A Fast and Efficient Heuristic ESOP Minimization Algorithm

S. Stergiou
stergiou@cslab.ece.ntua.gr

K. Daskalakis
kdask@cs.ntua.gr

G. Papakonstantinou
papakon@cslab.ece.ntua.gr

Department of Electrical and Computer Engineering
National Technical University of Athens
Greece

ABSTRACT

This work presents theoretical results and an efficient heuristic algorithm for minimizing single – output exclusive – or sum–of–products expressions, based on an iterative product term transformation paradigm. Experimental results verify the efficiency of the algorithm in terms of execution times and product term count of the produced expression, when compared to a state–of–the–art heuristic ESOP minimizer for single–output benchmark and randomly generated functions.

Categories and Subject Descriptors

B.6.0 [**Hardware**]: Logic Design—*General*

General Terms

Algorithms Design

Keywords

ESOP XOR heuristic minimization

1. INTRODUCTION

The problem of minimization of exclusive – or sum – of – products (ESOP) expressions is an active research area, due to advantages ESOP expressions possess relative to sum–of–products (SOPs), such as excellent testability properties, and experimentally observed smaller expression sizes [4]. Moreover, for an n-variable boolean function, the upper bound in the number of product terms of ESOPs is $29 \cdot 2^{n-7}, n > 6$ [2] as opposed to 2^{n-1} for SOPs.

The problem of exact ESOP minimization, interesting in its own respect, has not yet found an efficient solution for more than six variables in the general case [7, 8, 1, 5, 6], although there exist published works that provide results for up to twenty variables, when the number of the product terms in the final expression is restricted [8, 3, 12, 11, 13].

From a practical viewpoint, the problem of heuristic ESOP minimization seems more attractive and numerous heuristic ESOP minimizers have been proposed and gradually refined in the past decades [4, 9, 10, 11]. A comprehensive survey of previous work is presented in [4].

In this work, a single–output ESOP minimization algorithm is detailed. Given a boolean function f, an initial expression is generated for it. This expression is gradually minimized, by carefully selecting a number of product terms in it and transforming them in a way such that the expression size is reduced. The novelty of this work lies in the proposed product term transformation operations.

The structure of the paper is as follows. Section 2 presents basic definitions. Section 3 analyzes the theoretical formulation. Section 4 details the heuristic ESOP minimizer. Section 5 contains the experimental results. Section 6 concludes the paper.

2. DEFINITIONS

Let x_i^j denote $\overline{x_i}, x_i, 1$ for $j = 0, 1, 2$ respectively.

DEFINITION 1. *A subfunction $f^i, i = 0, 1, 2$ of a boolean function $f(x_1, x_2, \ldots, x_n)$ is defined as*

$$\begin{aligned} f^0 &= f(0, x_2, \ldots, x_n) \\ f^1 &= f(1, x_2, \ldots, x_n) \\ f^2 &= f^0 \oplus f^1 \end{aligned} \quad (1)$$

where $\oplus$ means modulo–2 addition.

DEFINITION 2. *The weight $w(f)$ of a boolean function f is defined as the minimum number of product terms (cubes) among all possible ESOP expressions of the function f.*

DEFINITION 3. *An exact ESOP expression of a boolean function f (or an exact cover for simplicity), is an ESOP expression of f, with the number of product terms equal to its weight.*

DEFINITION 4. *The distance $d(c_i, c_j)$ between two cubes c_i, c_j is defined as the number of literals in which they differ.*

We also recall here some basic formulas.

$$\begin{aligned} f &= \overline{x_1} f^0 \oplus x_1 f^1 && \text{Shannon expansion} \\ f &= f^0 \oplus x_1 f^2 && \text{positive Davio expansion} \\ f &= f^1 \oplus \overline{x_1} f^2 && \text{negative Davio expansion} \end{aligned} \quad (2)$$

GLSVLSI'04, April 26–28, 2004, Boston, Massachusetts, USA.

3. THEORETICAL FORMULATION

Let an expression F of a given function f comprise the non–zero cubes $c_1, \ldots, c_n$:

$$F = c_1 \oplus c_2 \oplus \cdots \oplus c_n \tag{3}$$

If $d(c_i, c_j) = 0$ for some i, j then these cubes can be removed from the cover. The following straightforward lemma states that two cubes c_i, c_j can be replaced by a single cube c if and only if $d(c_i, c_j) = 1$.

