Università di Pisa
Dipartimento di Informatica

Technical Report: TR–08–17

An Approximation Algorithm for Generalized EXOR Projected Sum of Products

Anna Bernasconi,
Department of Computer Science
University of Pisa
56100 Pisa, Italy
annab@di.unipi.it

Valentina Ciriani, Roberto Cordone
Department of Information Technologies
University of Milano
26013 Crema (CR), Italy
{ciriani, cordone}@dti.unimi.it

ADDRESS: Largo B. Pontecorvo 3, 56127 Pisa, Italy. TEL: +39 050 2212700 FAX: +39 050 2212726

An Approximation Algorithm for Generalized EXOR Projected Sum of Products

Anna Bernasconi,
Department of Computer Science
University of Pisa
56100 Pisa, Italy
annab@di.unipi.it

Valentina Ciriani, Roberto Cordone
Department of Information Technologies
University of Milano
26013 Crema (CR), Italy
{ciriani, cordone}@dti.unimi.it

Abstract

Fast minimization time, compact area and low delay are important issues in logic circuit design. In order to orchestrate these main goals, in this paper we propose a new four level logic form (*Generalized EXOR-Projected Sum of Products*, in short *GEP-SOP*) with low delay and compact area. Even if the problem of finding an optimal GEP-SOPs is computationally hard, we propose an efficient approximation algorithm that gives guaranteed near optimal solutions. A wide set of experimental results confirms that the GEP-SOP forms are often more compact than the SOP forms, and their synthesis is always a very fast reoptimization phase after SOP minimization.

1 Introduction

Binary Decision Diagrams (BDDs) are widely used in digital-system design for verification and synthesis [1]. A BDD for a Boolean function can be directly mapped onto a network of multiplexers, called *BDD-circuit*, as shown in [2]. Such circuits are quite compact for a large range of functions, however their depth can be equal to the number of primary inputs, thus producing high delays. An interesting extension to BDDs has been proposed in [3], called *Linearly-Transformed BDDs*, in short *LTBDDs*. LTBDDs are binary decision diagrams whose nodes are labeled with EXORs of variables, instead of just single variables. Each node in a LTBDD thus represents a sort of Shannon decomposition based on an EXOR of variables. We can observe that even LTBDDs could be mapped into circuits, using multiplexers combined with EXOR gates. In this way, we could have more compact circuits, whose depth, however, can still be equal to the number of input variables.

EXOR based Shannon decomposition has also been considered in [4] and [5], where an algebraic form for Boolean function representation, called *EXOR-Projected Sum of Products* (*EP-SOP*) has been defined. This form contains a single two input EXOR gate (2-EXOR) two AND gates and an OR, combined with two SOP expressions. The advantage of EP-SOP circuits is that their depth is bounded, since the number of logic levels is equal to four. Moreover, circuits with a bounded number of levels, three or four ([6, 7, 8, 9, 10, 11, 12]), often result to be much more compact in area than classical two level logic forms ([13, 14]).

Guided by the observation that LTBDDs, containing several EXOR gates, are much compact in size, in this paper, we generalize EP-SOP networks, introducing more EXOR gates, while keeping the number of levels equal to four. We call these new expressions *Generalized EP-SOPs*, in short *GEP-SOP forms*. From a mathematical point of view, a GEP-SOP can be seen as a projection of an initial SOP ϕ onto a set of disjoint subspaces of the Boolean space $\{0,1\}^n$, each represented by a product of 2-EXORs. The overall network is composed by the SOP forms representing the projections of ϕ onto each subspace, the ANDs of EXOR gates describing the projected subspaces, and a final OR gate, as shown in Figure 3.

Even if the problem of finding an optimal GEP-SOP representation of a Boolean function is computationally difficult, we propose a very efficient approximation algorithm that guarantees a constant approximation ratio, and therefore a near optimal solution (see Section 3). This algorithm has been implemented and tested with interesting results (see Section 4), showing how several standard Espresso benchmarks benefit from

our approach. Starting from minimal SOP forms, these compact bounded level networks can be obtained in really limited computational time. Therefore, evaluating GEP-SOP forms as an alternative to optimal SOPs appears to be an advisable reoptimization step to be performed after the SOP synthesis.

2 Generalized EP-SOP forms

In this section we first briefly recall the definition of EP-SOP forms from [4], and then introduce the new concept of *Generalized EP-SOP forms*.

2.1 EP-SOP Forms

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function depending on n variables x_1, x_2, ..., x_n, and let ϕ be a SOP representation of it. Let us consider a couple of variables x_i and x_j, where, without loss of generality, $i < j$. The space $\{0,1\}^n$ can be partitioned into two disjoint subspaces: the space where $x_i \neq x_j$ or, equivalently, $x_i = \overline{x}_j$, defined by the characteristic function $(x_i \oplus x_j)$, and its complementary subspace where $x_i = x_j$, defined by the function $(x_i \oplus \overline{x}_j)$.

The main idea of EP-SOP forms is that of representing the function f as the sum (union) of the two projections of ϕ, $\phi_{i\overline{j}}$ and ϕ_{ij}, onto these two spaces:

$$\text{EP-SOP}_{ij}(f) = (x_i \oplus x_j)\phi_{i\overline{j}} + (x_i \oplus \overline{x}_j)\phi_{ij}\,.$$

After the projection, one can further minimize $\phi_{i\overline{j}}$ and ϕ_{ij}, in order to minimize the overall form. Projecting a product onto the space $x_i = x_j$ (resp., $x_i \neq x_j$) simply means to substitute in it the occurrence of x_i, if any, with x_j (resp., $\overline{x}_j$). Observe that all products that do not depend on x_i are not modified by the projection. Moreover, the substitution of x_i with x_j in a product depending on both x_i and $\overline{x}_j$ produces an empty product.

