Exact Minimization of Fixed Polarity Reed-Muller Expressions for Incompletely Specified Functions

Debatosh Debnath and Tsutomu Sasao
Department of Computer Science and Electronics
Kyushu Institute of Technology
Iizuka 820-8502, Japan

Abstract—This paper presents an exact minimization algorithm for fixed polarity Reed-Muller expressions (FPRMs) for incompletely specified functions. For an n-variable function with α unspecified minterms there are $2^{n+\alpha}$ distinct FPRMs. A minimum FPRM is one with the fewest products. The minimization algorithm is based on the multi-terminal binary decision diagrams. Experimental results for a set of functions are shown. The algorithm can be extended to obtain exact minimum Kronecker expressions for incompletely specified functions.

Index Terms—AND-EXOR, Reed-Muller expression, Kronecker expression, exact minimization, incompletely specified function.

I. Introduction

Fixed polarity Reed-Muller expression (FPRM) is one of the canonical AND-EXOR expressions [15]. FPRMs are a generalization of positive polarity Reed-Muller expressions (PPRMs). A PPRM, which is unique for a completely specified function, is an AND-EXOR expression with only uncomplemented literals. PPRMs are also known as Zhegalkin polynomials after the Russian logician Ivan I. Zhegalkin who first published this canonical form [27]. Each variable in an FPRM can appear either in complemented or uncomplemented form. An n-variable completely specified function has 2^n distinct FPRMs. For incompletely specified function, the number of FPRMs increases exponentially with the increase in the number of unspecified minterms: $2^{n+\alpha}$ distinct FPRMs exist for an n-variable function with α unspecified minterms. An expression with the fewest products is a minimum expression.

FPRMs are important because they can be used to design easily testable circuits [14], to detect symmetric variables of switching functions [22], to design multilevel circuits [23], and in Boolean matching [24]. Moreover, for some classes of practical functions, FPRMs require fewer products than sum-of-products expressions (SOPs) [15–18].

For completely specified functions, numerous exact and heuristic minimization algorithms for FPRMs exist [7, 10, 14, 17, 21]. However, little research has been done to minimize FPRMs for incompletely specified functions. Tran discussed a graphical procedure, which is based on a trial-and-error method, to simplify FPRMs for incompletely specified functions [20]. The method can be applicable to functions with up to six variables. By using spectral techniques [9], Varma and Trachtenberg developed heuristic algorithms to simplify PPRMs for incompletely specified functions [25]. Chang and Falkowski reported methods to simplify FPRMs for incompletely specified functions [3, 4]. Recently, Zilic and Vranesic presented a heuristic scheme to compute multiple-valued Reed-Muller transform for incompletely specified functions [28].

McKenzie et al. developed a branch and bound algorithm for the exact minimization of PPRMs for incompletely specified functions [13]. Green described an exhaustive search method [8]. Zakrevskij formulated the exact minimization of PPRMs for incompletely specified functions as a solution of a system of linear logical equations, and presented experimental results for functions with up to 20 specified minterms [26]. McKenzie et al. [13] and Zakrevskij [26] also considered heuristic simplification methods.

In this paper we present an algorithm to obtain exact minimum FPRMs for incompletely specified functions. The method is based on the computation of extended truth vector and weight vector, which are also used for the exact minimization of FPRMs for completely specified functions [6, 17]. Every component of these vectors is an integer-valued function represented by multi-terminal binary decision diagram (MTBDD) [5]. Kronecker expressions [6, 12, 18], introduced by Bioul et al. [1], are a generalization of FPRMs. We also discuss an extension of the FPRM minimization algorithm to minimize Kronecker expressions for incompletely specified functions.

The remainder of the paper is organized as follows: Section II introduces terminology and presents basic properties. Section III develops an exact minimization algorithm. Section IV reports experimental results. Section V presents conclusion and outlines future work.

II. Definitions and Basic Properties

In this paper, the operators '+' and '$\oplus$' indicate arithmetic and mod-2 addition, respectively.

Definition 2.1 An n-variable ***switching function*** f is a mapping $f:\{0{,}1\}^n \rightarrow \{0{,}1\}$; and an n-variable ***integer-valued function*** g is a mapping $g:\{0{,}1\}^n \rightarrow \{0{,}1,\ldots,p-1\}$, where $p \geq 2$.

