Bi-Decomposition using Boolean Relations

Anna Bernasconi
Dipartimento di Informatica
Università di Pisa, Italy
anna.bernasconi@unipi.it

Robert K. Brayton
Department of EECS
University of California, Berkeley, USA
brayton@berkeley.edu

Valentina Ciriani
Dipartimento di Informatica
Università degli Studi di Milano, Italy
valentina.ciriani@unimi.it

Gabriella Trucco
Dipartimento di Informatica
Università degli Studi di Milano, Italy
gabriella.trucco@unimi.it

Tiziano Villa
Dipartimento di Informatica
Università degli Studi di Verona, Italy
tiziano.villa@univr.it

***Abstract*—We study three-level implementations where the first two levels represent a standard PLA form with an AND-plane and an OR-plane. This implements a 2*m*-output SOP. The final stage consists of *m* two-input programmable LUTs. The PLA outputs are paired so that the LUT outputs implement a set of *m* given incompletely specified functions (ISFs). Three-level structures have been studied previously, e.g. resulting in AND-OR-AND or AND-OR-XOR implementations. By using the LUT effectively, the composition of the AND-plane can be controlled to implement a PLA which has the optimum phase assignment for maximum cube sharing. For each output, we characterize the problem of all legal implementations of such a model, by defining Boolean relations that capture all the flexibility induced by the final LUT logic. The extra LUT level provides a dimension beyond simple phase assignment. We performed experiments using a Boolean relation minimizer to compare such realizations vs. SOP forms and published three-level forms, comparing areas and delays. To approximate the possible sharing in the PLA, we mapped the 2*m* PLA logic using SIS. We focused on experiments with two-input Boolean functions not captured by AND-OR-AND or AND-OR-XOR approaches and found good gains in many cases with affordable increases in synthesis runtimes.**

I. Introduction

Functional decomposition rewrites a logic function $f(X)$ as the composition of a set of functions $h, g_1, g_2, \ldots, g_m$ such that $f(X) = h(g_1(X), g_2(X), \ldots, g_m(X))$ ([16], [22]). It is a fundamental tool in logic synthesis for multi-level and FPGA implementations, and in the theory of circuit and communication complexity. The simplest case when $m = 2$, called bi-decomposition, appears often in the literature, because it fits quite well with the usual representations of logic as networks of 2-input gates, and if applied recursively generates more general decompositions ([21], [14]). Since the decomposition is top down and performed at the output level for each primary output separately, by bi-decomposition we transform initial two-level forms into three-level forms, about which there is an extensive literature for comparison. Afterwards, the remaining logic may be handled by standard synthesis methods, like two-level minimization of the two blocks followed by technology mapping onto a given implementation library.

In this work we study three-level forms, aiming at implementing incompletely specified functions (ISFs), $f = (f_{on}, f_{off})$, using a cover $c = u\ op\ v$, where op is a two input gate or LUT, and u and v are two SOP forms. For an example with $op(u, v) = \overline{u}v$, see the scheme depicted in Figure 2.

For an ISF f represented by its on-set f_{on}, dc-set f_{dc}, and off-set f_{off}, a cover g is a completely specified function such that $f_{on} \subseteq g \subseteq f_{on} \cup f_{dc}$. A cover h of its complement ISF, $f_{off} \subseteq h \subseteq f_{off} \cup f_{dc}$, may lead to a more optimal implementation.

Example: Consider $op(u, v) = \overline{u} + v = (u \Rightarrow v)$. We have the flexibility to implement a minterm in f_{on} in one of three ways: by adding it to both the cover v and the cover u (the output of the latter is negated and becomes a 0 input to the OR gate), or by adding it to the cover v and by not adding it to the cover u (i.e., it is put in $\overline{u}$ so that the 0 output of u is negated and becomes a 1 input to the OR gate), or by not adding it to the cover v and not adding it to the cover u (whose negated output inputs a 1 into the OR gate). In other words, a point in the onset can be realized by v only, or by v and $\overline{u}$, or by $\overline{u}$ only.

Intuitively, a LUT (or gate) at the output provides a generalization of the problem of choosing the best output phase assignment in the realization of a $2m$ SOP (see [17], [18]). With a LUT at the output, we can choose the best phase assignment for the SOP feeding the LUT where the SOP has $2m$ outputs. Even if the $2m$ outputs were implemented as a Boolean network, there is still an interesting phase assignment because we can choose a cover of the ISF f or a cover of its complement ISF. The example also illustrates the case where a two-input operator connecting two logic blocks u and v induces don't care conditions: e.g., when $op = OR$, if a point in the on-set is added to v then we do not care if it is added also to u, yielding the don't care condition (-1) for the two outputs u and v; dually, the same outputs can assume values in the cube $(1-)$, since if an on-set point is added to u we do not care if it is part of v. However, the two cubes $(1-)$ and (-1) cannot be expressed by a single cube (which would mean that it could be expressed as a don't care), but instead a Boolean relation is required to model this flexibility. Therefore, the problem of optimizing the implementation of a decomposition of f in the form $c = u\ op\ v$, is one of defining and optimizing a Boolean relation, depending on op. In this paper, we experiment with a set of ops which have one of its inputs inverted. We compare our implementations of such complemented-input circuits synthesized by a Boolean relation minimizer, BREL

[1], against published results that use special minimizers for deriving AND-OR-AND, OR-AND-OR, and AND-OR-XOR three-level forms. The use of a Boolean relation minimizer on such problems is not new [1] but experiments and comparisons in this context have not been done.