For example, let a SOP representation for Boolean function f be $\phi = \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 + \overline{x}_2\overline{x}_3\overline{x}_4 + x_5$. Consider the couple of variables x_1 and x_2. The projection of f onto the space $x_1 \neq x_2$ is covered by the SOP form $x_2\overline{x}_2x_3 + x_2x_2\overline{x}_3 + \overline{x}_2\overline{x}_2\overline{x}_3 + \overline{x}_2\overline{x}_3\overline{x}_4 + x_5$, which can be simplified to $x_2\overline{x}_3 + \overline{x}_2\overline{x}_3 + \overline{x}_2\overline{x}_3\overline{x}_4 + x_5$. Analogously, the projection of ϕ onto the space $x_1 = x_2$ is covered by $\overline{x}_2x_3 + \overline{x}_2\overline{x}_3\overline{x}_4 + x_5$. We can further minimize the first form because minterms, which were not adjacent in the original SOP form, now merge into new larger prime cubes. The EP-SOP representation of f is then $f \equiv (x_1 \oplus x_2)\,(\overline{x}_3 + x_5) + (x_1 \oplus \overline{x}_2)(\overline{x}_2x_3 + \overline{x}_2\overline{x}_3\overline{x}_4 + x_5)$.

Observe that the projections allow to eliminate one of the two variables x_i and x_j, and let us suppose to always remove the one with lower index, x_i. Thus, $\phi_{i\overline{j}}$ and ϕ_{ij} only depend on $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$, and their minimization should be easier. Moreover, thanks to the presence of EXOR gates, the projections can reduce the Hamming distances among the cubes appearing in each subspace, so that further merges can be possibly performed, even when the initial projected SOP ϕ for f is minimal. For these reasons, one could expect EP-SOP forms to be more compact than SOP forms, as the experimental results shown in [4] have partly confirmed.

It was also observed in [4] that it is not always convenient to project all products of ϕ. Indeed, while some products are entirely included into one of the two subspaces $x_i \neq x_j$ and $x_i = x_j$ (precisely, all products containing both the variables x_i and x_j, possibly complemented, in their algebraic expression), and will end up in just one projected SOP, other products, called *crossing products*, intersect both spaces producing two smaller products, which should be included one in $\phi_{i\overline{j}}$ and one in ϕ_{ij}. Therefore, one can choose to keep the crossing products in an unprojected SOP, named *remainder*. In this case, the resulting expression is called *EP-SOP with remainder*, in short *EPr-SOP*:

$$\text{EPr-SOP}_{ij}(f) = (x_i \oplus x_j)\phi_{i\overline{j}} + (x_i \oplus \overline{x}_j)\phi_{ij} + \rho\,,$$

where $\phi_{i\overline{j}}$ and ϕ_{ij} are the projections of the products of ϕ containing both x_i and x_j (possibly complemented) onto the spaces $(x_i \oplus x_j)$ and $(x_i \oplus \overline{x}_j)$, respectively, and ρ is the sum of all crossing products of ϕ.

For example, consider again the Boolean function f whose SOP representation is $\phi = \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 + \overline{x}_2\overline{x}_3\overline{x}_4 + x_5$. The projection with remainder of f onto the spaces $x_1 \neq x_2$ and $x_1 = x_2$ are covered by the SOP forms $x_2\overline{x}_3 + \overline{x}_2\overline{x}_3$, and $\overline{x}_2x_3$, respectively. The remainder is composed by the products

that do not contain both x_1 and x_2, i.e., $\rho = \overline{x}_2\overline{x}_3\overline{x}_4 + x_5$. The EPr-SOP representation of f is then $f \equiv (x_1 \oplus x_2)\,(\overline{x}_3) + (x_1 \oplus \overline{x}_2)(\overline{x}_2 x_3) + (\overline{x}_2\overline{x}_3\overline{x}_4 + x_5)$.

Finally, observe that instead of fixing a given couple of variables defining the projection subspaces, one could be interested in finding the *minimal EP-SOP representation* (with or without remainder) of a Boolean function, i.e., the expression containing the minimum number of products among all possible minimal EP-SOPs with respect to *any* couple of variables. Deriving these forms is computationally very hard, nevertheless a very efficient approximation algorithm has been proposed in [4].

2.2 GEP-SOP Forms

We now introduce a generalization of EP-SOP forms. The idea behind this generalization is simply that of further projecting the projected SOPs $\phi_{i\overline{j}}$ and ϕ_{ij}, and possibly the remainder, onto the subspaces defined by other couples of variables. The procedure can be further iterated, progressively reducing the variables in the projected SOPs, while increasing the variables defining the projection subspaces. We can define these new logic forms recursively as follows.