It should be noted that switching functions are a subset of integer-valued functions.

Definition 2.2 An n-variable integer-valued function $f(x_1,x_2,\ldots,x_n)$ can be written as $\sum_{j=0}^{2^n-1} m_j x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n}$, where $m_j \in \{0{,}1,\ldots,p-1\}$ $(p \geq 2)$, $b_1,b_2,\ldots,b_n \in \{0{,}1\}$ such that $b_1 b_2 \cdots b_n$ is the n-bit binary number representing j, $x_i^{b_i} = \bar{x}_i$ when $b_i = 0$, $x_i^{b_i} = x_i$ when $b_i = 1$, and $i = 1,2,\ldots,n$. Then $[m_0,m_1,\ldots,m_{2^n-1}]$ is the ***truth vector*** of f.

Example 2.1 The truth vector of the three-variable switching function $\bar{x}_1\bar{x}_2\bar{x}_3 \vee x_1$ is [1,0,0,0,1,1,1,1], and that of the three-variable integer-valued function $3x_1 + 4x_2x_3 + 2\bar{x}_3$ is [2,0,2,4,5,3,5,7].

Property 2.1 Let f be a switching function. Then $f + \bar{f} = 1$.

Example 2.2 Let the two-variable switching function f be [1,0,0,0]. Then $f + \bar{f} = [1{,}0{,}0{,}0] + [0{,}1{,}1{,}1] = [1{,}1{,}1{,}1] = 1$.

Property 2.2 Let f be an integer-valued function. Then $\underbrace{f+f+\cdots+f}_{k \text{ operands}} = k \cdot f$.

Example 2.3 Let the two-variable integer-valued function f be [1,0,3,5]. Then $f+f+f = 3 \cdot [1{,}0{,}3{,}5] = [3{,}0{,}9{,}15]$.

Definition 2.3 An n-variable switching function $f(x_1,x_2,\ldots,x_n)$ can be written as a ***fixed polarity Reed-Muller expression (FPRM)*** $\sum\oplus_{j=0}^{2^n-1} a_j x_1^{b_1} x_2^{b_2} \cdots x_n^{b_n}$, where $a_j \in \{0{,}1\}$, $b_1,b_2,\ldots,b_n \in \{0{,}1\}$ such that $b_1 b_2 \cdots b_n$ is the n-bit binary number representing j, $x_i^{b_i} = 1$ when $b_i = 0$, $x_i^{b_i} \in \{\bar{x}_i, x_i\}$ such that for each i either $\bar{x}_i$ or x_i appear throughout the expression when $b_i = 1$, and $i = 1,2,\ldots,n$.

In an FPRM, each variable can appear either in complemented or uncomplemented form, i.e., polarity of each variable can be chosen in two ways. Thus, for an n-variable completely specified function there are 2^n distinct FPRMs.

Definition 2.4 ***Polarity vector*** $(b_1,b_2,\ldots,b_n)$ for an FPRM of an n-variable switching function $f(x_1,x_2,\ldots,x_n)$ is a binary vector with n elements, where $b_i = 0$ indicates variable x_i is used in the uncomplemented form (x_i) and $b_i = 1$ indicates variable x_i is used in the complemented form $(\bar{x}_i)$.

Example 2.4 Let the three-variable switching function f be $x_1x_3 \vee \bar{x}_2\bar{x}_3$ and (0,1,1) be a polarity vector for an FPRM of f. Since x_1x_3 and $\bar{x}_2\bar{x}_3$ are disjoint, we can write $f = x_1x_3 \oplus \bar{x}_2\bar{x}_3$. By putting $x_3 = 1 \oplus \bar{x}_3$ in the expression for f, we have $f = x_1(1 \oplus \bar{x}_3) \oplus \bar{x}_2\bar{x}_3 = x_1 \oplus x_1\bar{x}_3 \oplus \bar{x}_2\bar{x}_3$, which is the FPRM for f with polarity vector (0,1,1).