The paper is organized as follows: we show a motivating example in Sec. II, we summarize briefly previous work in Sec. III, Boolean relations in Sec. IV, and describe in Sec. V the Boolean relations characterizing completely the flexibility in the realization of *op* circuits. In Sec. VI, we report experimental results comparing our forms against SOPs (with different phase assignments) and specialized three-level minimizers like AOXMIN [10], after running SIS to compute areas and delays of the underlying SOPs; the experiments show average gains of around $20-30\%$ in the majority of benchmarks. In Sec. VII we draw conclusions and discuss possible future research.

II. Motivating Example

In order to better describe the proposed approach we give here a simple example of the bi-decomposition $f_0 \Rightarrow f_1 \equiv \overline{f}_0 + f_1$ for the function f depicted in Figure 1(a). We first recall that, by the De Morgan laws, a complemented SOP (Sum of Products) can be seen as a POS (Product of Sums) form with the same number of literals. For example: $\overline{x_1\overline{x}_2 + x_1x_3\overline{x}_4} = (\overline{x}_1 + x_2)(\overline{x}_1 + \overline{x}_3 + x_4)$.

Considering the function f represented by the Karnaugh map in Figure 1(a). If we compute a standard SOP cover we have the minimal SOP in Figure 1(b):

$$f_{SOP} = f_1^{SOP} = x_1\overline{x}_2 + x_1x_3 + x_1x_4 + \overline{x}_2x_3 + \overline{x}_2x_4,$$

that has 10 literals. The corresponding minimal POS form in Figure 1(c) is:

$$f_{POS} = \overline{f}_0^{POS} = (x_1 + \overline{x}_2)(x_1 + x_3 + x_4)(\overline{x}_2 + x_3 + x_4),$$

containing 8 literals. In this case it is convenient to represent the function as the negation of its offset with 3 products and 8 literals, against a cost of 5 products and 10 literals if we represent its onset. Moreover, we can do even better if we enlarge the offset to include the onset point 1000, because we save 1 product and 4 literals in the representation of the offset; however, we must represent the onset point 1000 by adding a product of the onset that covers it, paying a penalty of 1 product and 2 literals, with an overall cost of 3 products and 6 literals (better than 3 products and 8 literals). In conclusion, a minimal $f_0 \Rightarrow f_1$ circuit for f in Figure 1(d) is:

$$f_B = \overline{f}_0 + f_1 = ((x_1 + \overline{x}_2)(x_3 + x_4)) + x_1\overline{x}_2,$$

that contains 6 literals (it is $f_0 = \overline{x}_1x_2 + \overline{x}_3\overline{x}_4$ and $f_1 = x_1\overline{x}_2$). Note that in the last Karnaugh map (Figure 1(d)) the point 1000 is in the the OFF set of $\overline{f}_0$ but is in the ON set of f_1, thus is in the ON set of the OR between $\overline{f}_0$ and f_1 ($\overline{f}_0 + f_1$). Moreover, the points $1001, 1010$, and 1011 are covered by both $\overline{f}_0$ and f_1. We can conclude that it is useful to define a strategy that finds the best cover $f_B = \overline{f}_0 + f_1$. Note that, in general, the best solution could be $f_B = f_1$ (i.e., SOP) or $f_B = \overline{f}_0$ (i.e., POS or complemented SOP).

III. Previous Work

Three-level logic has been studied for decades, a reason being that three levels are enough to produce a minimal network for most Boolean functions (see Sasao, [19]). The minimization of various forms of three-level logic has been studied in the literature, e.g., AND-OR-AND networks consisting of two SOPs with a two-input AND gate at the output (Malik, [13] and Dubrova, [9]); OR-AND-OR networks (see Sasao, [20], and Debnath, [8]); AND-OR-EXOR networks, called EX-SOP, with a single two-input EXOR gate at the output (see Debnath, [6], [7] and Dubrova, [10]); EXOR-AND-OR networks, called SPPs (Sums of Pseudo-Products) which generalize SOP expressions by replacing products of literals with products of EXOR gates (see Luccio, [12]), further restricted to k-SPPs where EXOR factors contain at most k literals and to 2-SPPs (see Ciriani, [4], [5]) for which an efficient ESPRESSO-like minimization procedure has been designed (see Bernasconi, [2]).

