Simplification of Exclusive-or Sum-of-Products Expressions Through Function Transformation

Takashi Hirayama, Masatoshi Takahashi, and Yasuaki Nishitani
Department of Computer and Information Sciences, Iwate University,
4-3-5 Ueda, Morioka, Iwate, 020-8551 Japan.
Email: {hirayama,masa,nisitani}@kono.cis.iwate-u.ac.jp

***Abstract*— Exclusive-or sum-of-products expressions (ESOPs) are the most general AND-EXOR expressions. This paper presents a new data structure called a function product (FP) and an algorithm for obtaining simplified ESOPs through transformation of FPs. Our algorithm takes the following steps: converting an initial ESOP into an EXOR of FPs (EX-FP), simplifying the EX-FP by repeating the transformation of FPs, and reconverting the resulting EX-FP into the simplified ESOP. We give experimental results on benchmarks to demonstrate the superiority of our method in reduction of literals.**

I. Introduction

Logic circuits including exclusive-or (EXOR) gates have some advantages over traditional circuits with only AND and OR gates. EXOR-based realization can improve the testability [6], [17], [22] and often reduces the circuit area [2], [3], [13]. For arithmetic functions, error-correcting functions, and telecommunication functions, AND-EXOR circuits are smaller than AND-OR ones [18]. AND-EXOR logic expressions, which correspond to AND-EXOR two-level circuits, have been studied as the fundamentals of the EXOR-based realization. Current applications of AND-EXOR expressions are logic synthesis of new type of circuits such as reversible logic circuits [26] and quantum circuits [10].

There are several classes of AND-EXOR expressions [7], [21] such as PPRMs (positive-polarity Reed-Muller expressions), FPRMs (fixed-polarity Reed-Muller expressions), DFPRMs (double-fixed-polarity Reed-Muller expressions), and ESOPs (exclusive-or sum-of-products expressions). ESOPs are the expressions such that arbitrary product terms are combined by EXORs. Among the classes of AND-EXOR expressions, ESOPs are the most general AND-EXOR expressions and require the fewest product terms to represent logic functions. The number of product terms of an ESOP F is called the size of F. Among all ESOPs that represent a logic function f, those with the minimum size are called exact minimum ESOPs of f. The size of an exact minimum ESOP of f is denoted by $\tau(f)$.

Although several algorithms for exact minimization of ESOPs are known [8], [16], [20], [25], they require huge computation time or memory to guarantee the minimality. Fast algorithms that simplify ESOPs without guaranteeing minimality are more practical. Therefore a lot of heuristic algorithms for simplifying ESOPs have been proposed [1], [4], [5], [15], [19], [24], [27]. These algorithms simplify ESOPs by applying heuristic rewriting rules. Efficiency of simplification is measured by the number of products and literals of the resulting ESOPs. Mainly, reduction of products has been studied for a long time. Sasao's algorithm [19] and Song-Perkowski's algorithm [24] are known to be especially efficient for product reduction. It has been confirmed that, for some benchmark circuits, the number of products of ESOPs obtained by these algorithms is consistent with that of exact minimum ESOPs [9]. From the fact, it is believed that these algorithms compute approximately minimum ESOPs for many cases. Literals are usually reduced with a decrease in products. Besides being product-efficient, Mishchenko-Perkowski's algorithm [15] is literal-efficient. However, it has been reported in [9] that further reduction of literals is possible. Reduction of literals saves the realization costs for not only synthesis of traditional logic circuits but also technology mapping of FPGAs.

In this paper, an algorithm for obtaining simplified ESOPs through transformation of EX-FPs is proposed. An EX-FP is a logic representation such that FPs are combined by EXOR operations. An FP is a new data structure given in Section II. Our simplification algorithm is described in Section III. The efficiency of our algorithm is confirmed by experiments in Section IV.

II. Definitions and Function Product

Definition 1. x and $\bar{x}$ are *literals* of a variable x. A logical product of at most one literal on each variable is a *product term*, or a *product* for short. An exclusive-or of products forms an *Exclusive-or Sum-Of-Products expression* (*ESOP*).