III. Minimization Techniques

For an n-variable completely specified switching function there are 2^n distinct FPRMs, and the minimization problem is to find a polarity vector that produces an FPRM with minimum number of products. On the other hand, for an n-variable incompletely specified switching function with α unspecified minterms there are $2^{n+\alpha}$ distinct FPRMs, and the minimization problem is to find a polarity vector and an assignment of the unspecified minterms to 0's and 1's that produce an FPRM with minimum number of products. Once the polarity vector and the assignment of the unspecified minterms are determined, generation of an FPRM is relatively easy [6, 17].

Fig. 1 illustrates a method for the exact minimization of FPRMs for three-variable switching function. The method is based on the computation of extended truth vector and weight vector [6, 17]. The extended truth vector $[t_0,t_1,\ldots,t_{26}]$ is computed from the truth vector $[m_0,m_1,\ldots,m_7]$ of a given switching function, and the weight vector $[w_0,w_1,\ldots,w_7]$ is computed from the extended truth vector. In general, for an n-variable completely specified switching function, extended truth vector is a binary vector $[t_0,t_1,\ldots,t_{3^n-1}]$ with 3^n elements, and weight vector is an integer vector $[w_0,w_1,\ldots,w_{2^n-1}]$ with 2^n elements. Each element of the weight vector is associated with a polarity vector, which is shown at the rightmost side in Fig. 1. In general, for an n-variable switching function f, polarity vector for w_j is a binary vector $(b_1,b_2,\ldots,b_n)$ such that $b_1 b_2 \cdots b_n$ is the n-bit binary number representing j $(j = 0,1,\ldots,2^n-1)$, and w_j represents the number of products in the FPRM for f with polarity vector $(b_1,b_2,\ldots,b_n)$.

