Brief Contributions

Pseudo-Kronecker Expressions for Symmetric Functions

Rolf Drechsler, *Member*, *IEEE*

Abstract—Pseudo-Kronecker Expressions (PSDKROs) are a class of AND/EXOR expressions. In this paper, it is proven that exact minimization of PSDKROs for totally symmetric functions can be performed in polynomial time. A new implementation method for PSDKROs is presented. Experimental results are given to show the efficiency of the presented approach in comparison to previously published work on AND/EXOR minimization.

Index Terms—Logic synthesis, AND/EXOR, PSDKRO, 2-level minimization, BDD.

1 Introduction

THE use of EXOR gates in the synthesis process reduces the hardware costs in many cases [16], [15]. Additionally, EXOR-based circuits often have nice testability properties [8], [11], [12], [4].

In contrast to AND/OR minimization—that, in the meantime, is well-understood—in AND/EXOR minimization, several restricted classes are considered, like *Fixed Polarity Reed-Muller Expression* (FPRM) [9] and *Kronecker Expression* (KRO) [5]. (For an excellent overview, see [13].) These subclasses are of interest since the minimization of general *Exclusive Sum of Product Expressions* (ESOPs) turned out to be computationally very hard, i.e., all programs presented so far have long runtimes and often fail to determine the optimal result (see, e.g., [14], [7]).

As one alternative, *Pseudo-Kronecker Expressions* (PSDKROs) [5], [13] have been proposed since they are an interesting compromise: The resulting 2-level forms are of moderate size, i.e., close to ESOPs, and, additionally, the minimization process can be handled within reasonable time bounds.

One subclass of Boolean functions has gained great interest—namely symmetric functions. For totally symmetric functions, it can be proven that ESOPs *always* allow smaller or equal representation with respect to number of terms than *Sum of Product Expressions* (SOP) [10]. For the more restricted classes of FPRMs and KROs, exact minimization of totally symmetric functions can be carried out in polynomial time and space [6], [1].

In this paper, it is shown that a similar result can also be proven for PSDKROs. An *Ordered Binary Decision Diagram* (OBDD) [3] based implementation is proposed that can easily be coded and performs very fast. Experimental results are reported that demonstrate the efficiency of this approach. Totally symmetric functions with more than 20 variables are exactly minimized in less than a hundredth of a second on a *Sun Sparc 4* workstation. A comparison to other AND/EXOR classes for some totally symmetric functions and, also, some functions without symmetries is given. Our algorithm is orders of magnitude faster than approaches based on term enumeration.

The paper is structured as follows: In Section 2, PSDKROs are defined. A brief introduction to symmetric functions is given in Section 3. The polynomial minimization algorithm is presented in Section 4. The implementation of the algorithm using OBDDs is discussed in Section 5. In Section 6, experimental results are given. We finish with a resume of the results in Section 7.

• *The author is with the Institute of Computer Science, Albert-Ludwigs-University, 79110 Freiburg im Breisgau, Germany.*
E-mail: drechsler@informatik.uni-freiburg.de.

Manuscript received 15 Oct. 1996; revised 4 Mar. 1999.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 102303.

```
psdkro (node v) {
    if (v == ZERO) return 0 ;
    if (v == ONE) return 1 ;

    if (v.val defined) return v.val ;

    v_l = cofactor_0 (v) ;
    v_h = cofactor_1 (v) ;
    v_ex = EXOR (v_l, v_h) ;

    x1 = psdkro (v_l) ;
    x2 = psdkro (v_h) ;
    x3 = psdkro (v_ex) ;

    v.val = x1 + x2 + x3 - max(x1,x2,x3) ;

    return v.val ;
}
```

Fig. 1. Sketch of the algorithm.

2 Pseudo-Kronecker Expressions

In this section, we briefly review the essential definitions of *Pseudo-Kronecker Expressions* (PSDKROs). (For more details, see [5], [13].)

Let f_i^0 (f_i^1) denote the *cofactor* of f with respect to $x_i = 0$ ($x_i = 1$) and f_i^2 is defined as $f_i^2 := f_i^0 \oplus f_i^1$, $\oplus$ being the Exclusive OR operation. A Boolean function $\mathbf{B}^n \rightarrow \mathbf{B}$ can then be represented by one of the following formulae:

$$f = \overline{x}_i f_i^0 \oplus x_i f_i^1 \quad \textit{Shannon } (S) \tag{1}$$

$$f = f_i^0 \oplus x_i f_i^2 \quad \textit{positive Davio } (pD) \tag{2}$$

$$f = f_i^1 \oplus \overline{x}_i f_i^2 \quad \textit{negative Davio } (nD). \tag{3}$$

If we apply to a function f either S, pD, or nD, we get two subfunctions. To each subfunction again S, pD, or nD can be applied. This is done until constant functions are reached. If we multiply out the resulting expression, we get a 2-level AND/EXOR form, called a PSDKRO.