A way to obtain three-level forms is to apply one step of bi-decomposition, which decomposes a given logic function $F(X)$ into three blocks as $F(X) = G(X)\ op\ H(X)$, where *op* is a two-input gate (usually AND, OR, or EXOR) (see Sasao, [21] and Mishchenko, [14]). A strong bi-decomposition has the form $G(X_1, X_3)\ op\ H(X_3, X_2)$ where X_1, X_2, X_3 is a partition of the input variables; When $X_2 = \emptyset$ the bi-decomposition is weak. In this paper we address the special case of weak decompositions where $X_1 = X_2 = \emptyset$. Some interesting results on bi-decomposition are described in [11], [15], [23], but they cannot be compared with the benchmark functions we have considered.

IV. Boolean Relations

The concept of Boolean relations was introduced as a more general scheme for the non-deterministic specification of logic networks, which cannot always be represented using don't cares [1], [3].

Definition 1: A *Boolean relation* is a one-to-many multi-output Boolean mapping $\mathcal{R} : \{0,1\}^n \rightarrow \{0,1\}^m$, where $\{0,1\}^n$ and $\{0,1\}^m$ are called the *input* and *output sets* of $\mathcal{R}$.

A Boolean relation $\mathcal{R}$ can be considered a generalization of a Boolean function, where a point in the input set $\{0,1\}^n$ can be associated with several points in the output set $\{0,1\}^m$; indeed, because of the one-to-many nature of Boolean relations, there may be several equivalent outputs for a given input. For example, consider the mapping $\mathcal{R} : \{0,1\}^2 \rightarrow \{0,1\}^3$ such that:

x	$\mathcal{R}(x)$
00	$\{001, 100\}$
01	$\{000\}$
10	$\{101, 111\}$
11	$\{100, 010, 001\}$

Note that we cannot represent this mapping using a simple incompletely specified Boolean function, since, for example, the input 00 can have as output 001 or 100 that cannot be merged into a single cube.

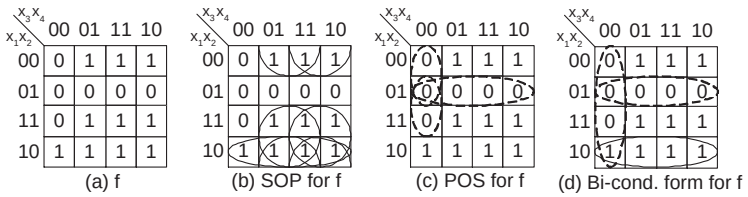

Fig. 1. Example of Karnaugh maps of a SOP form (b) a POS form (c) and a circuit $f_0 \Rightarrow f_1 \equiv \overline{f}_0 + f_1$ (d) for the Boolean function f in (a).

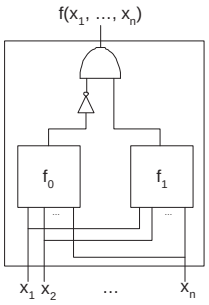

Fig. 2. Circuit decomposition with AND gate.

$u\,v$	AND
$\overline{u}\,v$	$\not\Leftarrow$
$u\,\overline{v}$	$\not\Rightarrow$
$\overline{u}\,\overline{v}$	NOR
$u+v$	OR
$\overline{u}+v$	$\Rightarrow$
$u+\overline{v}$	$\Leftarrow$
$\overline{u}+\overline{v}$	NAND
$u \oplus v$	XOR
$u\overline{\oplus}v$	XNOR

Fig. 3. Non-trivial binary operations

A relation $\mathcal{R}$ is *well-defined* if for all $x \in \{0,1\}^n$ there is $y \in \{0,1\}^m$ such that $(x,y) \in \mathcal{R}$. To any relation $\mathcal{R}$ we can associate a set $\mathcal{F}(\mathcal{R})$ of all *compatible* multi-output Boolean functions, i.e. the set of all functions g such that, for all inputs $x \in \{0,1\}^n$, $g(x)$ is contained in the set $\mathcal{R}(x)$ of the outputs related to x. In this case, we write $g \subseteq \mathcal{R}$. For example, consider the mapping $\mathcal{R}$ described in the previous example. $\mathcal{F}(\mathcal{R})$ contains the following Boolean function $g : \{0,1\}^n \rightarrow \{0,1\}^m$:

x	$g(x)$
00	001
01	000
10	101
11	100

The problem of the optimal implementation of a Boolean relation $\mathcal{R}$ is that of selecting, among the possible functions compatible with $\mathcal{R}$, one of minimum cost according to a given metric. More precisely, the *solution* of a Boolean relation $\mathcal{R}$ is a multi-output Boolean function $g \in \mathcal{F}(\mathcal{R})$. The function g is an *optimal solution* of $\mathcal{R}$ according to a given cost function μ, if for all $g' \in \mathcal{F}(\mathcal{R})$, $\mu(g) \leq \mu(g')$. In this paper, we will consider as cost $\mu(g)$ the number of literals in a minimal SOP form for f.