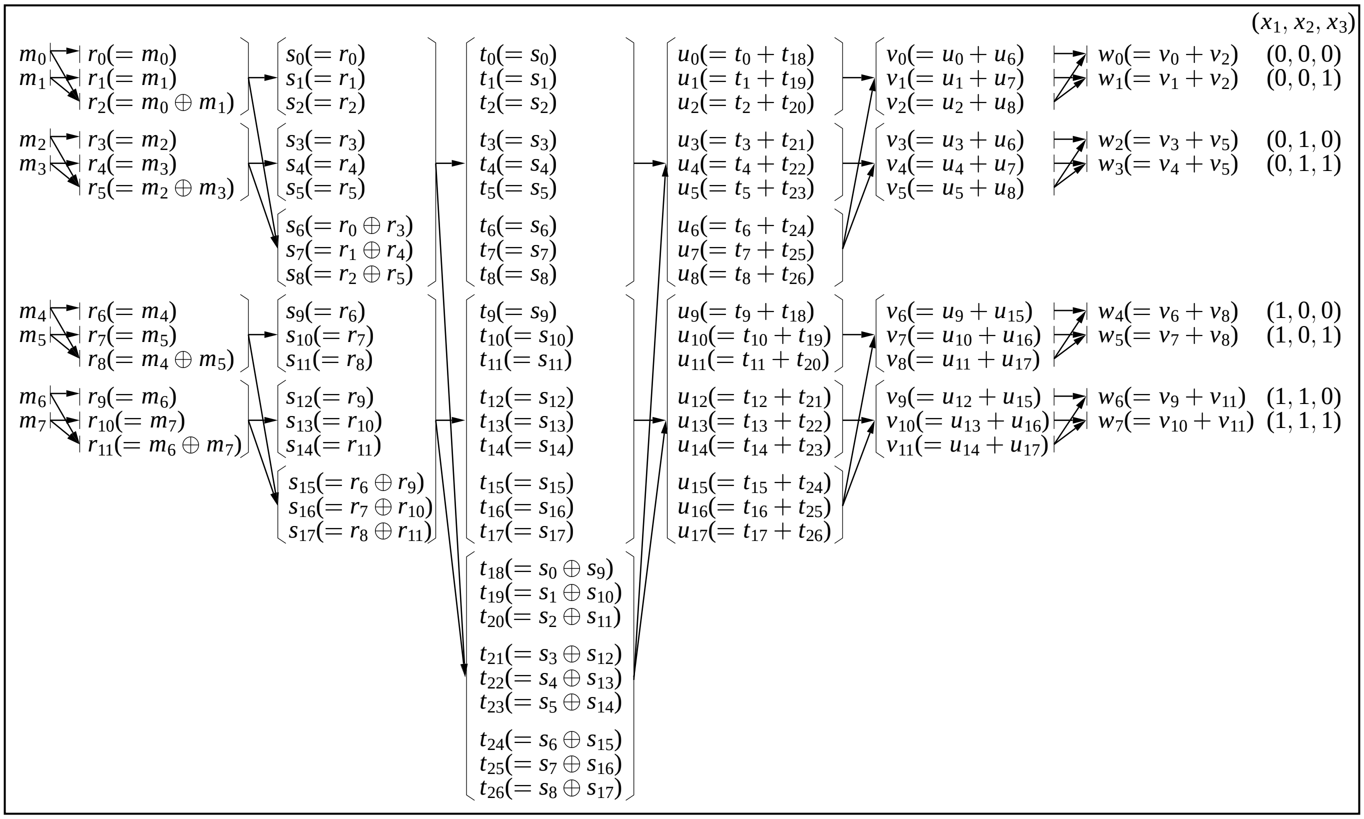

Figure 1: Computation of extended truth and weight vector for three-variable switching function.

For an n-variable switching function with α unspecified minterms $d_1,d_2,\ldots,d_\alpha$, extended truth vector is a vector of switching functions $t_i(d_1,d_2,\ldots,d_\alpha)$ $(i=0,1,\ldots,3^n-1)$, and weight vector is a vector of integer-valued functions $w_j(d_1,d_2,\ldots,d_\alpha)$ $(j=0,1,\ldots,2^n-1)$. For three-variable case, all the t_i's and w_j's can be obtained from Fig. 1. Extension to the functions with more variables is straightforward. Expressions for several t_i's and w_0 for three-variable function $[m_0,m_1,\ldots,m_7]$ are shown in the following:

$$w_0 = t_0+t_2+t_6+t_8+t_{18}+t_{20}+t_{24}+t_{26}, \qquad (3.1)$$

where

$$\begin{aligned} t_0 &= m_0,\\ t_2 &= m_0\oplus m_1,\\ t_6 &= m_0\oplus m_2,\\ t_8 &= m_0\oplus m_1\oplus m_2\oplus m_3,\\ t_{18} &= m_0\oplus m_4,\\ t_{20} &= m_0\oplus m_1\oplus m_4\oplus m_5,\\ t_{24} &= m_0\oplus m_2\oplus m_4\oplus m_6,\\ t_{26} &= m_0\oplus m_1\oplus m_2\oplus m_3\oplus m_4\oplus m_5\oplus m_6\oplus m_7. \end{aligned} \qquad (3.2)$$

Definition 3.1 Let the ***minimum value*** of the α-variable integer-valued function $w(d_1,d_2,\ldots,d_\alpha)$, denoted by w^{min}, be $\min_{0\le i\le 2^\alpha-1} m_i$, where $[m_0,m_1,\ldots,m_{2^\alpha-1}]$ represents the truth vector for w.

Let $[w_0,w_1,\ldots,w_{2^n-1}]$ be the weight vector for an n-variable incompletely specified switching function $f(x_1,x_2,\ldots,x_n)$, and w_j^{min} be the minimum value for $w_j(d_1,d_2,\ldots,d_\alpha)$ where $d_1,d_2,\ldots,d_\alpha$ represent unspecified minterms of f. Let $0\le k\le 2^n-1$ and $a_1,a_2,\ldots,a_\alpha\in\{0,1\}$ such that $w_k(a_1,a_2,\ldots,a_\alpha)=\min_{0\le j\le 2^n-1} w_j^{min}$. Let $c_1c_2\cdots c_n$ be the n-bit binary number representing k. Then, $(a_1,a_2,\ldots,a_\alpha)$ represents an assignment of $(d_1,d_2,\ldots,d_\alpha)$ and $(c_1,c_2,\ldots,c_n)$ represents a polarity vector that produce a minimum FPRM for f.