Definition 1 *Given a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ and a SOP ϕ for f, a* Generalized EXOR Projected Sum of Products form *(shortly,* GEP-SOP*) for f is either:*

- *a SOP form;*
- *an expression containing two GEP-SOP forms for the projections $\phi_{i\overline{j}}$ and ϕ_{ij} of ϕ onto the subspaces defined by $(x_i \oplus x_j)$ and $(x_i \oplus \overline{x}_j)$, where x_i and x_j are a given couple of variables:*

$$\begin{aligned} \textit{GEP-SOP}(f) &= (x_i \oplus x_j)\ \textit{GEP-SOP}(\phi_{i\overline{j}}) \\ &+ (x_i \oplus \overline{x}_j)\ \textit{GEP-SOP}(\phi_{ij})\,. \end{aligned}$$

For instance, given a function $f : \{0,1\}^n \rightarrow \{0,1\}$, a SOP ϕ for f, and three couples of variables (x_i, x_j), (x_h, x_k), (x_ℓ, x_m), with $i < j$, $h < k$, $\ell < m$, and $h, k, \ell, m \neq i$, we can consider the following GEP-SOP representation for f:

$$\begin{aligned} \text{GEP-SOP}(f) = (x_i \oplus x_j)[(x_h \oplus x_k)\phi_{i\overline{j},h\overline{k}} + (x_h \oplus \overline{x}_k)\phi_{i\overline{j},hk}] \\ +(x_i \oplus \overline{x}_j)[(x_\ell \oplus x_m)\phi_{ij,\ell\overline{m}} + (x_\ell \oplus \overline{x}_m)\phi_{ij,\ell m}]\,, \end{aligned}$$

where $\phi_{i\overline{j},h\overline{k}}$, $\phi_{i\overline{j},hk}$, $\phi_{ij,\ell\overline{m}}$, and $\phi_{ij,\ell m}$ are the projections of the initial SOP ϕ for f onto the four disjoint subspaces described by the linear systems of equations:

$$\left\{\begin{array}{l} x_i = \overline{x}_j \\ x_h = \overline{x}_k \end{array}\right. \quad \left\{\begin{array}{l} x_i = \overline{x}_j \\ x_h = x_k \end{array}\right. \quad \left\{\begin{array}{l} x_i = x_j \\ x_\ell = \overline{x}_m \end{array}\right. \quad \left\{\begin{array}{l} x_i = x_j \\ x_\ell = x_m \end{array}\right.$$

and whose characteristic functions are $(x_i \oplus x_j)(x_h \oplus x_k)$, $(x_i \oplus x_j)(x_h \oplus \overline{x}_k)$, $(x_i \oplus \overline{x}_j)(x_\ell \oplus x_m)$, and $(x_i \oplus \overline{x}_j)(x_\ell \oplus \overline{x}_m)$, respectively. The derivation of a GEP-SOP form for a function f can be represented in a very intuitive way using a binary tree, whose internal nodes are labeled with 2-EXORs representing the projection subspaces, and whose leaves represent the final projected SOPs, as shown in Figure 1. The root of the tree represents the first projection step applied to the starting SOP ϕ, the nodes on the first level represent the second projection step, that is the projections applied to $\phi_{i\overline{j}}$ and ϕ_{ij}, and so on.

The height of the tree representing a GEP-SOP thus is equal to the number of projection steps. Of course, one could perform more projections, replacing each leaf with another GEP-SOP.

The above definition can be extended in order to consider also the projections with remainder:

Definition 2 *Given a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ and a SOP ϕ for f, a* GEP-SOP form with remainder *(shortly,* GEPr-SOP*) for f is either:*

- *a SOP form for f;*

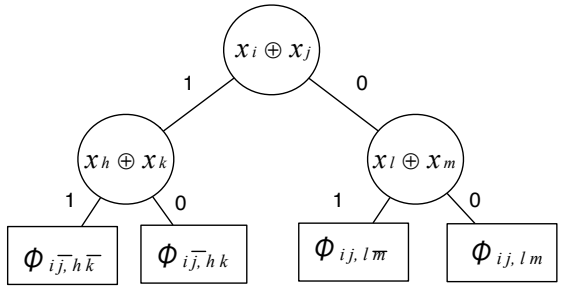

Figure 1: *A binary tree representing a GEP-SOP form.*

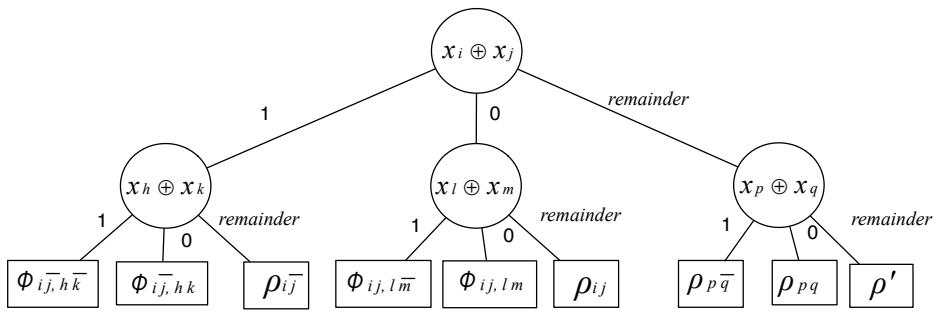

Figure 2: *A ternary tree representing a GEP-SOP form with remainder.*

- *an expression containing two GEPr-SOP forms for the projections with remainder $\phi_{i\overline{j}}$ and ϕ_{ij} of ϕ onto the subspaces defined by $(x_i \oplus x_j)$ and $(x_i \oplus \overline{x}_j)$, and a GEPr-SOP form for the remainder ρ, where x_i and x_j are a given couple of variables:*

$$\begin{aligned} \textit{GEPr-SOP}(f) &= (x_i \oplus x_j)\ \textit{GEPr-SOP}(\phi_{i\overline{j}}) \\ &+ (x_i \oplus \overline{x}_j)\ \textit{GEPr-SOP}(\phi_{ij}) \\ &+ \textit{GEPr-SOP}(\rho)\,. \end{aligned}$$