V. Bi-decomposed circuits

Given an incompletely specified function (ISF) f, defined as on-set f_{on}, off-set f_{off} and dc-set f_{dc}, we want to decompose f in u and v, such that f is covered by $u\ op\ v$, where op is a given binary operation, e.g. an AND, OR or XOR gate. The inputs of u and v are the same as the inputs of f. The output of f is the output of the chosen gate op which takes in input $u(x_1,\ldots,x_n)$ and $v(x_1,\ldots,x_n)$. Figure 2 reports the circuit obtained when op is represented by the $\overline{u}v$ gate ($\not\Leftarrow$).

This problem can be formulated as that of solving Boolean relations. For each binary operation op, we define a relation $\mathcal{R}_{op}$ whose set of compatible functions $\mathcal{F}(\mathcal{R}_{op})$ corresponds exactly to the set of pairs (u,v) occurring in all bi-decomposed circuit implementations of f with respect to the chosen operation op. An optimal solution of $\mathcal{R}_{op}$ is an optimal bi-decomposed circuit for f.

Let $\mathcal{R}_{op} : \{0,1\}^n \rightarrow \{0,1\}^2$ be a Boolean relation, describing all possible pairs of functions u, v defining a bi-decomposed circuit for f. We show how to construct the relation $\mathcal{R}_{op}$ for any binary operation op for the ten (out of sixteen) binary operations that depend on both input variables and omit operations $1, 0, u, \overline{u}, v, \overline{v}$. The 10 binary ops are shown in Figure 3. To construct $\mathcal{R}_{op}$, three cases are distinguished, depending on whether the input vector $x \in \{0,1\}^n$ belongs to the on-set, the off-set, or the dc-set of the function f. We partition these ops into the first four, the second four and the last two. For the first four, uv, $\overline{u}v$, $u\overline{v}$, $\overline{u}\overline{v}$, we note that each can be obtained from another by complementing one or more inputs. As an example, we construct $\mathcal{R}_{\not\Leftarrow}$ as follows:

- all points $x \in f_{on}$ must be associated to the output 01, so that the output of the circuit $\overline{u}v$ evaluates to 1, thus we define $\mathcal{R}_{\not\Leftarrow}(x) = \{01\}$;
- all points $x \in f_{off}$ must be associated to one of the three output values on which $\overline{u}v$ evaluates to 0, thus we define $\mathcal{R}_{\not\Leftarrow}(x) = \{00, 10, 11\} = \{1-, -0\}$;
- all points $x \in f_{dc}$ can be associated to any output, thus we have $\mathcal{R}_{\not\Leftarrow}(x) = \{--\}$.

The relations $\mathcal{R}_{AND}$, $\mathcal{R}_{\not\Rightarrow}$, and $\mathcal{R}_{NOR}$, corresponding to the other three operations in the first group (the AND group), can be defined in an analogous way. These are summarized in Table I. Similarly, the four Boolean relations $\mathcal{R}_{OR}$, $\mathcal{R}_{\Rightarrow}$, $\mathcal{R}_{\Leftarrow}$, and $\mathcal{R}_{NAND}$ (the OR group) are summarized in Table II and the last two (the XOR group) are summarized in Table III.

TABLE I. AND TABLE

	$\mathcal{R}_{\not\Leftarrow}$	$\mathcal{R}_{AND}$	NOR	$\mathcal{R}_{\not\Rightarrow}$
$x \in f_{on}$	$\{01\}$	$\{11\}$	$\{00\}$	$\{10\}$
$x \in f_{off}$	$\{1-,-0\}$	$\{0-,-0\}$	$\{1-,-1\}$	$\{0-,-1\}$
$x \in f_{dc}$	$\{--\}$	$\{--\}$	$\{--\}$	$\{--\}$

TABLE II. OR TABLE

	$\mathcal{R}_{\Rightarrow}$	$\mathcal{R}_{OR}$	$\mathcal{R}_{NAND}$	$\mathcal{R}_{\Leftarrow}$
$x \in f_{on}$	$\{0-,-1\}$	$\{1-,-1\}$	$\{0-,-0\}$	$\{1-,-0\}$
$x \in f_{off}$	$\{10\}$	$\{00\}$	$\{11\}$	$\{01\}$
$x \in f_{dc}$	$\{--\}$	$\{--\}$	$\{--\}$	$\{--\}$

Given an ISF, $f = (f_{on}, f_{off})$, the output z of the LUT must satisfy $f_{on} \subseteq z \subseteq \overline{f_{off}}$, i.e. z is a cover of f. This constraint is guaranteed by the Boolean relation minimizer. Let (f^1_{on}, f^1_{off}) be an ISF associated with the first output u of the PLA, and similarly (f^2_{on}, f^2_{off}) be that for the second output. Then either $u = u^1$ where $f^1_{on} \subseteq u^1 \subseteq \overline{f^1_{off}}$ or $u = u^0$ where $f^1_{off} \subseteq u^0 \subseteq \overline{f^1_{on}}$ and similarly for the second output, $v = v^1$ or $v = v^0$. Thus the LUT at the output affords us the flexibility of implementing an onset cover or offset cover for the two outputs leading to a two-output phase assignment problem.