Example 3.1 Consider a three-variable switching function $f(x_1,x_2,x_3)$ whose truth vector $[m_0,m_1,m_2,m_3,m_4,m_5,m_6,m_7]=[d_0,0,d_2,d_3,1,1,1,0]$, where d_0, d_2, and d_3 are unspecified minterms. By putting the value of m_i $(i=0,1,\ldots,7)$ in (3.2), we have

$$\begin{aligned} t_0 &= d_0, & t_{18} &= 1\oplus d_0,\\ t_2 &= d_0, & t_{20} &= d_0,\\ t_6 &= d_0\oplus d_2, & t_{24} &= d_0\oplus d_2,\\ t_8 &= d_0\oplus d_2\oplus d_3, & t_{26} &= 1\oplus d_0\oplus d_2\oplus d_3. \end{aligned} \qquad (3.3)$$

By using Property 2.1 to (3.3), we have $t_{18}+t_{20}=1$ and $t_8+t_{26}=1$. Also, by using Property 2.2 to (3.3), we have $t_6+t_{24}=2(d_0\oplus d_2)$ and $t_0+t_2=2d_0$. Thus, from (3.1) and (3.3), we obtain

$$w_0 = 2+2d_0+2(d_0\oplus d_2). \qquad (3.4)$$

Equation (3.4) shows that w_0 cannot be less than 2 and it is independent of d_3. By inspection, we have $w_0 = 2$, when $d_0 = d_2 = 0$; however $w_0 = 6$, when $d_0 = 1$ and $d_2 = 0$. Thus, the minimum value for w_0 is 2. Similarly, we can obtain minimum value for w_i, when $i = 1,2,\ldots,7$.

To manipulate integer-valued function we use multi-terminal binary decision diagram (MTBDD) [5]. An MTBDD, which is a natural extension of binary decision diagram (BDD) [2], is a directed acyclic graph with multiple terminal nodes each of which has an integer value. Arithmetic operations, such as addition and multiplication, between integer-valued functions can be efficiently performed by using MTBDDs. It should be noted that switching functions are a subset of integer-valued functions and an MTBDD for a switching function is a BDD. We use MTBDD data structure to perform Boolean operations between switching functions.

A straightforward method to build MTBDDs for weight vector requires excessive computation time and memory resources, because they represent all possible FPRMs for the given incompletely specified function. However, we are only interested in an FPRM with the fewest products. Suppose we have an FPRM for the given function with $t_{threshold} + 1$ products, then it is sufficient to search for an FPRM with $t_{threshold}$ or fewer products. If such an FPRM does not exist then the FPRM with $t_{threshold} + 1$ products is the minimum FPRM. Thus, to restrict the search space without sacrificing the minimality of the solution, we use threshold value, $t_{threshold}$, during construction of MTBDDs. The threshold value can be obtained by using any simplification program for FPRMs.

Based on the above discussions, we develop the following algorithm for exact minimization of FPRM for incompletely specified n-variable switching function f:

Algorithm 3.1 (Exact minimization)

1. Get the user supplied threshold value, $t_{threshold}$.
2. Prepare extended truth vector $[t_0,t_1,\ldots,t_{3^n-1}]$ for f. (Fig. 1 shows the computation method for extended truth vector for three-variable functions. Extension to the functions with more variables is straightforward. Each element of the extended truth vector is a switching function represented as an MTBDD.)
3. Let $[w_0,w_1,\ldots,w_{2^n-1}]$ be the weight vector for f. For $i = 0$ to $2^n - 1$, do the following:

   (a) Gather elements from the extended truth vector such that
   $$w_i = \sum_{t \in T_i} t, \qquad (3.5)$$
   where $T_i \subset \{t_0,t_1,\ldots,t_{3^n-1}\}$. (Fig. 1 shows how to gather elements corresponding to w_i from the extended truth vector for three-variable function. Extension to the functions with more variables is straightforward. It is obvious from Fig. 1 that the number of elements in T_i is 2^n.)

   (b) Apply Properties 2.1 and 2.2 to (3.5). Thus, we have
   $$w_i = a + \sum_{0 \le j \le L-1} b_j u_j,$$
   where $a \ge 0$, $b_j \ge 1$, $u_j \in T_i$, and $L \le 2^n$.

