2000-11

Comments on Sympathy: Fast exact minimization of fixed polarity Reed-Muller expansion for symmetric functions

Dueck, Gerhard W.

Comments on "Sym*pathy*: Fast Exact Minimization of Fixed Polarity Reed–Muller Expansion for Symmetric Functions"

Jon T. Butler, Gerhard W. Dueck, Vlad P. Shmerko, and Svetlana Yanuskevich

***Abstract*—The above paper[1] finds an optimal fixed-polarity Reed–Muller expansion of an n-variable totally symmetric function using an OFDD-based algorithm that requires $O(n^7)$ time and $O(n^6)$ storage space. However, an algorithm based on Suprun's transeunt triangles [1], [3], [4] requires only $O(n^3)$ time and $O(n^2)$ storage space. An implementation of this algorithm yields computation times lower by several orders of magnitude.**

***Index Terms*—FPRM (fixed polarity Reed–Muller expressions), two-level AND/EXOR forms, symmetric functions, logic synthesis, minimization.**

I. Introduction

A recent program, *Sympathy*,[1] for finding optimal polarity Reed–Muller (FPRM) expansions of symmetric functions is based on an algorithm whose data structure is an OFDD of the given function. It requires $O(n^7)$ operations and $O(n^6)$ storage space, where n is the number of variables. However, if one uses a more efficient data structure, specifically the transeunt triangle of Suprun [1], [3], [4], the same computation can be done with $O(n^3)$ operations and $O(n^2)$ storage space. The improvement is achieved because coefficients needed in various expansions are computed and stored only once, whereas Sym*pathy* builds a new OFDD for each polarity. On benchmark functions, the speed improvement is by orders of magnitude

II. Notation

A FPRM expansion for a general function $f(x_1, x_2, \ldots, x_n)$ is

$$f(x_1, x_2, \ldots, x_n) = c_0 \oplus c_1 x_1^* \oplus c_1 x_2^* \oplus \cdots \oplus c_n x_n^* \oplus c_{n+1} x_1^* x_2^* \oplus \cdots \oplus c_{2^n-1} x_1^* x_2^* \cdots x_n^* \quad (1)$$

where x_i^* is either x_i or $\overline{x}_i$ everywhere. The term *fixed-polarity* refers to the fact that each variable occurs in the expression in only one way, x_i or $\overline{x}_i$. For example, $f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2x_3$ has the following four FPRM expansions.

No variables complemented: $1 \oplus [x_1 \oplus x_2 \oplus x_3] \oplus [x_1x_2 \oplus x_1x_3 \oplus x_2x_3]$

One variable complemented: $\overline{x}_1 \oplus \overline{x}_1x_2 \oplus \overline{x}_1x_3 \oplus x_2x_3$

Two variables complemented: $x_3 \oplus \overline{x}_1\overline{x}_2 \oplus \overline{x}_1x_3 \oplus \overline{x}_2x_3$

All variables complemented: $1 \oplus [\overline{x}_1 \oplus \overline{x}_2 \oplus \overline{x}_3] \oplus [\overline{x}_1\overline{x}_2 \oplus \overline{x}_1\overline{x}_3 \oplus \overline{x}_2\overline{x}_3]$

Note the total number of product terms required to realize this function. In the first and fourth FPRM expansions, seven terms are required, while in the second and third, only four are required. The *FPRM simplification problem* is to determine which of $n+1$ polarities (number of complemented variables) yields the FPRM expansion with the fewest terms. In this example, the two middle polarities are both optimum, yielding an expansion of four terms each.

A function $f(x_1, x_2, \cdots x_n)$ is (*totally*) *symmetric* if and only if it is unchanged by any permutation of variables. For example, $f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2x_2$ is symmetric. Certain coefficients in the FPRM expansion of a symmetric function are identical. Let the *Reed–Muller expansion matrix* of a symmetric function be an $(n+1) \times (n+1)$ matrix of binary coefficients

$$RM_i = \begin{bmatrix} d_{00} & d_{01} \cdots d_{0n} \\ d_{10} & d_{11} \cdots d_{1n} \\ \vdots & \vdots \\ d_{n0} & d_{n1} \cdots d_{nn} \end{bmatrix} \quad (2)$$

where d_{jk} is the coefficient of a product term of x_i's in an FPRM expansion (1) in which j variables are complemented and k are not. For the four FPRM expansions of $f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2x_3$, we have

$$RM_0 = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad RM_1 = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$RM_2 = \begin{bmatrix} \mathbf{1} & \mathbf{0} & 0 & 0 \\ \mathbf{1} & \mathbf{0} & 0 & 0 \\ \mathbf{0} & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad RM_3 = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 \\ \mathbf{0} & 0 & 0 & 0 \end{bmatrix}.$$

The bold values represent coefficients in the corresponding FPRM expansion. The 0s not in bold are 0 in *all* Reed–Muller expansion matrices for the *same* polarity.