LEMMA 1. *If the distance $d(c_i, c_j)$ between two cubes c_i, c_j is greater than 1, then there does not exist cube c such as $c_j \oplus c_i = c$.*

THEOREM 1. *All exact covers of a function f can be derived from its subfunction f^j iff subfunction f^i is zero, with $j = 1, 0, 1$ for $i = 0, 1, 2$ respectively.*

PROOF. *From (2), when one subfunction f^i of f is zero, then the other two subfunctions are equal and f can be written as:*

$$f = x_1^e f^j \tag{4}$$

with $e = 1, 0, 2$ and $j = 1, 0, 1$ for $i = 0, 1, 2$ respectively.

Subfunction f^j does not depend on variable x_1. Multiplying an ESOP g of f^j by x_1 in any polarity, does not increase or decrease the number of product terms in g. From (4) we conclude that $w(f) = w(f^j)$ hence the theorem is proved. □

The following theorem provides a systematic way of detecting all possible exact expressions of a function f with $w(f) = 2$, given one of its exact expressions.

THEOREM 2. *Let two cubes c_i, c_j such as $d(c_i, c_j) > 1$ and $g = c_i \oplus c_j$. Also suppose that the literals of variable x_1 in cubes c_i, c_j are different. Then all exact expressions of g can directly be obtained from subfunction pairs g^a, g^b of g with $w(g^a) = w(g^b) = 1$.*

PROOF. *Since $d(c_i, c_j) > 1$, by Lemma 1, $w(g) = 2$. Function g can be written as:*

$$g = x_1^k g^a \oplus x_1^l g^b$$

where $k, l, a, b \in \{0, 1, 2\}$ are obtained from (2). Since $k \neq l$ and $w(g) = 2$, then $w(g^a) = w(g^b) = 1$. □

It is observed that, by Theorem 2, function g can have either only 1 cover, when one of its subfunctions has weight equal to 2, or 3 distinct covers, when all three subfunctions have weight equal to 1.

The following theorem constrains the weight of a function g comprising three arbitrary cubes of (3) to $w(g) = 3$, under the stated assumptions.

THEOREM 3. *$\forall i, j \in \{1, \ldots, n\}$ such as $i \neq j$, let*

$$d(c_i, c_j) > 1$$

and $\forall p_1, p_2$ such as $c_i \oplus c_j = p_1 \oplus p_2$, let

$$d(c_k, p_l) > 1$$

for $k \in \{1, \ldots, n\} \setminus \{i, j\}$ and $l \in \{1, 2\}$.

Then $\forall c_m$ such as $g = c_i \oplus c_j \oplus c_m$ it holds:

$$w(g) = 3$$

PROOF. *Since function g comprises three non–zero cubes, its weight cannot be zero. Moreover, $w(g) \leq 3$ since $c_i \oplus c_j \oplus c_k$ is an expression of g. It remains to be proved that $w(g) \neq 1, 2$.*

Let $G = x_1^a G^A \oplus x_1^b G^B \oplus x_1^c G^C$ be an expression of g, where $a, b, c \in \{0, 1, 2\}$ and subexpressions G^A, G^B, G^C do not depend on x_1.

If $a = b = c$ then G can be written as

$$G = x_1^a (G^A \oplus G^B \oplus G^C)$$

hence the weight of g is equal to $w(G^A \oplus G^B \oplus G^C)$, by Theorem 1.

Let one of a, b, c be different, for example $a \neq b = c$. Then G can be written as

$$G = x_1^a G^A \oplus x_1^b (G^B \oplus G^C) \tag{5}$$

Then (5) is of the form of (2) and $G^A, (G^B \oplus G^C)$ are expressions of subfunctions of g, for example $g^K = G^A, g^L = G^B \oplus G^C$.

Let us assume that $w(g) = 2$. Then the weights $(w(g^K), w(g^L))$ can be either $(1, 2)$ or $(1, 1)$. In the case of $(1, 2)$, in order for $w(g) = 2$ to hold, G^A must be equal to one of the cubes C in the possible exact expressions of g^L. Therefore $d(x_1^a G^A, x_1^b C) = 1$, a contradiction. In the case of $(1, 1)$, $d(x_1^b G^B, x_1^b G^C) = 1$, a contradiction.