   (c) Construct an MTBDD for w_i. During construction (i) if any terminal value of an intermediate MTBDD is greater than $t_{threshold}$, set that terminal value to ∞; (ii) if an intermediate MTBDD represent constant ∞, stop the construction and assign MTBDD for w_i to ∞.

   (d) Obtain minimum value w_i^{min} from the MTBDD for w_i. (This corresponds to finding a path to a terminal node of the MTBDD that produces a minimum value for w_i.)

   (e) If $w_i^{min} <= t_{threshold}$: (i) save the polarity vector and an assignment of the unspec-

```
procedure Construct_MTBDD(a,S,D_all) {
    D_remain, D_this: set of support variables;
    S_save, S_this: subset of S;
    c_save, c_this: integer (counter);
    w_i ← a;
    until S ≠ ∅ do {
        D_remain ← D_all − support(w_i);
        c_save ← 0;
        for each d_remain ∈ D_remain do {
            D_this ← {d_remain} ∪ support(w_i);
            c_this ← 0; S_this ← ∅;
            for each b_j u_j ∈ S do {
                if support(u_j) ⊆ D_this then {
                    c_this ← c_this + b_j;
                    S_this ← S_this ∪ {b_j u_j};
                }
            }
            if c_this > c_save then {
                c_save ← c_this;
                S_save ← S_this;
            }
        }
        if S_save = ∅ then
            S_save ← an element of S;
        for each b_j u_j ∈ S_save do
            w_i ← w_i + b_j u_j; /* MTBDD operation */
        S ← S − S_save;
    }
    return w_i; /* MTBDD */
}
```

Figure 2: Pseudocode Construct_MTBDD.

ified minterms corresponding to w_i^{min}; (ii) $t_{threshold} \leftarrow w_i^{min} - 1$.

4. If any polarity vector is saved in step 3(e) then obtain an FPRM by using the most recently saved polarity vector and assignment of the unspecified minterms, otherwise report "No solution exists with $t_{threshold}$ or fewer products."

To build an MTBDD for w_i at step 3(c) we must do arithmetic addition of a set of MTBDDs. The MTBDDs can be arranged in numerous ways to perform addition. The arrangement influences the computation time and the size of the intermediate MTBDDs during addition. A naive arrangement requires excessive memory resources and long computation time. To build MTBDD for w_i we use procedure Construct_MTBDD(a,S,D_{all}), the pseudocode of which is shown in Fig. 2, where S represents $\{b_0u_0, b_1u_1, \ldots, b_{L-1}u_{L-1}\}$, D_{all} represents $\{d_1, d_2, \ldots, d_\alpha\}$, and support($w_i$) represents the set of variables on which w_i depends. The procedure first chose an MTBDD that depends on the fewest variables and then arranged other MTBDDs to slowly increase the number of variables in the intermediate MTBDDs.

The minimization method presented in this section can be adapted to obtain exact minimum Kronecker expressions [1, 6, 15] for incompletely specified switching functions. In this case we must compute extended weight vector [1, 6] instead of weight vector.

IV. Experimental Results

We implemented Algorithm 3.1 in C by using CUDD package [19] and conducted experiments by using a set of switching functions. The detail experimental results are shown in Table 1. The factors on which the computation time mainly depends are the threshold value, the number of variables in the function, and the number of unspecified minterms.

The current implementation works favorably for many functions with eight or fewer variables and with any number of unspecified minterms. However, for functions with nine or more variables it often requires excessive CPU time and memory resources when the number of unspecified minterms is more than 30.

A comparison with the other algorithms is difficult, because none of them explicitly reported the benchmark functions.