A symmetric function is completely specified by a *carry vector* of logic values $A = [a_0, a_1, \ldots, a_n]$, such that $f(x_1, x_2, \ldots, x_n)$ is a_i for all assignments of values to $x_1, x_2, \ldots, x_n$ that have i 1s, where $0 \le i \le n$. For example, the carry vector of $f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2x_3$ is $[1, 0, 0, 1]$.

III. Transeunt Triangle Representation of Reed–Muller Expansions

Consider a triangle of 0s and 1s, where the base is a symmetric function's $n+1$-bit carry vector. Immediately below this is a vector of n 1s and 0s formed by the exclusive OR of adjacent bits in the carry vector. Immediately below this is a vector of $n-1$ 1s and 0s formed by the exclusive OR of adjacent bits in the previous vector, etc.. At the bottom is a single 1 or 0. Doing this for $f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2x_3$ yields the triangle in Fig. 1.

```
1  0  0  1
 1  0  1
  1  1
   0
```

Fig. 1. The transeunt triangle for $f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2x_3$.

The resulting triangle is the *transeunt triangle*, originated by Suprun [3], [4]. Notice that the bits along the triangle's left side are coefficients in RM_0, while bits along the right side are coefficients in RM_3. Additionally, embedded rectangles represent the coefficients in RM_1 and RM_2. This can be seen in Fig. 2.

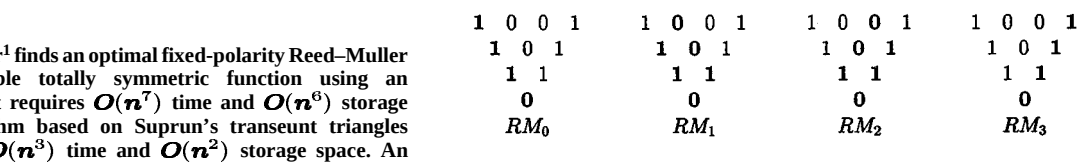

Fig. 2. Reed–Muller expansion matrices embedded in the transeunt triangle of $f(x_1, x_2, x_3) = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2x_3$.

Manuscript received February 11, 2000. This paper was recommended by Associate Editor M. Papaefthymiou.

J. T. Butler is with the Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA 93943-5121 USA.

G. W. Dueck is with the Department of Computer Science, University of New Brunswick, Fredericton, N.B. E3B 5A3 Canada.

V. P. Shmerko and S. Yanuskevich are with the Insitute of Computer Science and Inf. Science, Technical University of Szczecin, 71210 Szczecin, Poland.

Publisher Item Identifier S 0278-0070(00)10293-3.

[1]R. Drechsler and B. Becker, "**Sym***pathy*: Fast exact minimization of fixed polarity Reed–Muller expansion for symmetric functions," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 1–5, Jan 1997.