Let us assume that $w(g) = 1$. Then the weights $(w(g^K), w(g^L))$ can additionally be $(1, 0)$, in which case $d(x_1^b G^B, x_1^b G^C) = 0$, a contradiction.

In the case where a, b, c are pairwise different, G can be written as

$$G = x_1^a (G^A \oplus G^C) \oplus x_1^b (G^B \oplus G^C)$$

Then the weights $(w(g^K), w(g^L))$ can additionally be $(2, 2)$ or $(2, 1)$. The case of $(2, 1)$ is similar to the case of $(1, 2)$. In the case of $(2, 2)$ one cube C_1 in one of the exact expressions of g^K must be equal to one cube C_2 in one of the exact expressions of g^L. Therefore, $d(x_1^{a'} C_1, x_1^{b'} C_2) < 2$, a contradiction. □

The above theorem can be utilized to guarantee that no three-cube expression in a particular cover can have weight less than three if the preconditions are satisfied. The following theorem provides a way to reconstruct all possible covers of a function g with $w(g) = 3$. The proof of a generalization of this is presented in [12].

THEOREM 4. *Let a function g with $w(g) = 3$. All exact expressions of g can directly be obtained from the exact expressions of its subfunctions.*

4. ALGORITHM DESCRIPTION

Although the theoretical analysis provided in the previous section may seem somewhat involved, the actual algorithm is conceptually quite simple.

An initial cover F is generated from a given boolean function f. The goal is to gradually minimize F by iteratively selecting a number of cubes of it and transform them appropriately.

There are numerous alternatives for generating an initial cover. A straightforward example is the minterm cover, which is the exclusive–or sum of the minterms of f, or the

Algorithm 1: Minimizer $(f, quality)$

```
begin
    F=Generate Pseudo–Kronecker Cover (f);
    loop=quality;
    while loop > 0 do
        s=Size (F);
        Eliminate Distance 01 Cubes (F);
        Perform All 2to2 (F);
        Shuffle (F);
        if s = Size(F) then
            loop=loop-1;
        else
            loop=quality;
        endif
    endw
end
```

Algorithm 2: Find 2to2 (c_1, c_2)

```
begin
    Fix an ordering x_1, x_2, ..., x_n of the variables in
    c_1, c_2 such as x_1, ..., x_k appear in same literal form
    (k ≥ 0);
    Form P = common part of c_1, c_2;
    Form g = c'_1 ⊕ c'_2, where c'_1, c'_2 are equal to c_1, c_2
    with P removed;
    Find g^1, g^0, g^2 according to (1);
    Find w(g^1), w(g^0), w(g^2) according to Lemma 1;
    if max(w(g^1), w(g^0), w(g^2)) = 2 then
        return [(c_1 ⊕ c_2)];
    endif
    return [(P x_{k+1} g^1 ⊕ P x̄_{k+1} g^0),
    (P x_{k+1} g^2 ⊕ P g^0),
    (P x̄_{k+1} g^2 ⊕ P g^1)];
end
```

pseudo–Kronecker cover that is selected for the proposed algorithm.

The pseudo–Kronecker cover of a function f can be obtained recursively by the pseudo–Kronecker covers of its subfunctions f^1, f^0, f^2, by simply selecting two of the subfunctions with the minimum (pseudo–Kronecker) size and forming a cover for f according to (2). Pseudo–Kronecker covers, also adopted in [4], are experimentally observed to have cover sizes not much larger than the weight $w(f)$ of f.

The cover is maintained as a list of cubes. Each cube is maintained within a 64–bit vector. Every two bits represent the three possible states of a single boolean variable, thus limiting the applicability of the current implementation to functions that depend on at most 32 variables.

For a fixed number of repetitions, perform the following steps: Scan F for all distance–0 or distance–1 cubes and either remove them or replace them with a single cube respectively.

Subsequently, select two cubes c_i, c_j and obtain all alternative expressions for them, according to Theorem 2. For each of these expressions $c_1 \oplus c_2$, examine whether c_1, c_2 have distance less than 2 with a cube c of the rest cover, for example $d(c_1, c) = 1$. If such a cube is found, replace c_i, c_j with c_1, c_2 and then c_1, c with the single cube $c' = c_1 \oplus c$.