Definition 2. The size of an ESOP F is the number of product terms of F, denoted by $\tau(F)$. Among all ESOPs that represent a logic function f, those with the minimum size are called minimum ESOPs of f, and the size of a minimum ESOP of f is denoted by $\tau(f)$.

Definition 3. Let X be the set of n variables $\{x_1, x_2, \ldots, x_n\}$ and k $(1 \leq k \leq n)$ be a constant. A *function product* (FP) is a functional decomposition written as $f_1(X_1) \cdot f_2(X_2) \cdot \cdots \cdot f_{n'}(X_{n'})$ where $X_1, X_2, \ldots, X_{n'}$ are the partition of X such that $|X_i| = k$ and $n' = \lceil n/k \rceil$. If n is not a multiple of k, $|X_1| = n \bmod k$. $f_1, f_2, \ldots, f_{n'}$ are called subfunctions of the FP, and k is called the size of the variable partition. An *EXOR of FPs* (*EX-FP*) is an exclusive-or combination of FPs. The number of FPs of an EX-FP F is denoted by $t(F)$.

1-4244-0387-1/06/$20.00

Example 1. $f_1 \cdot f_2 = x_2 \cdot x_6$, $f_1' \cdot f_2' = (x_1 x_2 \oplus x_3) \cdot \bar{x}_5 x_6$, and $f_1'' \cdot f_2'' = x_1 \cdot (x_4 \bar{x}_5 \oplus \bar{x}_4 \oplus x_5 x_6)$ are 6-variable FPs with $k = 3$. $f_1 \cdot f_2 \oplus f_1' \cdot f_2' \oplus f_1'' \cdot f_2''$ is an EX-FP. The subfunctions f_1, f_1', and f_1'' have 3 variables $\{x_1, x_2, x_3\}$, and f_2, f_2', and f_2'' have 3 variables $\{x_4, x_5, x_6\}$.

An ordinary product can be regarded as an FP because a product is a form of AND decomposition. Thus, EX-FPs, which are EXOR-sums of FPs, are a generalization of ESOPs.

The following is a useful property of EX-FPs. An EX-FP can be converted into an ESOP by applying the distributive law if the subfunctions of the FPs are represented by ESOPs like Example 1. By applying the distributive law, FPs in Example 1 are written as follows: $f_1 \cdot f_2 = x_2 x_6$, $f_1' \cdot f_2' = (x_1 x_2 \oplus x_3) \cdot \bar{x}_5 x_6 = x_1 x_2 \bar{x}_5 x_6 \oplus x_3 \bar{x}_5 x_6$, and $f_1'' \cdot f_2'' = x_1 \cdot (x_4 \bar{x}_5 \oplus \bar{x}_4 \oplus x_5 x_6) = x_1 x_4 \bar{x}_5 \oplus x_1 \bar{x}_4 \oplus x_1 x_5 x_6$. Hence the EX-FP $f_1 \cdot f_2 \oplus f_1' \cdot f_2' \oplus f_1'' \cdot f_2''$ can be converted into $x_2 x_6 \oplus x_1 x_2 \bar{x}_5 x_6 \oplus x_3 \bar{x}_5 x_6 \oplus x_1 x_4 \bar{x}_5 \oplus x_1 \bar{x}_4 \oplus x_1 x_5 x_6$.

Definition 4. Let $P = f_1 \cdot f_2 \cdot \cdots \cdot f_n$ be an FP, and $F_1, F_2, \ldots, F_n$ be the minimum ESOPs of $f_1, f_2, \ldots, f_n$, respectively. The size of FP P, which is denoted by $\tau(P)$, is the number of products of the ESOP converted from $F_1 \cdot F_2 \cdot \cdots \cdot F_n$. For an EX-FP $F = P_1 \oplus P_2 \oplus \cdots \oplus P_m$, the size of EX-FP F is denoted by $\tau(F)$, and defined as $\tau(F) = \sum_{i=1}^{m} \tau(P_i)$.