The decompositions are applied with respect to a fixed variable ordering. Notice that the choice of the variable ordering in which the decompositions are applied and the choice of the decomposition per subfunction largely influences the size of the resulting representation [13].

Example 1. Let $f(x_1, x_2, x_3) = x_1 + x_2 x_3$. If we first decompose f using S, we get:

0018-9340/99/$10.00 © 1999 IEEE
Authorized licensed use limited to: National Taiwan University. Downloaded on December 15,2020 at 10:51:46 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Single Output Totally Symmetric Functions

name	*in*	FPRM		KRO		ESOP		PSDKRO	
		terms	time	terms	time	terms	time	terms	time
border20	20	-	-	2	72.2	2	0.1	2	0.01
border25	25	-	-	-	-	2	0.1	2	0.01
co14	14	-	-	14	2.8	14	0.2	14	0.01
sym9	9	173	1851.6	173	0.9	52	32.7	90	0.01
sym10	10	-	-	266	1.2	82	88.6	134	0.01

$$f_1^0 = x_2x_3 \text{ and } f_1^1 = 1.$$

Then, we decompose f_1^0 using nD:

$$(f_1^0)^1 = x_3 \text{ and } (f_1^1)^2 = x_3.$$

Finally, we apply pD:

$$((f_1^0)^1)^0 = 0 \text{ and } ((f_1^1)^1)^2 = 1.$$

If we multiply out the expression we obtain:

$$\begin{aligned} f &= (x_3 \oplus \overline{x}_2x_3)\overline{x}_1 \oplus x_1 \\ &= x_1 \oplus \overline{x}_1\overline{x}_2x_3 \oplus \overline{x}_1x_3. \end{aligned}$$

3 Symmetric Functions

Let $f : \mathbf{B}^n \to \mathbf{B}$ be a totally defined Boolean function and $X_n = \{x_1, \ldots, x_n\}$ be the corresponding set of variables. The function f is said to be *symmetric* with respect to a set $S \subseteq X_n$ if f remains invariant under all permutations of the variables in S. For completely specified functions, the symmetry is an equivalence relation which partitions the set X_n into disjoint classes $S_1, \ldots, S_k$ that will be named the *symmetry sets*. A function f is called *partially symmetric* if it has at least one symmetry set S_i with $|S_i| > 1$. If a function f has only one symmetry set $S = X_n$, then f is called *totally symmetric*. If $x_i, x_j \in S_l \subseteq X_n$ ($x_i \neq x_j$, $1 \leq l \leq k$), f is called *pairwise symmetric* in (x_i, x_j). A simple consequence of pairwise symmetry is the following lemma:

Lemma 1. *A function is pairwise symmetric in* (x_i, x_j) *iff* $f_{x_i\overline{x_j}} = f_{\overline{x}_ix_j}$.

A totally symmetric Boolean function can easily be characterized by its value vector, i.e., a Boolean $n+1$-tuple [18]. Since the output value of a totally symmetric function only depends on the number of variables set to 1, $n+1$ bits are sufficient to completely specify the functional behavior.

Example 2. The value vector for the EXOR function dependent on 5 variables is given by $(0, 1, 0, 1, 0, 1)$.

4 PSDKROs for Totally Symmetric Functions

In this section, we prove that the minimization of PSDKROs for totally symmetric functions can be performed in polynomial time and space.

Using the decomposition formulae from Section 2, each decomposition of the Boolean function $f : \mathbf{B}^n \to \mathbf{B}$ into two subfunctions $g, h : \mathbf{B}^{n-1} \to \mathbf{B}$ needs two out of the three functions f_i^0, f_i^1, and f_i^2.

In the following, we show that no more than n^3 distinct Boolean functions can be obtained out of a totally symmetric Boolean function f by "repeated applications" of the operations $f \mapsto f_i^0$, $f \mapsto f_i^1$, and $f \mapsto f_i^2$. Repeated application in this context means that, starting from the totally symmetric function f, the operators are applied. (It is easy to see that if f is symmetric, f_i^0, f_i^1, and f_i^2 are also symmetric.) Then, the operators are applied again to the newly derived functions. This is iterated until constant functions are obtained.