Table 1: Experimental results.

f(n,t,d,s)*	Prod†	Thre‡	Peak§	Time¶
f(6,15,30,25)	9	10	1732	0.33
f(6,12,40,50)	6	10	11133	1.06
f(7,35,50,5)	21	25	41733	10.57
f(7,20,80,5)	10	15	70379	24.33
f(7,20,90,5)	8	12	134166	31.30
f(8,25,200,50)	12	12	1075438	3043.23
f(8,100,80,10)	51	51	5638312	5208.40
f(8,35,180,10)	15	15	4931098	7521.47
f(8,60,160,5)	21	22	9874309	10811.68
f(8,80,100,50)	41	45	11554627	26617.01
		41	11554627	24978.05
f(9,250,50,5)	167	170	2072822	3706.12
f(9,15,480,80)	6	7	8192276	29666.09
f(10,500,40,25)	397	420	4283146	16147.14
		397	2227298	5072.87
f(12,2000,30,25)	1874	1880	455409	3685.07
f(14,8000,30,50)	7836	7850	855023	7012.62

*f(n,t,d,s): Function (defined in Appendix).
†Prod: Number of products in minimum FPRM.
‡Thre: Threshold value.
§Peak: Peak number of live MTBDD nodes.
¶Time: CPU seconds on SUN ULTRA 10.

V. Conclusions and Comments

Exact minimization of FPRMs for incompletely specified switching functions is a computationally hard problem, because the search space increases exponentially with the increase in the unspecified minterms. Although Algorithm 3.1 requires only one MTBDD at a time, the present implementation of it uses a fixed variable order for all the MTBDDs. Currently, we are working to find a better variable order for individual MTBDDs. This strategy would be useful to solve larger problems with less memory resources.

Acknowledgement

This work was supported in part by a Postdoctoral Fellowship of the Japan Society for the Promotion of Science and in part by a Grant-in-Aid for the Scientific Research of the Ministry of Education, Science, Culture, and Sports of Japan. The authors thank anonymous referees for giving constructive suggestions.

References

[1] G. Bioul, M. Davio, and J.-P. Deschamps, "Minimization of ring-sum expansions of Boolean functions," Philips Res. Repts., Vol. 28, pp. 17–36, 1973.

[2] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE Trans. Comput., Vol. C-35, No. 8, pp. 677–691, Aug. 1986.

[3] C.-H. Chang and B. J. Falkowski, "Flexible optimization of fixed polarity Reed-Muller expansions for multiple output completely and incompletely specified Boolean functions," Proc. Asia and South Pacific Design Automation Conference, pp. 335–340, Sept. 1995.

[4] C.-H. Chang and B. J. Falkowski, "Adaptive exact optimisation of minimally testable FPRM expansions," IEE Proceedings-Computers and Digital Techniques, Vol. 145, No. 6, pp. 385–394, Nov. 1998.

[5] E. M. Clarke, K. L. McMillan, X. Zhao, M. Fujita, and J. Yang, "Spectral transforms for large Boolean functions with applications to technology mapping," Proc. Design Automation Conference, pp. 54–60, June 1993.

[6] M. Davio, J.-P. Deschamps, and A. Thayse, Discrete and Switching Functions, McGraw-Hill International, 1978.

[7] R. Drechsler, M. Theobald, and B. Becker, "Fast OFDD-based minimization of fixed polarity Reed-Muller expressions," IEEE Trans. Comput., Vol. C-45, No. 11, pp. 1294–1299, Nov. 1996.

[8] D. H. Green, "Reed-Muller expansions of incompletely specified functions," IEE Proceedings, Vol. 134, Pt. E, No. 5, pp. 228–236, Sept. 1987.

[9] S. L. Hurst, D. M. Miller, and J. C. Muzio, Spectral Techniques in Digital Logic, Academic Press Inc., 1985.

[10] U. Kebschull and W. Rosenstiel, "Efficient graph-based computation and manipulation of functional decision diagrams," Proc. European Conference on Design Automation, pp. 278–282, Mar. 1993.

[11] B. W. Kernighan and D. M. Ritchie, The C Programming Language, Second Edition, Prentice-Hall, 1988.

[12] P. K. Lui and J. C. Muzio, "Boolean matrix transforms for the minimization of modulo-2 canonical expressions," IEEE Trans. Comput., Vol. C-41, No. 3, pp. 342–347, Mar. 1992.

[13] L. McKenzie, A. E. A. Almaini, J. F. Miller, and P. Thomson, "Optimisation of Reed-Muller logic functions," International Journal of Electronics, Vol. 75, No. 3, pp. 451–466, Sept. 1993.