III. Transformation and Algorithm

The key idea of our strategy is that a simplified ESOP is obtained by transformation of the EX-FP. An ESOP represents a function whereas a function can be represented by multiple ESOPs. Similarly (subfunctions of) an FP can be represented by multiple ESOPs. This suggests that manipulation of EX-FPs leads to simplification of multiple candidates of ESOPs. Based on the above idea, we present transformation rules for EX-FPs and an algorithm for simplifying EX-FPs. After simplifying an EX-FP, each subfunction of the FPs can be converted into its corresponding ESOP if the table of ESOPs for subfunctions is prepared in advance. So we use the table of the minimum ESOPs of all 3-variable functions and fix the size of variable partition as $k = 3$ throughout the paper. The reason we chose $k = 3$ is that the table can be generated by a known exact minimization algorithm [11] and has a moderate size to store in a computer memory.

Transformation Rules

$$
\begin{aligned}
& f_1 \cdot f_2 \cdot \alpha \oplus f_1' \cdot f_2' \cdot \alpha \\
&= f_1 \cdot (f_2 \oplus f_2') \cdot \alpha \oplus (f_1 \oplus f_1') \cdot f_2' \cdot \alpha && (1) \\
&= (f_1 \oplus f_1') \cdot f_2 \cdot \alpha \oplus f_1' \cdot (f_2 \oplus f_2') \cdot \alpha && (2) \\
f_1 \cdot \alpha \oplus f_1' \cdot \alpha &= (f_1 \oplus f_1') \cdot \alpha && (3) \\
f_1 \cdot \alpha &= (f_1 \oplus h) \cdot \alpha \oplus h \cdot \alpha && (4)
\end{aligned}
$$

Eqs. (1) and (2) are called Reshape. For an EX-FP F, this transformation keeps the number of FPs ($t(F)$). Eq. (3), which is a special case of $f_2 = f_2'$ in Reshape, is called Merge. Eq. (4) is called Split. $t(F)$ increases with Split and decreases with Merge. Merge directly simplifies an EX-FP. Reshape and Split are used to enhance the chances of applying Merge.

Eqs. (1)–(4) have been presented in [19] as a general idea of ESOP transformation. By replacing subfunctions f_1, f_2, f_1', f_2', and h with their possible candidates of ESOPs, a lot of variations of ESOP transformation rules have been proposed and simplification algorithms using them have been implemented [1], [19], [24]. Unlike those former algorithms, our algorithm can use the general transformation rules (1)–(4) directly. Instead of considering the variations in ESOP representation, we use the table of minimum ESOPs to convert subfunctions of FPs into their ESOPs.

---

1) An initial EX-FP F is given. An ESOP or a DSOP is also acceptable as F.
2) As the initial simplification, check each pair of FPs of F and apply Merge to them if applicable.
3) Repeat the following until F is simplified enough.
   3-1) Check each pair of FPs. Apply Reshape if the rule is applicable and $\tau(P_1) + \tau(P_2) \geq \tau(P_3) + \tau(P_4)$ holds, where P_1 and P_2 are a pair of FPs before transformation and P_3 and P_4 are the resulting FPs after transformation.
   3-2) Check each FP. Apply Split if the rule is applicable and $\tau(P_1) = \tau(P_2) + \tau(P_3)$ holds, where P_1 is an FP before transformation and P_2 and P_3 are the resulting FPs after transformation.
4) Convert the EX-FP F into the ESOP by looking up the table of minimum ESOPs.
5) Return the ESOP.

Fig. 1. Simplification Algorithm

Our simplification algorithm is described in Fig. 1. In Step 1, an arbitrary ESOP can be used as an initial EX-FP because EX-FPs are general expressions of ESOPs. DSOPs (Disjoint Sum-Of-Products expressions) are a kind of ESOPs. Since the initial EX-FP given in a form of an ESOP tends to be inefficient as EX-FPs, Step 2 makes it compact with Merge. After Step 2, Merge is not used because Reshape includes it. In Step 3-1, the condition $\tau(P_1) + \tau(P_2) \geq \tau(P_3) + \tau(P_4)$ is used to avoid raising $\tau(F)$ during the transformation. The aim of our algorithm is to reduce not $t(F)$ (the FPs of an EX-FP) but $\tau(F)$ (the products of the resulting ESOP). With a decrease in products, literals will also be reduced. If both of Eqs. (1) and (2) are applicable, smaller one in the scale of $\tau(P_3) + \tau(P_4)$ is chosen. In Step 3-2, if Eq. (4), Split, is applicable with multiple candidates of h, one of them is chosen randomly. The constant 0 function and f_1 itself are excluded from the candidates of h in order to omit the trivial transformation. The condition $\tau(P_1) = \tau(P_2) + \tau(P_3)$ is used to avoid raising $\tau(F)$ during the transformation.