Since our three-level form consists of a PLA with each pair of outputs feeding into a two-input LUT, only what is implemented in the PLA is important; any cost function should be independent of the *op* implemented in the LUT.

Note that the ISFs associated with u and v depend on the *op* in the LUT as dictated by the Boolean relations given by the three Tables I, II, III. Any entry (column) in a table can be obtained from any other entry in the same table by simply inverting one or more of the inputs. Thus for a given table, phase assignment maps one column into another. The three tables are distinguished by whether the offset is partitioned (AND table), the onset is partitioned (OR table) or both are partitioned (XOR table). For example, in the first table for $\mathcal{R}_{\not\Leftarrow}$, the care minterms of f are distributed as follows: f_{on} is put in f^1_{off} as well as in f^2_{on}, and f_{off} is partitioned into three parts, those in f^1_{on} only, those in f^2_{off} only, and those in both. How this partitioning is done is the task of the Boolean relation minimizer. Thus, given a distribution of care minterms (i.e. partitionings), the four choices for the two outputs implemented in the PLA are $(u, v) \in \{(u^1, v^1), (u^0, v^1), (u^1, v^0), (u^0, v^0)\}$ and these choices correspond to the choices of Boolean relations in each of the three tables. We note for future reference, that for the Boolean relation minimizer, BREL [1], complementing an input in the Boolean relation will simply switch the implementation of that output of the PLA from a cover of the onset to a cover of the offset.

In the literature, the following methods for three level minimization have been investigated: 1) AND-OR-AND [8,11], 2) OR-AND-OR [7,17], and 3) AND-OR-XOR [5,6,9]. To compare their results against ours is difficult because each method uses a different minimizer which may induce different partitionings. Although a partitioning induced by any method can be inferred from u and v by determining which onset/offset minterms are in $u, \overline{u}, v, \overline{v}$ for any of the methods, it is not easy to force the algorithms to use the same partitionings. Thus a controlled experiment can't be done easily. However, we note that the AND-OR-AND method might be similar to using BREL on $\mathcal{R}_{AND}$, OR-AND-OR similar to $\mathcal{R}_{OR}$, and AND-OR-XOR similar to $\mathcal{R}_{XOR}$. With the formalism of the Boolean relations, we can rephrase our complemented circuit minimization problem as the problem of finding an optimal implementation of $\mathcal{R}_{op}$ (for *op* corresponding to one of the selected binary operations), that is, of selecting among all possible two-output functions compatible with $\mathcal{R}_{op}$, the one defining the couple (u, v) leading to a circuit of minimal cost, according to a given cost metric:

TABLE III. XOR TABLE

	$\mathcal{R}_{XNOR}$	$\mathcal{R}_{XOR}$
$x \in f_{on}$	$\{00, 11\}$	$\{01, 10\}$
$x \in f_{off}$	$\{01, 10\}$	$\{00, 11\}$
$x \in f_{dc}$	$\{--\}$	$\{--\}$

TABLE V. GAINS OF BEST *op* CIRCUITS

	EXACT			HEURISTIC		
	$\not\Leftarrow$	$\Rightarrow$	XNOR	$\not\Leftarrow$	$\Rightarrow$	XNOR
time	2%	4%	2%	11%	11%	8%
area	89%	87%	78%	65%	63%	47%
delay	63%	69%	50%	63%	69%	43%

TABLE VI. AVERAGE GAIN OF COMPLEMENTED CIRCUITS

	EXACT			HEURISTIC		
	$\not\Leftarrow$	$\Rightarrow$	XNOR	$\not\Leftarrow$	$\Rightarrow$	XNOR
time	-86%	-83%	-63%	-1077%	-853%	-1030%
area	29%	29%	26%	16%	16%	13%
delay	19%	20%	15%	12%	15%	6%

Theorem 1: The set $\mathcal{F}(\mathcal{R}_{op})$ of all two-output functions compatible with the relation $\mathcal{R}_{op}$ specifies exactly the set of all pairs (u, v), occurring in all possible circuit implementations where $z = u\ op\ v$ is a cover of the ISF $f = (f_{on}, f_{off})$.