After all $2 \rightarrow 2$ reductions have been performed, perform a shuffling of the cover. This is realized in two substeps. First, the cubes in F are randomly permuted. Afterwards, the cover is partitioned in groups of three cubes, possibly ignoring one or two cubes.

All exact expressions of each group are obtained, and one of them is randomly selected to replace the original three cubes of the group in the cover. By the $2 \rightarrow 2$ step and Theorem 3, the function represented by each group has weight equal to three.

The process is repeated for a fixed number of repetitions during which the size of the cover has not diminished. The core of the algorithm is depicted in Algorithm 1. The algorithms to obtain all $2 \rightarrow 2$ and $3 \rightarrow 3$ transformations are presented in Algorithm 2 and Algorithm 3 respectively.

5. EXPERIMENTAL RESULTS

The performance of the algorithm was measured for a number of benchmark functions, decomposed to single – output functions where applicable, as well as on randomly generated functions, under an AMD Athlon XP 1.8GHz based system with 256MB of RAM.

The results were compared to a publicly available implementation of Exorcism4 [4], in terms of algorithm execution times and resulting cover size and are depicted in Tables 1 and 2.

In table 1, execution times of Exorcism4 are omitted for the benchmark functions for which its solution was worse than that of the proposed algorithm. In these cases, [4] did not manage to reduce the size of the cover further after executing for one minute. For the random functions case, sets of 25 n–variable functions were generated, for $n = 5, \ldots, 9$.

The quality of the solutions is observed to match and in some cases, surpass [4]. Execution times are of the order of tens of milliseconds in most cases and are also comparable with [4]. It is noted that there exist functions where Exorcism4 converges faster that the proposed algorithm, since the latter lacks the excellent bail-out heuristics of [4].

6. CONCLUDING REMARKS

In this work, a single–output heuristic ESOP minimization algorithm was presented, based on novel cube transformations. By experimental results on random and bench-

Function Name	Solution Size		Execution Time (ms)	
	Proposed	[4]	Proposed	[4]
9sym	51	52	481	
clip:1	16	16	29	105
clip:2	18	18	30	91
clip:3	21	21	32	43
clip:4	27	27	67	65
clip:5	15	15	31	41
t481	13	13	34	46
life	46	47	4962	
xor5	5	5	30	34
ryy6	40	40	29	30

Table 1: Solution Size and Execution Times on Benchmark Functions

Algorithm 3: Find 3to3 (c_1, c_2, c_3)

begin
Fix an ordering $x_1, \ldots, x_n$ of the variables in c_1, c_2, c_3 such as $x_1, \ldots, x_k$ appear in same literal form $(k \geq 0)$;
Form P = common part of c_1, c_2, c_3;
Form $g = c_1' \oplus c_2' \oplus c_3'$, where c_1', c_2', c_3' are equal to c_1, c_2, c_3 with P removed;
$ResultList = []$;
Find g^1, g^0, g^2 according to (1);
Find $w(g^1), w(g^0), w(g^2)$ according to Lemma 1 and Theorem 3;
Find all exact expressions of g^1 if $w(g^1) < 3$;
Find all exact expressions of g^0 if $w(g^0) < 3$;
Find all exact expressions of g^2 if $w(g^2) < 3$;
Choose all function pairs (g^A, g^B), such as $(w(g^A), w(g^B))$ is equal to $(2,2)$ or $(2,1)$ only;
for *each such pair* **do**
Let the corresponding literals according to (2) be x^a_{k+1}, x^b_{k+1}, such as $g = x^a_{k+1} g^A \oplus x^b_{k+1} g^B$, and x^c_{k+1} be the third form of the literal;
if $(w(g^A), w(g^B)) = (2,2)$ **then**
for *all expressions* $(c_{A1} \oplus c_{A2})$ *of* g^A **do**
for *all expressions* $(c_{B1} \oplus c_{B2})$ *of* g^B **do**
if $c_{A1} = c_{B1}$ **then**
$ResultList = ResultList+$
$(Px^a_{k+1} c_{A2} \oplus Px^b_{k+1} c_{B2}$
$\oplus Px^c_{x+1} c_{A1})$;
else if $c_{A1} = c_{B2}$ **then**
$ResultList = ResultList+$
$(Px^a_{k+1} c_{A2} \oplus Px^b_{k+1} c_{B1}$
$\oplus Px^c_{x+1} c_{A1})$;
else if $c_{A2} = c_{B1}$ **then**
$ResultList = ResultList+$
$(Px^a_{k+1} c_{A1} \oplus Px^b_{k+1} c_{B2} \oplus$
$Px^c_{x+1} c_{A2})$;
else if $c_{A2} = c_{B2}$ **then**
$ResultList = ResultList+$
$(Px^a_{k+1} c_{A1} \oplus Px^b_{k+1} c_{B1} \oplus$
$Px^c_{x+1} c_{A2})$;
endfor
endfor
else
for *all expressions* $(c_{A1} \oplus c_{A2})$ *of* g^A **do**
$ResultList = ResultList+$
$(Px^a_{k+1} c_{A1} \oplus Px^a_{k+1} c_{A2} \oplus Px^b_{k+1} g^B)$;
if $d(c_{A1}, g^B) = 1$ **then**
$c' = c_{A1} \oplus g^B$;
$ResultList = ResultList+$
$(Px^a_{k+1} c_{A2} \oplus Px^b_{k+1} c' \oplus Px^c_{k+1} c_{A1})$;
else if $d(c_{A2}, g^B) = 1$ **then**
$c' = c_{A2} \oplus g^B$;
$ResultList = ResultList+$
$(Px^a_{k+1} c_{A1} \oplus Px^b_{k+1} c' \oplus Px^c_{k+1} c_{A2})$;
endfor
endif
endfor
return $ResultList$;
end