IV. Experimental Results

We implemented our simplification algorithm in Scheme. In our implementation, Step 3 is repeated 1,000 times. The program was executed on Intel Celeron 2.2GHz computers, whose operating system is FreeBSD 4.10-R. It is known that the minimization problem for multiple-output functions is equivalent to that of their characteristic functions and that the transformation rules (1)–(4) are also applicable to the characteristic functions [19], [24]. By extending our algorithm to the simplification of the characteristic functions, we obtained simplified ESOPs of the MCNC benchmark set [14]. Since the benchmark set is given in the PLA format and the format can be converted into DSOPs easily, we used DSOPs as initial EX-FPs in our experiments.

Table I shows a comparison of the number of products of simplified ESOPs between our algorithm and previous methods [12], [15], [19], [23]. 'Name' is the name of the circuit and '(in, out)' is the number of inputs and outputs of the circuit. Since our algorithm has randomness, we made 10 experiments for each benchmark circuit. Among these experiments, the minimum results are shown at 'min' and the results obtained most frequently are shown at '(mode)'. 'Time' is the average CPU time in second. For relatively smaller benchmark circuits, the results of our method are better than or equal to those of other methods. Our algorithm obtained the smallest results for clip and rd73. However, our results tend to be worse for larger benchmarks. Our computation time is also worse; requiring 3–10 times longer execution than EXORCISM-4 [15], which is known as the fastest algorithm. The comparison of computation time is omitted in the table.

Table II shows a comparison of the number of literals of simplified ESOPs. 'Proposed' is the number of literals of ESOPs that are shown at 'min' in Table I. For the most benchmark circuits (except alu4 and con1), the proposed method generated better results than the other methods [12], [15], [19], [23], [24]. These experimental results show that our method can reduce literals efficiently. Although no explicit literal reduction have been considered in our algorithm, the strategy of simplifying EX-FPs and looking up the table of minimum ESOPs seems to work effective for literals.

V. Conclusion

We have presented a new strategy for simplifying ESOPs, in which ESOPs are simplified through transformation of EX-FPs. EX-FPs are a new data structure to deal with multiple candidates of ESOPs. We have proposed a simple algorithm that generates optimum ESOPs by simplifying EX-FPs. The experimental results have shown that our method produces better ESOPs in the number of literals than previous methods for the most benchmark circuits. In the number of products $\tau(F)$, however, our superiority is limited for relatively smaller circuits only. Longer computation time is also our disadvantage. Improvements of time and $\tau(F)$ for larger circuits is our future work.

TABLE I

Comparison of the number of products with other methods

Benchmark		Previous Methods				Proposed		Time
Name	(in, out)	[19]	[23]	[12]	[15]	min	(mode)	sec.
5xp1	(7,10)	34	33	32	31	31	(31)	39
9sym	(9,1)	53	51	51	51	51	(52)	118
alu4	(14,8)	-	-	-	411	448	(457)	4101
apex1	(45,45)	-	-	-	285	286	(286)	2224
apex2	(39,3)	-	-	-	-	1646	(1646)	175574
apex3	(54,50)	-	-	-	-	276	(282)	2791
apex4	(9,19)	-	-	-	-	537	(568)	4253
apex5	(117,88)	-	-	-	398	399	(399)	12848
b12	(15,9)	28	28	28	28	28	(28)	20
clip	(9,5)	68	65	64	63	62	(63)	50
con1	(7,2)	-	9	-	-	9	(9)	6
cordic	(23,2)	-	-	-	-	776	(776)	66297
duke2	(22,29)	-	-	-	-	79	(79)	91
e64	(65,65)	-	-	-	65	65	(65)	109
ex5p	(8,63)	-	-	-	-	72	(74)	247
misex1	(8,7)	-	12	-	-	12	(12)	7
misex2	(25,18)	-	27	-	-	27	(27)	12
misex3	(14,14)	-	-	-	501	547	(550)	6574
rd53	(5,3)	15	14	15	14	14	(15)	28
rd73	(7,3)	42	38	36	36	35	(35)	58
rd84	(8,4)	59	57	54	58	59	(60)	116
sao2	(10,4)	29	28	27	28	27	(28)	14
seq	(41,35)	259	249	248	246	247	(248)	1533
sqrt8	(8,4)	-	-	-	-	17	(17)	12
squar5	(5,8)	-	19	-	-	18	(18)	8
t481	(16,1)	13	13	13	13	13	(13)	89
table3	(14,14)	-	-	-	166	166	(166)	295
table5	(17,15)	-	-	-	156	156	(156)	485
vg2	(25,8)	184	184	184	184	184	(184)	511