Proof: Let $C(f)$ be a circuit with two outputs u and v, such that $z = u\ op\ v$ is a cover of f. Then $C(f)$ is compatible with R_{op} because it is easy to verify that for all $x \in \{0,1\}^n$, $(u(x), v(x)) \in \mathcal{R}_{op}(x)$. Conversely, let $C(f)$ be any two-output function compatible with $\mathcal{R}_{op}$. Observe that the definition of R_{op} guarantees that the two outputs of each function $g \in \mathcal{F}(\mathcal{R}_{op})$ combined with the chosen operation *op*, evaluates to 1 on all points in f_{on} and to 0 on all points in f_{off}. Thus, with u as the first output of g and v as the second, we get that $u\ op\ v$ is a cover of f. ■

Corollary 1: An optimum solution of the Boolean relation $\mathcal{R}_{op}$, according to a given cost function μ, defines an optimum bi-decomposed circuit, $z = u_{opt}\ op\ v_{opt}$ for f with the minimum cost μ.

Proof: An optimum bi-decomposed circuit for f is one where $z = u_{opt}\ op\ v_{opt}$ is a cover of f and $\mu(z)$ is minimum. By Theorem 1, the two-output function (u_{opt}, v_{opt}) is compatible with $\mathcal{R}_{op}$ and hence a solution of $\mathcal{R}_{op}$. Since $\mu(u_{opt}, v_{opt})$ is minimum then also $z = u_{opt}\ op\ v_{opt}$ is a minimum cover with respect to *op* and μ ■

VI. EXPERIMENTAL RESULTS

In this section we report the experimental results for the minimization of the Boolean relations of three *op* circuitss, one from each of the three Tables I, II, III. The representative