For instance, given a function f, a SOP ϕ for f, and four couples of variables (x_i, x_j), (x_h, x_k), (x_ℓ, x_m), (x_p, x_q), with $i < j$, $h < k$, $\ell < m$, $p < q$, and $h, k, \ell, m \neq i$, we can consider the following GEPr-SOP representation for f:

$$\begin{aligned} &\text{GEPr-SOP}(f) = (x_i \oplus x_j)[(x_h \oplus x_k)\phi_{i\overline{j},h\overline{k}} + (x_h \oplus \overline{x}_k)\phi_{i\overline{j},hk} \\ &\quad +\rho_{i\overline{j}}] + (x_i \oplus \overline{x}_j)[(x_\ell \oplus x_m)\phi_{ij,\ell\overline{m}} + (x_\ell \oplus \overline{x}_m)\phi_{ij,\ell m} + \rho_{ij}] \\ &+(x_p \oplus x_q)\rho_{p\overline{q}} + (x_p \oplus \overline{x}_q)\rho_{pq} + \rho'\,, \end{aligned}$$

where $\phi_{i\overline{j},h\overline{k}}$, $\phi_{i\overline{j},hk}$, $\phi_{ij,\ell\overline{m}}$, $\phi_{ij,\ell m}$ are the projections with remainder of the initial SOP ϕ onto the four disjoint subspaces $(x_i \oplus x_j)(x_h \oplus x_k)$, $(x_i \oplus x_j)(x_h \oplus \overline{x}_k)$, $(x_i \oplus \overline{x}_j)(x_\ell \oplus x_m)$, and $(x_i \oplus \overline{x}_j)(x_\ell \oplus \overline{x}_m)$. Moreover, $\rho_{i\overline{j}}$ and ρ_{ij} represent the sums of all crossing products of $\phi_{i\overline{j}}$ and ϕ_{ij}, respectively. Finally, ρ' is the remainder left by the projections of the initial remainder ρ onto the spaces $(x_p \oplus x_q)$ and $(x_p \oplus \overline{x}_q)$.

As before, we can represent a GEP-SOP form with remainder in a very intuitive way using a tree. This time the tree will be a ternary tree, indeed we add a child to each internal node in order to represent the remainder, as shown in Figure 2. It is interesting to observe that, no matter how many times the projection procedure has been performed, the final overall logic networks representing GEP-SOP forms without and with remainder will contain four levels of logic, as shown in Figure 3. Finally, observe that we could also

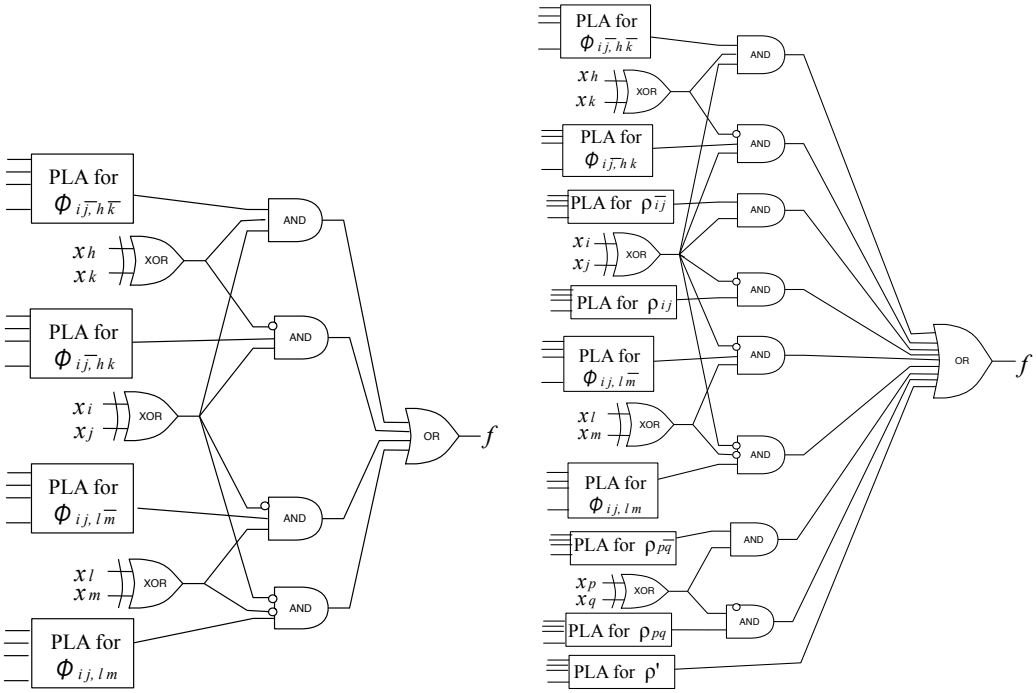

Figure 3: *GEP-SOP networks without remainder (left) and with remainder (right).*

consider "mixed" GEP-SOP forms, where at each projection step we can choose between a projection without or with remainder.

3 Approximation algorithm

The problem of finding a minimal GEP-SOP representation for a given Boolean function f is computationally very hard. Indeed, it contains EP-SOP minimization as a subproblem (when just one projection step is performed), and in [4] it was proved that deriving an EP-SOP form minimal with respect to a given couple of variables is an NP^{NP}-hard problem, even if the initial input SOP for f is already minimal. Recall that NP^{NP} is the class of all decision problems solved by a non-deterministic polynomial-time machine equipped with a non-deterministic polynomial time oracle. In the polynomial hierarchy [15], this class is a superset of NP.