TABLE II

Comparison of the number of literals with other methods

Name	[19]	[12]	[23]	[24]	[15]	Proposed
5xp1	186	181	178	-	175	117
9sym	433	427	425	-	426	375
alu4	-	-	4816	4430	4520	4443
apex1	-	-	3796	3820	3998	2758
apex2	-	-	-	-	-	35299
apex3	-	-	-	-	-	1953
apex4	-	-	-	-	-	3632
apex5	-	-	4038	4027	4065	3591
b12	164	167	164	-	166	131
clip	517	492	490	-	479	365
con1	-	-	37	-	-	38
cordic	-	-	-	-	-	10845
duke2	-	-	-	-	-	689
e64	-	-	2210	2272	2270	2157
ex5p	-	-	-	-	-	408
misex1	-	-	89	-	-	48
misex2	-	-	210	-	-	170
misex3	-	-	6837	6632	6141	5589
rd53	60	69	57	-	57	39
rd73	221	194	191	-	197	135
rd84	330	303	317	-	333	284
sao2	308	311	286	-	288	226
seq	5305	5187	4833	4822	5048	3290
sqrt8	-	-	-	-	-	67
squar5	-	-	-	-	-	49
t481	53	53	53	-	53	42
table3	-	-	2491	2491	2630	1854
table5	-	-	2453	2449	2545	1847
vg2	1992	2033	1993	1988	2010	1880

REFERENCES

[1] D. Brand, and T. Sasao, "Minimization of AND-EXOR expressions using rewrite rules," *IEEE Trans. Comput.*, vol.42, no.5, pp.568–576, May 1993.

[2] S. Chattopadhyay, S. Roy, and P. P. Chaudhuri, "KGPMIN: an efficient multilevel multioutput AND-OR-XOR minimizer," *IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst.*, vol.16, no.3, pp.257–265, March 1997.

[3] D. Debnath and T. Sasao, "Minimization of AND-OR-EXOR three-level networks with AND gate sharing," *IEICE Trans. Inf. & Syst.*, vol.E80-D, no.10, pp.1001–1008, Oct. 1997.

[4] H. Fleisher, M. Tavel, and J. Yeager, "Computer algorithm for minimizing Reed-Muller canonical forms," *IEEE Trans. Comput.*, vol.C-36, no.2, pp.247–250, Feb. 1987.

[5] M. Helliwell, and M. A. Perkowski, "A fast algorithm to minimize multi-output mixed-polarity generalized Reed-Muller forms," *Proc. 25th ACM/IEEE Design Automation Conference*, pp.427–432, June 1988.

[6] T. Hirayama, G. Koda, Y. Nishitani, and K. Shimizu, "Easily testable realization based on single-rail-input OR-AND-EXOR expressions," *IEICE Trans. Inf. & Syst.*, vol.E82-D, no.9, pp.1278–1286, Sept. 1999.

[7] T. Hirayama, K. Nagasawa, Y. Nishitani, and K. Shimizu, "Double fixed-polarity Reed-Muller expressions: a new class of AND-EXOR expressions for compact and testable realization," *Trans. IPS Japan*, vol.42, no.4, pp.983–991, April 2001.