[14] A. Sarabi and M. A. Perkowski, "Fast exact and quasi-minimal minimization of highly testable fixed polarity AND/XOR canonical networks," Proc. Design Automation Conference, pp. 30–35, June 1992.

[15] T. Sasao, "AND-EXOR expressions and their optimization," in T. Sasao, ed., Logic Synthesis and Optimization, Kluwer Academic Publishers, 1993.

[16] T. Sasao, "Representations of logic functions using EXOR operators," in T. Sasao and M. Fujita, eds., Representations of Discrete Functions, Kluwer Academic Publishers, 1996.

[17] T. Sasao and F. Izuhara, "Exact minimization of FPRMs using multi-terminal EXOR TDDs," in T. Sasao and M. Fujita, eds., Representations of Discrete Functions, Kluwer Academic Publishers, 1996.

[18] T. Sasao, Switching Theory for Logic Synthesis, Kluwer Academic Publishers, 1999.

[19] F. Somenzi, CUDD: CU Decision Diagram Package, Release 2.3.0, University of Colorado at Boulder, 1998 (http://vlsi.colorado.edu/~fabio/).

[20] A. Tran, "Graphical method for the conversion of minterms to Reed-Muller coefficients and the minimization of exclusive-OR switching functions," IEE Proceedings, Vol. 134, Pt. E, No. 2, pp. 93–99, Mar. 1987.

[21] C. Tsai and M. Marek-Sadowska, "Minimisation of fixed-polarity AND/XOR canonical networks," IEE Proceedings-Computers and Digital Techniques, Vol. 141, No. 6, pp. 369–374, Nov. 1994.

[22] C. Tsai and M. Marek-Sadowska, "Generalized Reed-Muller forms as a tool to detect symmetries," IEEE Trans. Comput., Vol. 45, No. 1, pp. 33–40, Jan. 1996.

[23] C. Tsai and M. Marek-Sadowska, "Multilevel logic synthesis for arithmetic functions," Proc. 33rd Design Automation Conference, pp. 242–247, June 1996.

[24] C. Tsai and M. Marek-Sadowska, "Boolean functions classification via fixed polarity Reed-Muller forms," IEEE Trans. Comput., Vol. C-46, No. 2, pp. 173–186, Feb. 1997.

[25] D. Varma and E. A. Trachtenberg, "Computation of Reed-Muller expansions of incompletely specified Boolean functions from reduced representations," IEE Proceedings-E, Vol. 138, No. 2, pp. 85–92, Mar. 1991.

[26] A. Zakrevskij, "Minimizing polynomial implementation of weakly specified logic functions and systems," Proc. 3rd International Workshop on Applications of the Reed-Muller Expansion in Circuit Design, pp. 157–166, Sept. 1997.

[27] I. I. Zhegalkin, "The technique of calculation of statements in symbolic logic," Mathe. Sbornik, Vol. 34, pp. 9–28, 1927 (in Russian).

[28] Z. Zilic and Z. G. Vranesic, "A multiple-valued Reed-Muller transform for incompletely specified functions," IEEE Trans. Comput., Vol. C-44, No. 8, pp. 1012–1020, Aug. 1995.

Appendix

To generate the truth vector of the test instances shown in Table 1 we used the following C code, the first two lines of which have been adapted from [11, p. 46]:

```
#define E(m) s = s * 1103515245 + 12345, \
    r = (unsigned int) (s >> 16) % w, m > 0
char *f(int n, int t, int d, unsigned long s)
{
    int r, w = 1 << n;
    char *v = (char *) calloc(w, 1);
    while (E(t)) if (!v[r]) v[r] = 1, t--;
    while (E(d)) if (!v[r]) v[r] = 2, d--;
    return v;
}
```

The function f(n,t,d,s) returns a pointer v for the truth vector $[\mathtt{v}[0],\mathtt{v}[1],\ldots,\mathtt{v}[2^n-1]]$ — where $\mathtt{v}[i]$ $(0 \le i \le 2^n-1)$ represents a true, false, or incompletely specified minterm if $\mathtt{v}[i]$ is 1, 0, or 2, respectively — for an n-variable switching function with t true and d incompletely specified minterms.