n	Average Initial Cover	Average Proposed	Average Exorcism4
5	5.72	5.08	5.08
6	10.76	8.60	8.64
7	18.88	13.88	14.00
8	37.16	25.40	25.76
9	77.20	51.16	51.56

Table 2: Solution Size for Random Functions

mark functions, the quality of the resulting cover matches and, in some cases, surpasses the state of the art in heuristic ESOP minimization. Moreover, algorithm execution times are of the order of tenths of milliseconds in most cases and also compare favorably with previous published work. Future work will focus on the generalization of the algorithm to multiple–output and incompletely specified boolean functions.

7. REFERENCES

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

[2] A. Gaidukov. Algorithm to derive minimum esop for 6–variable function. In *5th International Workshop on Boolean Problems*, September 2002.

[3] T. Hirayama, T. Sato, and Y. Nishitani. Minimizing and-exor expressions of some benchmark functions. In *6th International Workshop on Applications of the Reed Muller Expansion in Circuit Design*, March 2003.

[4] A. Mishchenko and M. Perkowski. Fast heuristic minimization of exclusive-sums-of-products. In *5th International Workshop on Applications of the Reed Muller Expansion in Circuit Design*, August 2001.

[5] G. Papakonstantinou. Minimization of modulo-2 sum of products. *IEEE Transaction on Computers*, 28(2):163–167, 1979.

[6] G. Papakonstantinou. Minimal modulo-2 expressions of switching functions with five variables. *International Journal of Electronics*, 50(3):211–214, 1981.

[7] T. Sasao. An exact minimization of and-exor expressions using reduced covering functions. In *Proc. of the Synthesis and Simulation Meeting and International Interchange*, pages 374–383, October 1993.

[8] T. Sasao. An exact minimization of and-exor expressions using reduced covering functions. In *Proc. of the Synthesis and Simulation Meeting and International Interchange*, pages 374–383, October 1993.

[9] T. Sasao. Exmin2: A simplification algorithm for exclusive-or sum-of-products expressions for multiple-valued input two-valued output functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(5):621–632, May 1993.

[10] N. Song and M. Perkowski. Minimization of exclusive sum of products expressions for multiple-valued input incompletely specified functions. *IEEE Transactions on Computer Aided Design of Circuits and Systems*, 15(4):385–395, April 1996.

[11] S. Stergiou and G. Papakonstantinou. An efficient algorithm for exact esop minimization. In *The 2002 International Conference on VLSI*, June 2002.

[12] S. Stergiou and G. Papakonstantinou. Towards a general novel exact esop minimization methodology. In *6th International Workshop on Applications of the Reed Muller Expansion in Circuit Design*, March 2003.

[13] S. Stergiou and G. Papakonstantinou. Exact minimization of esop expressions with less than eight product terms. *Journal of Circuits, Systems, and Computers*, February 2004.