TABLE I
EXECUTION TIMES (IN SECS.) FOR SYMMETRIC BENCHMARK FUNCTIONS*

Function	In	Out #	Car. Vec.	Opt. Pol.	Products	FDD	Sym*pathy*	Symphony
co14	14	0	010^{13}	14	14	368.1	0.3	0.000118
m1	6	6	10^{6}	6	1	1.0	0.1	0.000016
m4	6†	8	10^{6}	6	1	4.2	0.1	0.000016
misex2	5†	5	$0^{5}1$	0	1	‡	0.4	0.000011
misj	10†	10	$1^{10}0$	0	2	‡	0.6	0.000050
xor5	5	0	$(01)^{3}$	0,2,4	5	0.5	0.1	0.000011
rd53	5	0	$0^{4}1^{2}$	0	5	-	-	0.000012
rd53	5	1	$0^{2}1^{2}0^{2}$	0,5	10	-	-	0.000011
rd53	5	2	$(01)^{3}$	0,2,4	5	-	-	0.000011
rd73	7	0	$0^{4}1^{4}$	0	35	-	-	0.000022
rd73	7	1	$(0^{2}1^{2})^{2}$	0	21	-	-	0.000023
rd73	7	2	$(01)^{4}$	0,2,4,6	7	-	-	0.000022
rd84§	8	0	$(0^{2}1^{2})^{2}0$	0	28	4.7	0.1	0.000029
rd84§	8	1	$(01)^{4}0$	0,2,4,6,8	8	-	-	0.000030
rd84§	8	2	$0^{8}1$	0	1	-	-	0.000030
rd84§	8	3	$0^{4}1^{4}0^{2}$	0	70	5.3	0.1	0.000029
sym4	4	0	$01^{2}0^{2}$	4	6	-	-	0.000008
sym6	6	0	$0^{2}1^{3}0^{2}$	0,6	36	-	-	0.000016
sym9	9	0	$0^{3}1^{4}0^{3}$	4,5	173	11.7	0.1	0.000039
sym10	10	0	$0^{4}1^{5}0^{2}$	0	266	27.6	0.2	0.000050
sym12	12	0	$0^{4}1^{5}0^{4}$	0,12	1288	-	-	0.000079
sym15	15	0	$0^{5}1^{6}0^{5}$	7,8	15139	-	-	0.000139
dbruijn_2	4	0	$0^{2}1^{2}0$	0	6	-	-	0.000008
dbruijn_3	9	0	$0^{3}101^{3}0^{2}$	7	256	-	-	0.000039
dbruijn_4	18	0	$0^{4}10^{2}1^{2}0101^{4}0^{3}$	17	106,284	-	-	0.000297
dbruijn_5	35	0	$0^{5}10^{3}1^{2}0^{2}1010^{2}1^{3}0101^{2}01^{5}0^{4}$	6	15,215,790,080	-	-	0.002739

IV. THE ALGORITHM AND ITS TIME AND SPACE COMPLEXITY

A. The Algorithm

Note that a single element of the transeunt triangle represents one or more coefficients in the various Reed–Muller expansion matrices. The efficiency of the transeunt triangle is due to the fact that it is not necessary to recompute this coefficient for each polarity.

```
Algorithm 1 [4]
1) Generate the transeunt triangle.
2) For each RM_i, extract the coefficients
   (d_jk), and compute the number of product
   terms.
3) Choose an RM_i with the fewest product
   terms.
```

B. Time and Space Complexity

The following lemma gives both the time and space complexity of the above algorithm. The time complexity is due to [4].

Lemma 4.1: Algorithm 1 is an $O(n^3)$-time algorithm that requires $O(n^2)$ storage space for computing the optimal fixed-polarity Reed–Muller expansion of a symmetric function on n variables.

Proof: In applying the algorithm, O(n^2) storage locations are required for the coefficients in the triangles. O(n) locations are required to store the number of product terms, one for each of the $n+1$ polarities, for a total of O(n^2) locations. ■

The OFDD approach has time complexity $O(n^7)$ and space complexity $O(n^6)$. Thus, Algorithm 1 represents a significant improvement.

V. EXPERIMENTAL RESULTS

Suprun [3], [4] did not apply his algorithm to benchmark functions. Our implementation is called **Symphony**, (**sy**mmetric **ph**unction **o**ptimizi**n**g s**y**stem), which is written in C++ and compiled under Microsoft's Visual Studio Version 6.0 for Windows98. It was run on a 400 MHz. Pentium system.

A. Comparison of Symphony on benchmark functions

Table I shows, for certain symmetric benchmark functions, the execution time of **Symphony** compared to Sym*pathy* and to FDD, another OFDD-based minimizer that does not consider symmetry [2]. Table I also shows the number of inputs (In), the Output Number (Out), the Carrier Vector expressed as a regular expression (Car. Vec.), the polarity(ies) that produced the optimal realization (Opt. Pol.), and the number of product terms in the optimal solution (Products). The three execution times (FDD, Sym*pathy*, and **Symphony**) are shown in seconds.

As can be seen, **Symphony** is very fast, requiring no more than 0.0002 secs. on any of the functions considered by Dreschler and Becker. Indeed, these execution times are less than the time interval between real time clock interrupts. As a result, timing functions in C++ return zero elapsed time for program execution. To achieve the necessary resolution, each function was minimized 2 000 000 times and the total time was divided by 2 000 000.

Each dbruijn_k entry in Table I is a d'Brujin sequence indexed by k. That is, each sequence contains exactly one copy of each of the 2^k binary k-tuples. Overall, it contains a total of $2^k + k - 1$ bits. This sequence is such that decision diagram representations for such functions will have many nodes, as there are few repeated subsequences. As a result, algorithms based on decision diagrams will require more computation time than for other symmetric functons.

