).

Repeat the same procedure for \( x_4, x_1 \) and \( x_0 \).

\[
T(x_4) = \begin{bmatrix} 1010 & 1001 & 0111 & 1101 \\ 1111 & 1111 & 1111 & 1111 \end{bmatrix} x_4 \, (x_4 \text{XNOR} \, \overline{x}_4) \\
N = \begin{bmatrix} 1010 & 1001 & 0111 & 1101 \end{bmatrix}
\]

\( \text{NZ} \, (x_4) = 12 \)

\[
T(x_1) = \begin{bmatrix} 1010 & 0111 & 1111 & 1111 \\ 1001 & 1101 & 1111 & 1111 \end{bmatrix} x_1 \, (x_1 \text{XNOR} \, \overline{x}_1) \\
N = \begin{bmatrix} 1100 & 0101 & 1111 & 1111 \end{bmatrix}
\]

\( \text{NZ} \, (x_1) = 7 \)

\[
T(x_0) = \begin{bmatrix} 1110 & 0110 & 1111 & 1111 \\ 0001 & 1111 & 1111 & 1111 \end{bmatrix} x_0 \, (x_0 \text{XNOR} \, \overline{x}_0) \\
N = \begin{bmatrix} 0000 & 0110 & 1111 & 1111 \end{bmatrix}
\]

\( \text{NZ} \, (x_0) = 9 \)

Since the total number of zeros from each operation is greater than \( \text{TNZ} \) 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 \( A \, \text{XNOR} \, B \) 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>
<thead>
<tr>
<th>Name</th>
<th>( n )</th>
<th>PPDRM Terms</th>
<th>POS Terms</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clip</td>
<td>9</td>
<td>92</td>
<td>480</td>
<td>0.001</td>
</tr>
<tr>
<td>Con1</td>
<td>7</td>
<td>9</td>
<td>88</td>
<td>0.001</td>
</tr>
<tr>
<td>Ex1010</td>
<td>10</td>
<td>480</td>
<td>142</td>
<td>0.010</td>
</tr>
<tr>
<td>Rd84</td>
<td>8</td>
<td>37</td>
<td>136</td>
<td>0.001</td>
</tr>
<tr>
<td>Table3</td>
<td>14</td>
<td>2528</td>
<td>1859</td>
<td>1.29</td>
</tr>
<tr>
<td>Table5</td>
<td>17</td>
<td>3359</td>
<td>28552</td>
<td>4.927</td>
</tr>
</tbody>
</table>
### Table 2: Conversion table from POS to PPDRM

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

### References