Due to the complexity of the problem, we must give up exact GEP-SOP minimization for fast not exact polynomial algorithms, mirroring what has been done for EP-SOP minimization. There are two possible strategies for not exact minimization: heuristics and approximation algorithms. Both strategies do not guarantee the minimality of their solutions, but while we cannot perform any prediction on the result of a heuristic, an approximation algorithm gives guaranteed near-optimum solutions. In a minimization framework, an algorithm with approximation ratio p guarantees that the cost C of its solution is such that $C/C^* \leq p$, where C^* is the cost of an optimal solution [15]. We now describe a polynomial approximation algorithm for the problem of finding a minimal GEP-SOP representation of a function f starting from a *minimal SOP* ϕ for f (Section 3.1); in Section 3.2 we prove how this algorithm guarantees a constant approximation ratio.

3.1 Algorithm

Figure 4 shows recursive Algorithm 1 for computing a GEP-SOP, with and without remainder, with a given number of projection steps (s) starting from a SOP form. The first call of this algorithm is performed on

Algorithm 1 (Construction of a GEP-SOP)

First input: The number of projection steps s; a minimal SOP form ϕ for f (note that a SOP form is a particular GEPSOP); the boolean variable rem that is true only for generating GEP-SOP with remainder.
Final output: A GEP-SOP equivalent to ϕ (with remainder iff rem is true).

```
GEPSOP GepSop(step s, GEPSOP φ, boolean rem)
    if (s == 0)
        return Espresso non exact(φ);
    else /* (s > 0) */
        (x_i, x_j) = FindMostFrequentPair(φ);
        if (rem) /* GEP-SOP with remainder */
            (φ_ij, φ_i~j~, ρ) = ProjWithRem(x_i, x_j, φ);
            φ^G_ij = GepSop(s − 1, φ_ij, rem);
            φ^G_i~j~ = GepSop(s − 1, φ_i~j~, rem);
            ρ^G = GepSop(s − 1, ρ, rem);
            return (x_i ⊕ x_j)φ^G_ij + (x_i ⊕ x̄_j)φ^G_i~j~ + ρ^G ;
        else /* GEP-SOP without remainder */
            (φ_ij, φ_i~j~) = ProjWithoutRem(x_i, x_j, φ);
            φ^G_ij = GepSop(s − 1, φ_ij, rem);
            φ^G_i~j~ = GepSop(s − 1, φ_i~j~, rem);
            return (x_i ⊕ x_j)φ^G_ij + (x_i ⊕ x̄_j)φ^G_i~j~ ;
```

Figure 4: *Approximation algorithm for generating GEP-SOPs with and without remainder.*

a minimal SOP form ϕ of the function f. The function *FindMostFrequentPair* computes the most frequent pair of variables appearing in the given SOP. The two functions *ProjWithRem* and *ProjWithoutRem* compute the projections (and the remainder, in the first case) of ϕ with respect to the pair of variables x_i and x_j, as described in Section 2.1. The time complexity of this algorithm is polynomial in the size of the input SOP form, and it is exponential in the number of projection steps (s) required. In particular, the cost of the recursive calls is polynomial in the number of products in ϕ, and the algorithm performs $O(2^s)$ and $O(3^s)$ recursive calls for GEP-SOPs without and with remainder, respectively. Moreover, the cost of the case $s = 0$ is polynomial since Espresso non exact is used, and Algorithm 1 invokes this procedure $O(2^s)$ times for GEP-SOPs without remainder, and $O(3^s)$ times for GEP-SOPs with remainder. Finally observe that the number of projection steps s is then a critic parameter and should assume a low value.

For the sake of simplicity, hereafter we consider only GEP-SOP forms, with and without remainder, derived with only two projection steps ($s = 2$), as those represented by the trees in Figure 1 and Figure 2. Experimental results described in Section 4 show that even with only two projections we obtain good solutions in terms of area and computational time.

3.2 Approximation ratio

We now prove that the proposed synthesis strategy is indeed an approximation algorithm for the GEP-SOP minimization problem, restricted to forms derived with just two projection steps ($s = 2$). In this analysis, we will measure the cost of an algebraic form through the overall number of products contained in it. For instance, the cost of the GEP-SOPs represented in Figure 1 and Figure 2 are $|\text{GEP-SOP}(f)| = |\phi_{i\bar{j},h\bar{k}}| + |\phi_{i\bar{j},hk}| + |\phi_{ij,\ell\overline{m}}| + |\phi_{ij,\ell m}|$, and $|\text{GEPr-SOP}(f)| = |\phi_{i\bar{j},h\bar{k}}| + |\phi_{i\bar{j},hk}| + |\rho_{i\bar{j}}| + |\phi_{ij,\ell\overline{m}}| + |\phi_{ij,\ell m}| + |\rho_{ij}| + |\rho_{n\overline{p}}| + |\rho_{np}| + |\rho'|$, respectively, where we have denoted by $|\psi|$ the number of products in a given SOP ψ. In order to prove that the costs $|\text{GEP-SOP}(f)|$ and $|\text{GEPr-SOP}(f)|$ of our solutions are upper bounded by a constant times the costs of the optimal solutions $|\text{GEP-SOP}(f)^{MIN}|$ and $|\text{GEPr-SOP}(f)^{MIN}|$, respectively, we first find a lower bound for the optimal forms, and then an upper bound for the approximated solutions. We first consider GEP-SOPs without remainder.