For a Boolean vector $\mathbf{a} = (a_0, \ldots, a_n) \in \{0,1\}^{n+1}$ we define the totally symmetric Boolean function $S[\mathbf{a}] : \mathbf{B}^n \to \mathbf{B}$ by

$$S[\mathbf{a}](x_1, \ldots, x_n) = a_{\sum_{1 \leq k \leq n} x_k}.$$

(Notice that every totally symmetric Boolean function is uniquely represented by a function $S[\mathbf{a}]$.)

On Boolean vectors, we define three transformations that correspond to f_i^0, f_i^1, and f_i^2 on the vector characterization of totally symmetric functions:

$$\begin{aligned} (a_0, \ldots, a_n)^{(0)} &= (a_0, \ldots, a_{n-1}) \\ (a_0, \ldots, a_n)^{(1)} &= (a_1, \ldots, a_n) \\ (a_0, \ldots, a_n)^{(2)} &= (a_0 \oplus a_1, a_1 \oplus a_2, \ldots, a_{n-1} \oplus a_n). \end{aligned}$$

Then, it directly follows:

Lemma 2. *For every vector* $\mathbf{a} \in \{0,1\}^{n+1}$, *every* $j \in \{0,1,2\}$, *and every* $i \in \{1, \ldots, n\}$, *we have*

$$S[\mathbf{a}]_i^j = S[\mathbf{a}^{(j)}].$$

Example 3. Let $\mathbf{a} = (0, 1, 0, 1)$. Then, we get

$$\begin{aligned} (0,1,0,1)^{(0)} &= (0,1,0) \\ (0,1,0,1)^{(0)} &= (1,0,1) \\ (0,1,0,1)^{(0)} &= (1,1,1). \end{aligned}$$

Now, let $f = S[\mathbf{a}]$ be the totally symmetric Boolean function for which we want to determine the minimal PSDKRO. It is sufficient to prove that no more than n^3 distinct Boolean vectors can be obtained out of $\mathbf{a}$ by repeated applications of the operations $a \mapsto a^{(0)}$, $a \mapsto a^{(1)}$, and $a \mapsto a^{(2)}$.

For this, we define a function δ that describes the mapping of the vectors that results from the applications of the corresponding operations. Starting with a totally symmetric function for each derivation, the corresponding operation can easily be described.

More formally, we obtain:

For $\mathbf{a} = (a_0, \ldots, a_n) \in \{0,1\}^{n+1}$, $r, l \in \{0, \ldots, n\}$, and $C \subseteq \{0, \ldots, n\}$ we define:[1]

1. To keep the definition as simple as possible, the function δ is only well defined in the range that is needed later.

Authorized licensed use limited to: National Taiwan University. Downloaded on December 15,2020 at 10:51:46 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
Multiple Output Totally Symmetric Functions

name	*in*	*out*	ESOP		PSDKRO	
			terms	time	terms	time
rd53	5	3	15	2.1	20	0.01
rd73	7	3	36	24.1	63	0.01
rd84	8	4	54	50.4	107	0.01

$$\delta(\mathbf{a}, r, l, C) := (b_0, \ldots, b_t) \in \{0,1\}^{l+1}$$

with

$$b_i = \bigoplus_{j \in C \cup \{0,\ldots,n-l-r\}} a_{r+j+i}.$$

For two sets C, D let $C \uplus D$ denote their symmetric difference

$$C \uplus D = (C \cup D) \setminus (C \cap D).$$

For a set $C \subseteq \{0, \ldots, n\}$ define

$$C^{(2)} = (C \uplus \{i+1 \mid i \in C\}) \cap \{0, \ldots, n\}.$$

For a totally symmetric Boolean function that is characterized by its value vector

$$\mathbf{a} = (a_0, \ldots, a_n) \in \{0,1\}^{n+1},$$

we have:

$$\mathbf{a} = \delta(\mathbf{a}, 0, n, \{0\}).$$

Then, it holds for the three transformations with $r, l \in \{0, \ldots, n\}$ and $C \subseteq \{0, \ldots, n-l-r\}$:

1. $\delta(\mathbf{a}, r, l, C)^{(0)} = \delta(\mathbf{a}, r, l-1, C)$,
2. $\delta(\mathbf{a}, r, l, C)^{(1)} = \delta(\mathbf{a}, r+1, l-1, C)$,
3. $\delta(\mathbf{a}, r, l, C)^{(2)} = \delta(\mathbf{a}, r, l-1, C^{(2)})$.