TABLE IV. COMPARISON OF SOP VS *op* CIRCUITS

	SOP exact			SOP heuristic			$\not\Leftarrow$ exact			$\not\Leftarrow$ heuristic			$\Rightarrow$ exact			$\Rightarrow$ heuristic			XNOR exact			XNOR heuristic		
benchmark	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**
add6	0.77	292	25.20	0.03	292	25.20	7.60	297	25.30	0.08	290	25.40	6.97	294	22.70	0.09	**287**	24.10	5.32	**292**	26.60	0.07	327	28.10
addm4	0.04	934	42.70	0.02	959	41.90	7.31	781	30.10	0.02	918	38.50	7.51	807	36.90	0.02	**899**	39.60	8.10	**745**	31.00	0.02	1016	44.80
al2	0.86	340	15.10	0.00	340	15.10	3.76	332	14.00	0.01	**335**	14.20	0.58	**331**	13.10	0.00	443	13.80	4.87	505	16.10	0.01	425	15.10
alu2	0.01	176	16.40	0.00	176	14.50	1.71	160	12.30	0.03	203	20.50	1.68	158	14.00	0.01	195	16.50	1.81	**155**	17.80	0.02	**187**	19.90
alu3	0.01	187	16.80	0.00	157	13.30	1.66	158	12.60	0.03	199	20.80	1.69	**157**	13.30	0.02	199	17.00	1.75	159	17.80	0.01	**179**	19.10
amd	0.02	982	38.50	0.01	986	37.30	7.38	**736**	37.00	0.09	**815**	34.20	7.00	752	38.60	0.06	852	34.50	7.50	818	35.10	0.06	892	35.80
b2	0.08	3984	76.90	0.01	3957	73.00	168.31	3219	69.10	1.32	3597	69.70	170.34	3207	66.30	1.10	**3492**	68.70	151.47	**3156**	65.20	1.60	3567	76.30
b3	0.81	1126	45.00	0.03	1095	44.70	281.90	877	34.60	0.51	1250	36.20	279.14	**853**	35.80	0.47	1337	39.20	192.95	903	36.80	0.45	**1048**	34.70
b4	1.17	649	31.20	0.00	649	31.20	7.39	596	26.60	0.75	**657**	28.00	7.19	**594**	26.60	0.69	731	30.40	8.18	616	28.80	0.62	697	29.70
bcc	0.03	5167	92.10	0.03	5150	93.30	66.16	3850	66.80	1.13	5334	94.60	84.34	**3799**	64.30	0.72	**5195**	93.80	66.51	3985	67.40	0.84	5224	92.90
bench	0.01	172	15.00	0.00	144	12.10	0.97	**98**	12.10	0.00	120	14.00	0.62	**98**	11.30	0.00	**110**	11.70	1.03	116	15.00	0.00	141	15.10
br2	0.00	280	28.30	0.00	280	28.30	1.81	237	18.40	0.00	265	19.00	1.28	**234**	18.10	0.00	**261**	25.10	2.04	256	21.70	0.00	268	24.90
dc1	0.00	85	11.80	0.00	85	12.10	0.69	**72**	10.30	0.00	**77**	13.40	0.41	74	10.80	0.00	79	13.80	0.74	80	11.60	0.00	109	15.40
dc2	0.00	243	24.70	0.00	233	20.40	1.46	**184**	20.30	0.00	210	18.30	1.59	185	19.40	0.00	**199**	18.20	1.42	197	20.50	0.00	220	21.10
ex7	0.14	225	21.20	0.00	225	21.20	3.06	194	24.90	0.04	**231**	20.10	2.78	**193**	19.70	0.04	238	20.60	3.37	219	23.00	0.04	251	24.00
exep	0.04	1285	31.40	0.01	1275	36.10	9.18	1290	31.00	0.46	**1295**	33.50	1.86	**1271**	31.20	0.10	1300	31.40	11.08	1454	36.60	0.56	1311	36.90
exps	0.06	3579	94.50	0.02	3549	101.10	17.69	2405	62.50	0.01	2697	73.50	17.98	**2387**	64.30	0.01	**2574**	71.90	16.87	2429	65.10	0.00	2713	68.20
fout	0.05	535	30.40	0.00	472	27.10	2.82	339	23.10	0.00	**398**	25.00	2.80	336	21.90	0.00	418	26.20	2.40	**325**	23.10	0.00	402	28.00
in1	0.07	3984	76.90	0.01	3957	73.00	167.55	3219	69.10	1.28	3597	69.70	170.23	3207	66.30	1.11	**3492**	68.70	151.62	**3156**	65.20	1.56	3567	76.30
in3	0.12	1111	34.10	0.00	1111	34.10	8.71	928	32.80	0.16	985	32.80	7.79	**913**	32.90	0.22	**970**	33.60	8.95	971	36.20	0.08	974	37.80
in4	0.65	1127	44.90	0.02	1077	44.80	286.03	924	35.60	0.59	1297	36.60	280.34	**900**	36.80	0.51	1384	39.30	194.52	950	37.80	0.46	**1114**	35.60
in6	0.77	662	31.30	0.00	662	31.30	7.06	609	26.70	0.23	**669**	28.30	6.64	**607**	26.70	0.14	744	30.50	7.67	629	28.80	0.08	710	29.80
in7	0.25	319	23.60	0.00	319	23.60	4.80	305	23.30	0.13	340	26.90	4.81	**304**	23.10	0.09	**325**	28.40	4.99	338	27.10	0.09	379	33.50
luc	0.00	821	40.70	0.00	845	40.20	5.40	**596**	28.00	0.01	649	31.80	4.66	599	27.30	0.01	**641**	28.20	4.71	620	28.40	0.00	727	34.00
m181	0.24	234	20.50	0.00	174	16.40	0.92	114	11.10	0.00	141	18.10	1.08	**113**	12.00	0.00	**127**	11.50	1.34	134	14.80	0.00	169	19.30
m3	0.00	1169	47.10	0.00	1269	51.50	3.96	502	29.90	0.00	547	28.00	3.75	**492**	30.60	0.01	536	30.90	3.67	535	33.60	0.00	**531**	31.30
m4	0.04	1904	66.50	0.02	1778	54.20	5.91	**799**	35.20	0.00	**863**	36.50	5.80	807	34.80	0.02	867	36.90	5.64	812	38.50	0.01	901	38.70
mlp4	0.10	702	34.50	0.01	686	36.40	4.69	**569**	30.50	0.01	629	30.10	4.59	579	31.10	0.00	**595**	29.30	4.64	573	35.10	0.01	630	36.90
mp2d	0.08	268	20.60	0.00	251	19.80	1.72	224	17.00	0.00	269	20.30	1.36	**222**	19.00	0.00	**233**	17.20	2.17	239	17.10	0.00	238	15.20
p1	0.03	651	32.20	0.02	645	36.00	3.63	382	24.60	0.00	466	28.50	3.71	**375**	21.20	0.00	**441**	27.80	3.74	396	24.40	0.02	478	26.80
p3	0.02	447	26.70	0.00	448	26.60	2.35	**245**	21.50	0.00	293	23.90	2.32	248	18.70	0.01	**285**	19.10	2.40	258	21.50	0.00	308	23.50
shift	801.94	372	27.00	0.00	372	27.00	3.64	**371**	27.20	0.03	**372**	27.40	3.45	372	27.00	0.02	**372**	27.00	4.17	377	27.20	0.01	**372**	27.40
spla	0.80	2422	57.40	0.19	2470	68.60	14.56	**1678**	44.70	0.05	**1859**	48.40	14.38	1688	46.30	0.09	1862	47.40	16.68	1763	40.60	0.04	1932	49.00
sym10	1.23	519	29.90	0.02	592	31.70	59.06	344	35.70	0.00	379	31.50	54.97	**301**	32.40	0.01	**350**	27.50	32.66	**301**	30.40	0.01	379	31.50
t1	6.83	546	22.10	0.01	458	21.90	3.66	**347**	18.00	0.00	**378**	19.30	3.46	353	17.90	0.03	388	20.50	4.49	374	19.30	0.00	412	22.90
t4	0.01	102	14.10	0.00	96	11.60	0.91	96	14.90	0.00	**97**	14.80	0.63	**93**	13.00	0.00	100	9.40	0.97	114	11.90	0.00	116	19.30
test1	39.11	1392	47.50	0.02	1318	46.60	7.95	**935**	39.40	0.10	1351	44.90	8.37	**935**	35.30	0.07	1369	47.00	8.00	944	39.00	0.10	**1302**	46.30
tial	1.31	2376	68.70	0.20	2327	66.00	123.41	1516	51.60	0.51	**1654**	50.40	104.32	**1489**	50.80	0.31	1715	49.30	142.05	1537	53.10	0.64	2428	60.00
vg2	0.07	341	18.60	0.00	341	18.60	3.80	287	22.00	0.10	495	21.70	4.24	**284**	20.00	0.09	**369**	18.60	5.16	292	22.00	0.06	376	18.60
vtx1	0.05	324	21.30	0.00	324	21.30	3.19	**238**	17.90	0.07	424	28.30	3.46	259	20.50	0.10	**383**	25.30	4.13	257	19.70	0.05	429	27.20
x6dn	0.04	789	34.40	0.00	762	31.20	6.91	**725**	36.40	0.13	772	32.20	6.91	738	32.50	0.13	**770**	33.50	6.75	734	32.90	0.08	777	34.80
x9dn	0.06	384	23.00	0.00	384	23.00	3.33	262	22.70	0.08	530	28.00	4.90	**254**	20.90	0.12	**464**	24.60	4.68	258	19.40	0.09	590	32.70