[8] T. Hirayama and Y. Nishitani, "A faster algorithm of minimizing AND-EXOR expressions," *IEICE Trans. Fundamentals*, vol. E85-A, no. 12, pp. 2708–2714, Dec. 2002.

[9] T. Hirayama, T. Sato, and Y. Nishitani, "Minimizing AND-EXOR expressions of some benchmark functions," *Proc. of 6th International Symposium on Representations and Methodology of Future Computing Technologies, Trier, Germany*, pp. 69–76, Mar. 2003.

[10] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation rules for designing CNOT-based quantum circuits," *Proc. 39th ACM/IEEE Design Automation Conference*, pp.419–424, June 2002.

[11] N. Koda and T. Sasao, "Four variable AND-EXOR minimization expressions and their properties," *IEICE Trans. Inf. & Syst. D-I*, vol.J74-D-I, no.11, pp.765–773, Nov. 1991.

[12] T. Kozlowski, *Application of exclusive-OR logic in technology independent logic optimisation*, Ph.D Thesis, Jan. 1996.

[13] F. Luccio and L. Pagli, "On a new boolean function with applications," *IEEE Trans. Comput.*, vol.48, no.3, pp.296–310, March 1999.

[14] K. McElvain, "IWLS'93 Benchmark Set: Version 4.0," *Distributed as part of the MCNC International Workshop on Logic Synthesis '93 benchmark distribution.*, May. 1993.

[15] A. Mishchenko and M. Perkowski, "Fast heuristic minimization of exclusive-sums-of-products," *Proc. of Reed-Muller Workshop 2001*, Mississippi, U.S.A., pp.242–250, Aug. 2001.

[16] M. A. Perkowski and M. Chrzanowska-Jeske, "An exact algorithm to minimize mixed-radix exclusive sums of products for incompletely specified Boolean functions," *Proc. International Symposium Circuits & Syst.*, USA, pp.1652–1655, May 1990.

[17] S. M. Reddy, "Easily testable realization for logic functions," *IEEE Trans. Comput.*, vol.C-21, no.11. pp.1183-1188, Nov. 1972.

[18] T. Sasao and P. Besslich, "On the complexity of mod-2 sum PLA's," *IEEE Trans. Comput.*, vol.39, no.2, pp.262–266, Feb. 1990.

[19] T. Sasao, "EXMIN2: A simplification algorithm for exclusive-OR sum-of-products expressions for multiple-valued-input two-valued-output functions," *IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst.*, vol.12, no.5, pp.621–632, May 1993.

[20] T. Sasao, "An exact minimization of AND-EXOR expressions using BDD's," *Proc. IFIP WG10.5 Reed-Muller'93*, Germany, pp.91–98, 1993.

[21] T. Sasao, "Representations of logic functions using EXOR operators," in *Representations of Discrete Functions*, eds. T. Sasao and M. Fujita, pp.29–54, Kluwer Academic Publishers, 1996.

[22] T. Sasao, "Easily testable realizations for generalized Reed-Muller expansions," *IEEE Trans. Comput.*, vol.46, no.6, pp.709–716, June 1997.

[23] N. Song and M. A. Perkowski, "EXORCISM-MV-2: Minimization of exclusive sum-of-products expressions for multiple-valued input, incompletely specified functions," *Proc. IEEE ISMVL*, pp.132–137, 1993.

[24] N. Song and M. A. Perkowski, "Minimization of exclusive sum-of-products expressions for multiple-valued input, incompletely specified functions," *IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst.*, vol.15, no.4, pp.385–395, April 1996.

[25] S. Stergiou and G. Papakonstantinou, "Exact minimization of esop expressions with less than eight product terms," *Journal of Circuits, Systems, and Computers*, World Scientific, vol. 13, no. 1, pp. 1–15, Feb. 2004.

[26] G. Yang, W. N. N. Hung, X. Song, and M. Perkowski, "Majority-based reversible logic gates," *Theoretical Computer Science*, vol.334, issues 1-3, pp.259–274, April 2005.

[27] Y. Ye and K. Roy, "An XOR-based decomposition diagram and its application in synthesis of AND/XOR networks," *IEICE Trans. Fundamentals*, vol.E80-A, no.10, pp.1742–1748, Oct. 1997.