Table II shows, for certain symmetric functions that are also threshold functions, the relative execution times of FDD, Sym*pathy*, and **Symphony**. Again, **Symphony** is fast.

VI. CONCLUSION

Rather than computing the entire FPRM expansion for each polarity, **Symphony** computes and stores expansion coefficients only once,

TABLE II
EXECUTION TIMES (IN SECS.) FOR SYMMETRIC THRESHOLD FUNCTIONS

Function	In	Out #	Car. Vec.	Opt. Pol.	Products	FDD	Sym*pathy*	Symphony
$thres_{9,10}$	10	0	$0^9 1^2$	0	11	21.0	0.1	0.000090
$thres_{10,11}$	11	0	$0^{10} 1^2$	0	11	42.1	0.2	0.000114
$thres_{11,12}$	12	0	$0^{11} 1^2$	0	13	87.3	0.2	0.000143
$thres_{12,13}$	13	0	$0^{12} 1^2$	0	13	177.5	0.2	0.000171
$thres_{13,14}$	14	0	$0^{13} 1^2$	0	15	364.2	0.3	0.000209
$thres_{14,15}$	15	0	$0^{14} 1^2$	0	15	756.5	0.3	0.000251
$thres_{15,16}$	16	0	$0^{15} 1^2$	0	17	1555.8	0.3	0.000295
$thres_{16,17}$	17	0	$0^{16} 1^2$	0	17	3140.3	0.4	0.000346
$thres_{17,18}$	18	0	$0^{17} 1^2$	0	19	6483.9	0.4	0.000406
$thres_{18,19}$	19	0	$0^{18} 1^2$	0	19	13033.1	0.4	0.000468
$thres_{19,20}$	20	0	$0^{19} 1^2$	0	21	25870.1	0.4	0.000540
$thres_{20,21}$	21	0	$0^{20} 1^2$	0	21	52549.4	0.5	0.000612
$thres_{21,22}$	22	0	$0^{21} 1^2$	0	23	‡	0.5	0.000691
$thres_{22,23}$	23	0	$0^{22} 1^2$	0	23	‡	0.5	0.000781
$thres_{23,24}$	24	0	$0^{23} 1^2$	0	25	‡	0.6	0.000883
$thres_{24,25}$	25	0	$0^{24} 1^2$	0	25	‡	0.6	0.000996
$thres_{25,26}$	26	0	$0^{25} 1^2$	0	27	‡	0.7	0.001109
$thres_{26,27}$	27	0	$0^{26} 1^2$	0	27	‡	0.7	0.001230
$thres_{27,28}$	28	0	$0^{27} 1^2$	0	29	‡	0.7	0.001343
$thres_{28,29}$	29	0	$0^{28} 1^2$	0	29	‡	0.8	0.001492
$thres_{29,30}$	30	0	$0^{29} 1^2$	0	31	‡	0.8	0.001658

using the transeunt triangle, and extracts them, as needed, to form the various expansions. In this way, it achieves a major savings in computation time and storage over Sym*pathy*, which computes a decision diagram for each polarity.

An abbreviated version of **Symphony** can be accessed at http://www.oc.nps.navy.mil/~butler/transeunt.html (word length restrictions on the server preclude carrier vectors with more than 31 bits). Users can input a carrier vector and see the transeunt triangle along with the number of product terms for each polarity.

ACKNOWLEDGMENT

The authors gratefully acknowledge the comments by the editors and three referees which led to improvements.

REFERENCES

[1] J. T. Butler, G. W. Dueck, S. N. Yanushkevich, and V. P. Schmerko, "On the number of generators for transeunt triangles," Discrete Appl. Math., 2000, to be published.

[2] R. Drechsler, M. Theobald, and B. Becker, "Fast OFDD based minimization of fixed-polarity Reed–Muller expressions," in *Proc. Eur. Design Automation Conf.*, 1994, pp. 2–7.

[3] V. P. Suprun, "Polynomial expression of symmetric Boolean functions" (in Russian), *Izvestija AN USSR. Techn. Kibernetika*, no. 4, pp. 123–127, 1985.

[4] ——, "Fixed polarity Reed–Muller expressions of symmetric Boolean functions," in *Proc. IFIP WG 10.5 Workshop on Application of the Reed–Muller Expansions in Circuit Design*, 1995, pp. 246–249.