It is now sufficient to notice that no more than $n+1$ distinct sets can be obtained out of $\{0\}$ by at most n applications of the operation $C \mapsto C^{(2)}$. Furthermore, we have at most $n^2/2$ choices for r and l and at most $n+1$ choices for C. Thus, there are at most n^3 different vectors $\delta(\mathbf{a}, r, l, C)$. And, so, there are at most n^3 different functions to be represented.

Thus, we have proven:

Theorem 1. *At most n^3 different functions can be obtained by the decompositions of type S, pD, and nD starting from a totally symmetric function.*

Notice that the number of different expressions might be larger, but the number of expressions that differ in size with respect to number of product terms is also bound by n^3. Now, it is sufficient to use a data structure whose representation size is proportional to the number of subfunctions to be represented. This property is fulfilled by OBDDs (which are used in the next section). Using the fact that only n^3 different choices have to be considered, the exact algorithm from [13] can be applied and has polynomial worst case behavior.[2]

5 Implementation of the Algorithm

For the implementation of the algorithm, we used *Ordered Binary Decision Diagrams* (OBDDs) [3]. The starting point of our algorithm is the OBDD representation of the totally symmetric function that has to be minimized. The size of the representation is $O(n^2)$ due to Lemma 1 [3].

Starting from the root of the OBDD, the graph is recursively traversed and, at each node, an EXOR operation is carried out. An EXOR operation on OBDDs can be performed in polynomial time. Using the fact that, for each of the decomposition formulae from Section 2, two out of the three possible successors f_i^0, f_i^1, and f_i^2 are needed, the two minimizing the resulting PSDKRO are chosen.[3] For each node v, a minimal number of terms that is needed for the representation as a PSDKRO is stored in the variable $v.val$. Thus, each node has to be evaluated only once. Since, due to Theorem 1, only n^3 nodes have to be considered and, at each node, only a polynomial algorithm has to be carried out the complete algorithm has polynomial worst case behavior.

A sketch of the algorithm is given in Fig. 1.

TABLE 3
Multiple Output Benchmark Functions

name	*in*	*out*	SOP	ESOP		PSDKRO	
				terms	time	terms	time
add6	12	7	355	127	229.49	132	0.36
vg2	25	8	110	184	932.61	324	2.58
t481	16	1	481	13	161.49	13	0.01
5xp1	7	10	75	32	10.97	38	0.09

6 Experimental Results

In this section, we present experimental results for benchmark functions. All runtimes (if not mentioned otherwise) are given in CPU seconds on a *Sun Sparc 4* workstation.

First, we consider some single output functions from [2], [1]. The results, including the runtimes, are given in Table 1. *in* denotes the number of inputs of the corresponding benchmarks. In column FPRM, the results determined by MINGRM are given. MINGRM is the exact FPRM minimizer presented in [17]. The runtimes for MINGRM are given in CPU seconds on a Sequent S27—two 386 processors—machine. (The approach from [17] does not consider symmetry.) In column KRO, the results determined by the exact KRO minimizer for symmetric functions from [1] are given. The runtimes are given in CPU seconds on an *HP Apollo 715/50* workstation. The results for the heuristic ESOP minimizer MINT [7] are given in column ESOP. The results of the approach presented in this paper are given in column PSDKRO. As can easily be seen, only the programs who consider symmetry can efficiently minimize the functions. In contrast to the exact KRO minimizer from [1], PSDKROs often allow more efficient representation. Especially on functions of many inputs, the PSDKRO minimizer is much faster (see, e.g., *border25*). MINT is, in some cases, able to further minimize the number of terms needed for the representation of the function, but, in these cases, it also needs more than 500 times longer. The runtimes of the OBDD-based PSDKRO minimizer for all considered examples are negligible.

In the next series of experiments, we consider multi-output functions that are totally symmetric with respect to each output.[4] The results in comparison with the heuristic ESOP minimizer

2. The ordering of the variables has not to be considered, since we deal with totally symmetric Boolean functions.

3. The idea of the algorithm is the same as used in [13] for PSDKRO minimization using *Ternary Decision Diagrams*.

4. The multiple output minimization has been reduced to single output minimization by multivalued interpretation [15].

Authorized licensed use limited to: National Taiwan University. Downloaded on December 15,2020 at 10:51:46 UTC from IEEE Xplore. Restrictions apply.

MINT are given in Table 2. The relations between the two minimizers are the same as for the single output case.