Lemma 1 *Given a minimal SOP form ϕ for a Boolean function f, and a minimal GEP-SOP form, without remainder, for f, we have*

$$|\textit{GEP-SOP}(f)^{MIN}| \geq \frac{1}{4}|\phi|\,.$$

Proof. Suppose that an optimal GEP-SOP for f can be derived by projecting ϕ onto the four subspaces described by three particular couples of variables, and then minimizing the four projected SOPs. The cost of GEP-SOP$(f)^{MIN}$ is given by the overall number of products in these four SOPs. Now, observe that we can represent the characteristic functions of the four subspaces as sums of exactly four products; indeed they are given by the product of two EXORs of two literals. Thus, we can build a SOP for f multiplying each product in the four projected SOPs by the products in the SOP representing the corresponding projection subspaces. The SOP σ derived in this way has size $|\sigma| = 4|\text{GEP-SOP}(f)^{MIN}|$ and, since ϕ is minimal, we have $|\phi| \leq |\sigma|$, from which the thesis follows. ■

We now prove that if we project the starting minimal SOP ϕ with respect to the couples of variables selected by Algorithm 1, we get a solution whose approximation ratio is bounded by a constant. For a couple of variables x_i and x_j and a SOP ϕ, let us denote with ν_{ij} the number of products in ϕ containing both x_i and x_j, possibly complemented.

Theorem 1 *Let GEP-SOP(f) be the GEP-SOP representation of a Boolean function f computed by Algorithm 1, projecting a minimal SOP ϕ for f onto the four subspaces described by the couples of variables (x_i, x_j), (x_k, x_h) and (x_ℓ, x_m), and let GEP-SOP$(f)^{MIN}$ be a minimal GEP-SOP representation of f. Then*

$$\frac{|\textit{GEP-SOP}(f)|}{|\textit{GEP-SOP}(f)^{MIN}|} \leq 16 - \frac{4(2\nu_{ij} + \nu'_{hk} + \nu'_{\ell m})}{|\phi|}\,,$$

where ν_{ij} is the frequency of the couple (x_i, x_j) in the initial SOP ϕ, and ν'_{hk} and $\nu'_{\ell m}$ are the frequencies of the couples (x_h, x_k) and (x_ℓ, x_m) in the projections $\phi_{i\bar{j}}$ and ϕ_{ij} of ϕ onto $(x_i \oplus x_j)$ and $(x_i \oplus \overline{x}_j)$, respectively.

Proof. First of all observe that the two SOPs $\phi_{i\bar{j}}$ and ϕ_{ij} produced by the first projection of ϕ onto the spaces $(x_i \oplus x_j)$ and $(x_i \oplus \overline{x}_j)$ satisfy

$$|\phi_{i\bar{j}}| + |\phi_{ij}| = \nu_{ij} + 2(|\phi| - \nu_{ij}) = 2|\phi| - \nu_{ij}\,,$$

since the ν_{ij} products containing both x_i and x_j appear only once in one of the two projections, while the $(|\phi| - \nu_{ij})$ crossing products are projected onto both spaces, and appear in $\phi_{i\bar{j}}$ and ϕ_{ij}. Consider now the projection of $\phi_{i\bar{j}}$ onto $(x_h \oplus x_k)$ and $(x_h \oplus \overline{x}_k)$, resulting into the two new SOPs $\phi_{i\bar{j},h\bar{k}}$ and $\phi_{i\bar{j},hk}$. Using the same argumentation, we can conclude that $|\phi_{i\bar{j},h\bar{k}}| + |\phi_{i\bar{j},hk}| = \nu'_{hk} + 2(|\phi_{i\bar{j}}| - \nu'_{hk}|) = 2|\phi_{i\bar{j}}| - \nu'_{hk}$. Analogously, for the projection of ϕ_{ij} onto $(x_\ell \oplus x_m)$ and $(x_\ell \oplus \overline{x}_m)$, we have $|\phi_{ij,\ell\overline{m}}| + |\phi_{ij,lm}| = 2|\phi_{ij}| - \nu'_{\ell m}$. Thus, we get $|\text{GEP-SOP}(f)| = |\phi_{i\bar{j},h\bar{k}}| + |\phi_{i\bar{j},hk}| + |\phi_{ij,\ell\overline{m}}| + |\phi_{ij,lm}| = 2(|\phi_{i\bar{j}}| + |\phi_{ij}|) - \nu'_{hk} - \nu'_{\ell m} = 4|\phi| - 2\nu_{ij} - \nu'_{hk} - \nu'_{\ell m}$, and the thesis follows applying Lemma 1. ■

A better result holds for the GEP-SOP forms with remainder.

Theorem 2 *Let GEPr-SOP(f) be the GEP-SOP with remainder representation of a Boolean function f computed by Algorithm 1 from a minimal SOP ϕ for f, and let GEPr-SOP$(f)^{MIN}$ be a minimal GEP-SOP with remainder representation of f. Then*

$$\frac{|\textit{GEPr-SOP}(f)|}{|\textit{GEPr-SOP}(f)^{MIN}|} \leq 4\,.$$

Proof. First of all, observe that $|\text{GEPr-SOP}(f)^{MIN}| \geq |\phi|/4$. Indeed, we can derive a SOP from the expression GEPr-SOP$(f)^{MIN}$, as shown in the proof of Lemma 1, whose number of products is upper bounded by $4|\text{GEPr-SOP}(f)^{MIN}|$. Now observe that at each projection step all crossing products are kept unprojected, therefore no product in ϕ is projected onto more than one subspace. This implies that the overall number of products in the GEPr-SOP computed by Algorithm 1 is upper bounded by $|\phi|$. Therefore, $|\text{GEPr-SOP}(f)| \leq |\phi|$ and the thesis easily follows. ■

	SOP			GEP-SOP (s=1)			GEPr-SOP (s=1)			GEPSOP (s=2)			GEPr-SOP (s=2)		
Benchmark	**area**	**delay**	**CPU**	**area**	**delay**	**CPU**	**area**	**delay**	**CPU**	**area**	**delay**	**CPU**	**area**	**delay**	**CPU**
adr4	221	19.2	0.03	172	15.2	0.01	**104**	11.1	0.03	326	28	0.06	326	28	0.03
alcom	**217**	10.8	0.87	441	36.5	0.01	228	17.8	0.01	545	39.3	0.12	246	15.2	0.29
alu1	**53**	6.8	0.04	128	15.4	0.01	60	8.8	0.01	122	15	0.03	60	8.8	0.01
b2	3876	79.8	0.23	4113	81.3	0.06	4169	82.6	0.07	**3847**	76.3	0.21	3888	83.6	0.22
b4	**645**	30.5	2.89	802	33.3	0.01	717	34.4	0.01	798	33	0.13	708	31.6	0.17
b12	**167**	18.4	0.48	374	22.9	0.03	285	22.5	0.01	378	24.9	0.06	219	22.5	0.03
in2	1112	41.4	0.06	999	36.7	0.03	1001	37.3	0.01	1122	43.9	0.12	**968**	38.5	0.08
in5	**905**	38.5	0.12	976	39.2	0.01	923	40.9	0.01	1053	35.9	0.07	935	38.6	0.07
in7	366	23.8	0.57	432	25.9	0.01	**360**	25.8	0.01	415	25.9	0.05	395	27.6	0.06
log8mod	323	27.2	0.04	260	22.8	0.01	260	26.9	0.01	**242**	20.5	0.01	**242**	20.5	0.03
luc	806	41.0	0.04	779	52.8	0.01	**758**	52.4	0.01	863	49.3	0.02	837	52.2	0.05
m3	**839**	38.4	0.04	1286	48.4	0.03	1172	51.7	0.03	1019	38.2	0.08	1142	48.7	0.11
m181	**166**	18.4	0.56	327	22.4	0.03	240	22.5	0.03	338	24.3	0.03	220	22.5	0.02
mlp4	**734**	36.6	0.35	982	43.0	0.03	838	40.5	0.03	1179	50.6	0.08	1179	50.6	0.12
mp2d	362	26.0	0.29	427	25.3	0.03	**332**	23.7	0.01	383	22.1	0.06	389	23.8	0.10
newcwp	40	9.1	0.01	37	13.1	0.01	34	13.1	0.01	35	12.4	0.03	**32**	12.4	0.01
newxcpla1	**366**	21.4	0.04	731	35.1	0.03	619	31.5	0.01	731	36.4	0.07	488	30.3	0.08
p82	239	18.4	0.03	**238**	25.8	0.01	240	25.0	0.01	248	28.4	0.05	248	28.4	0.06
rckl	**341**	49.7	0.04	495	72.3	0.03	495	72.3	0.03	641	55.2	0.04	494	51.5	0.09
rd73	**217**	25.5	0.03	389	27.6	0.01	338	26.9	0.01	292	26.3	0.02	268	25.6	0.02
sqr6	**278**	25.5	0.06	397	27.0	0.01	330	24.9	0.01	418	28.8	0.04	418	28.8	0.04
tms	**587**	35.4	0.03	675	35.2	0.01	675	35.2	0.01	657	38.8	0.05	657	38.8	0.05
vg2	**341**	18.6	0.20	624	25.7	0.01	468	22.5	0.01	720	28.4	0.03	633	26.7	0.04
vtx1	**324**	21.3	0.17	441	25.5	0.01	365	23.4	0.01	437	23.5	0.03	364	27.2	0.02
x6dn	1053	36.8	0.15	854	34.9	0.01	817	34.8	0.01	**808**	34	0.06	851	36.6	0.02
x9dn	**384**	23.0	0.17	496	25.4	0.01	424	24.7	0.01	496	23.3	0.06	420	24.5	0.04
z4	168	18.3	0.01	157	18.6	0.01	**98**	14.2	0.01	334	27.6	0.01	334	27.6	0.03

Table 1: Comparison between the results of SOP and GEP-SOP forms after the technology mapping (area, delay and synthesis time in seconds).

4 Experimental Results and Conclusion

This section discusses the computational results obtained by building minimal GEP-SOP forms for a given Boolean function f. The aim of our experiments is twofold: we are interested in obtaining GEP-SOPs with compact area, to be compared with optimal SOP forms, while evaluating the practical performances of the proposed approximation algorithm. The results refer to the well known Espresso benchmark suite [16]. The first projection step is performed by projecting f with respect to the most frequent pair of variables in the optimal SOP form, either with or without a remainder. As most benchmarks have multiple outputs, we refer the definition of frequency to the whole set of outputs (*global frequency*), thus employing the same couple of variables for all outputs. This choice is based on the experimental results previously obtained on EP-SOPs in [4] and its rationale is to limit the number of EXOR gates in the final network, as the physical implementation of such gates can be expensive.

As our first aim is to obtain compact area networks, and thanks to the large degree of freedom allowed by the definition of GEP-SOP forms, in the experiments we followed a strategy slightly different from Algorithm 1. While at the first level the starting SOP form is always projected, in the second step the most frequent pair of variables in the whole set of resulting SOP components is selected, and each component is projected with respect to this couple only if the result yields a *more* compact expression. The rationale is once again to introduce few EXOR gates and only when they produce more compact expressions. After the projection, all SOP components are synthesized together with multi-output synthesis by Espresso non exact, in order to keep the worst-case complexity of the algorithm polynomial while exploiting product sharing.