Finally, we compare our minimizer to MINT on a set of (not totally symmetric) benchmark functions. Notice that our minimizer also computes exact solutions in the case of nonsymmetric functions, but the runtime cannot be guaranteed to be polynomial, i.e., the number of OBDD nodes that have to be created might become exponential. The results with respect to runtime and number of terms are given in Table 3. In column SOP, the number of terms for the minimized SOP is given. As can easily be seen, our results can be obtained much faster (up to a factor of 1,000), although our tool was not designed for this case. The results counted in number of terms are only slightly worse than those of the rule-based ESOP minimizer MINT. On the other hand, the results are often much better than the SOP representation (see, e.g., *t481*).

7 Conclusions

In this paper, a polynomial algorithm for exact minimization of PSDKROs for totally symmetric functions has been presented. An OBDD-based implementation was given. It performs very fast on all considered symmetric benchmark examples.

With this result, PSDKROs are the largest class of AND/EXOR expressions for which a polynomial algorithm for exact minimization of totally symmetric functions is known.

Finally, we compared our approach to general ESOP minimization. It has been demonstrated that PSDKROs are a good alternative. They are only slightly larger, but can be minimized much faster.

Acknowledgments

The author would like to thank Ralph Werchner for inspiring the idea of the proof of Theorem 1.

References

[1] B. Becker and R. Drechsler, "Exact Minimization of Kronecker Expressions for Symmetric Functions," *Proc. Int'l Symp. Circuits and Systems,* pp. IV:388-IV:391, 1996.

[2] R.K. Brayton, G.D. Hachtel, C. McMullen, and A.L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic, 1984.

[3] R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. Computers,* vol. 35, pp. 677-691, 1986.

[4] M. Chatterjee, D.K. Pradhan, and W. Kunz, "LOT: Logic Optimization with Testability—New Transformations Using Recursive Learning," *Proc. Int'l Conf. Computer-Aided Design,* pp. 318-325, 1995.

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

[6] R. Drechsler and B. Becker, "Sym*pathy*: Fast Exact Minimization of Fixed Polarity Reed-Muller Expressions for Symmetric Functions," *Proc. European Design and Test Conf.,* pp. 91-97, 1995.

[7] T. Kozlowski, E.L. Dagless, and J.M. Saul, "An Enhanced Algorithm for the Minimization of Exclusive-Or Sum-of-Products for Incompletely Specified Functions," *Proc. Int'l Conf. Computer Design,* pp. 244-249, 1995.

[8] S.M. Reddy, "Easily Testable Realizations for Logic Functions," *IEEE Trans. Computers,* vol. 21, pp. 1,183-1,188, 1972.

[9] I.S. Reed, "A Class of Multiple-Error-Correcting Codes and Their Decoding Scheme," *IRE Trans. Information Theory,* vol. 3, pp. 6-12, 1954.

[10] U. Rollwage, "The Complexity of mod-2 Sum PLA's for Symmetric Functions," *Proc. IFIP WG 10.5 Workshop Applications of the Reed-Muller Expansion in Circuit Design,* pp. 6-12, Hamburg, 1993.

[11] K.K. Saluja and S.M. Reddy, "Fault Detection Test Sets for Reed-Muller Canonical Networks," *IEEE Trans. Computers,* vol. 24, pp. 995-998, 1975.

[12] A. Sarabi and M.A. Perkowski, "Design for Testability Properties of AND/XOR Networks," *Proc. IFIP WG 10. 5 Workshop Applications of the Reed-Muller Expansion in Circuit Design,* pp. 147-153, Hamburg, 1993.

[13] T. Sasao, "AND-EXOR Expressions and Their Optimization," *Logic Synthesis and Optimization,* T. Sasao, ed. Kluwer Academic, 1993.

[14] T. Sasao, "EXMIN2: A Simplification Algorithm for Exclusive-OR-Sum-of Products Expressions for Multiple-Valued-Input Two-Valued-Output Functions," *IEEE Trans. Computer-Aided Design,* vol. 12, pp. 621-632, 1993.

[15] T. Sasao, *Logic Synthesis and Optimization.* Kluwer Academic, 1993.

[16] J. Saul, B. Eschermann, and J. Frössl, "Two-Level Logic Circuits Using EXOR Sums of Products," *IEE Proc.,* vol. 140, pp. 348-356, 1993.

[17] I. Schäfer and M.A. Perkowski, "Multiple-Valued Input Generalized Reed-Muller Forms," *Proc. Int'l Symp. Multi-Valued Logic,* pp. 40-48, 1991.

[18] I. Wegener, *The Complexity of Boolean Functions.* John Wiley & Sons Ltd. and B.G. Teubner, Stuttgart, 1987.

Authorized licensed use limited to: National Taiwan University. Downloaded on December 15,2020 at 10:51:46 UTC from IEEE Xplore. Restrictions apply.