We have evaluated the practical performance of five forms: optimal SOPs, GEP-SOPs, with and without remainder, obtained with one projection step (s=1), i.e., EP-SOP forms, and GEP-SOPs, with and without remainder, obtained with two projection steps (s=2).

We have used a technology mapping (*mcnc.genlib*) provided by the SIS tool [17] to evaluate the area and the delay corresponding to their physical implementation. Due to the limited space available, we report in Table 1 only a significant subset of the experiments. The first column reports the name of the instance considered. The following ones report, by groups of three, the area and the delay of the physical implementation of each form as estimated by SIS and the computational time in seconds required to synthesize it by our algorithms. The first group (labeled SOP) concerns the optimal SOP form computed by Espresso exact. The next two ones, labeled GEP-SOP (s=1) and GEPr-SOP (s=1), correspond to GEP-SOP forms obtained with just one projection step. The last two groups of columns, labeled GEP-SOP (s=2) and GEPr-SOP (s=2), regard the GEP-SOP forms obtained with two projection steps. The best result on each instance with respect to the area is in bold face. The computational time is measured in seconds and it is the time required to generate the form, that is to select the projection variables, to perform the projections and to

minimize by ESPRESSO NON EXACT the projected components.The experiments have been performed on a Pentium 2.4 GHz PC with 512 MB of RAM.

On slightly more than half of the instances, the SOP form has the lowest area. Almost half of the time, however, the GEP-SOP forms have a lower area, with approximately the same number of best results. The gain with respect to the optimal SOP form can be quite relevant: for example, the GEP-SOP (s=1) with remainder exhibits a 53% area gain on instance *adr4*; the GEP-SOPs (s=2) with and without remainder gain, respectively, 25% on instance *x6dn* and 23% on instance *newcwp*.

Since the time overhead required to obtain such improvements is very limited, the evaluation of the GEP-SOP forms as a possible alternative to the optimal SOP forms appears to be an advisable reoptimization after SOP minimization, in order to obtain alternative expressions for the given Boolean function f.

A promising direction for future research is the investigation of more refined strategies to build GEP-SOPs, i.e., to select the components to project and the projection variables, and to decide whether to leave or not a remainder.

References

[1] R. Bryant, "Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams," *ACM Computing Surveys*, vol. 24, no. 3, pp. 293–318, 1992.

[2] R. Drechsler, J. Shi, and G. Fey, "Synthesis of Fully Testable Circuits from BDDs." *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 23, no. 3, pp. 440–443, 2004.

[3] W. Günther and R. Drechsler, "Efficient Minimization and Manipulation of Linearly-Transformed Binary Decision Diagrams," *IEEE Trans. on Computers*, vol. 52, no. 9, pp. 1196–1209, 2003.

[4] A. Bernasconi, V. Ciriani, and R. Cordone, "EXOR Projected Sum of Products," in *14th International Conference on Very Large Scale Integration*, 2006.

[5] ——, "An Approximation Algorithm for Fully Testable kEP-SOP Networks," in *17th ACM Great Lakes Symposium on VLSI*, 2007.

[6] M. Perkowski, "A New Representation of Strongly Unspecified Switching Functions and its Application to Multi-Level AND/OR/EXOR Synthesis," in *IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion*, 1995, pp. 143–151.

[7] T. Sasao, "A Design Method for AND-OR-EXOR Three Level Networks," in *Int. Workshop on Logic Synthesis*, 1995, pp. 11–20.

[8] E. Dubrova and P. Ellervee, "A Fast Algorithm for Three-Level Logic Optimization," in *Int. Workshop on Logic Synthesis*, 1999, pp. 251–254.

[9] D. Debnath and T. Sasao, "Multiple–Valued Minimization to Optimize PLAs with Output EXOR Gates," in *IEEE International Symposium on Multiple-Valued Logic*, 1999, pp. 99–104.

[10] F. Luccio and L. Pagli, "On a New Boolean Function with Applications," *IEEE Transactions on Computers*, vol. 48, no. 3, pp. 296–310, 1999.

[11] D. Debnath and Z. Vranesic, "A Fast Algorithm for OR-AND-OR Synthesis," *IEEE Transactions on CAD*, vol. 22, no. 9, pp. 1166–1176, 2003.

[12] R. Ishikawa, T. Hirayama, G. Koda, and K. Shimizu, "New Three-Level Boolean Expression Based on EXOR Gates," *IEICE Trans. on Information and Systems*, no. 5, pp. 1214–1222, 2004.

[13] P. McGeer, J. Sanghavi, R. Brayton, and A. Sangiovanni-Vincentelli, "ESPRESSO-SIGNATURE: A New Exact Minimizer for Logic Functions," *IEEE Transactions on VLSI*, vol. 1, no. 4, pp. 432–440, 1993.

[14] O. Coudert, "Two-Level Logic Minimization: an Overview," *INTEGRATION*, vol. 17, pp. 97–140, 1994.

[15] M. Garey and D. Johnson, *Computer and Intractability: A Guide to the Theory of NP-completeness.* W.H. Freeman and C., 1979.

[16] S. Yang, "Logic synthesis and optimization benchmarks user guide version 3.0," Microelectronic Center," User Guide, 1991.

[17] E.M. Sentovich et al., "SIS: A system for sequential circuit synthesis," Tech. Rep., 1992.