TABLE VII. MINIMIZATION RESULTS

		$\not\Leftarrow$ exact		$\not\Leftarrow$ heuristic		$\Rightarrow$ exact		$\Rightarrow$ heuristic		XNOR exact		XNOR heuristic		[9]		[10]	
benchmark	**in/out**	**time**	**lit**	**time**	**lit**	**time**	**lit**	**time**	**lit**	**time**	**lit**	**time**	**lit**	**time**	**lit**	**time**	**lit**
5xp1	7/10	0.00	**17**	0.00	**17**	0.00	20	0.00	20	0.00	20	0.00	20	14.00	55	14.80	42
9sym	9/1	7.79	73	0.00	73	7.21	**71**	0.00	72	7.40	72	0.00	72			1.70	73
alu2	10/8	1.71	67	0.03	79	1.68	68	0.01	83	1.81	**48**	0.02	64	32.00	52		
alu3	10/8	1.66	67	0.03	79	1.69	70	0.02	86	1.75	49	0.01	64	24.00	**47**		
alu4	14/8	107.70	518	0.41	631	101.58	520	0.27	632	87.42	472	0.53	658			131.90	**447**
b12	15/9	0.82	42	0.01	51	0.96	40	0.00	47	1.28	42	0.00	58	30.00	**28**	9.10	31
bw	5/28	0.00	56	0.00	56	0.00	56	0.00	56	0.00	56	0.00	56			25.40	**24**
clip	9/5	3.60	149	0.01	158	3.55	150	0.00	159	3.71	116	0.03	162			10.50	**95**
con1	7/2	0.18	10	0.00	10	0.25	10	0.00	11	0.29	10	0.00	13			1.00	**9**
cordic	23/2	1909.46	345	0.15	1180	1908.45	243	0.14	2058	1024.40	158	0.15	1187			231.80	**156**
dist	8/5	5.12	153	0.01	173	5.08	153	0.01	179	4.64	146	0.01	149	170.00	**108**		
ex1010	10/10	35.51	639	0.26	1118	35.47	630	0.27	1124	34.99	**608**	0.49	1051			109.70	725
inc	7/9	1.26	58	0.00	64	1.25	57	0.00	59	1.44	55	0.00	59			6.50	**33**
misex1	8/7	1.04	43	0.00	48	0.89	43	0.00	44	0.97	40	0.00	46			11.50	**13**
misex2	25/18	1.67	76	0.00	133	0.32	104	0.00	97	2.29	116	0.00	121			6.00	**28**
misex3	14/14	106.90	585	0.44	1051	116.77	576	0.29	961	84.78	672	0.49	1270			112.00	**191**
newapla2	6/7	0.52	30	0.00	41	0.00	42	0.00	42	0.63	42	0.00	34	0.39	**5**		
newbyte	5/8	0.59	25	0.00	35	0.00	40	0.00	40	0.71	40	0.00	37	0.52	**5**		
newcpla1	9/16	1.85	81	0.00	94	1.41	81	0.01	95	2.30	99	0.00	92	26.00	**27**		
newtpla	15/5	0.99	33	0.01	50	0.74	36	0.00	54	1.04	40	0.00	38	1.20	**19**		
radd	8/5	1.54	76	0.01	86	1.43	77	0.00	75	1.18	49	0.00	52	14.00	**39**		
rd53	5/3	0.00	11	0.00	11	0.09	**6**	0.00	8	0.17	8	0.00	7	5.90	26	15.70	19
rd73	7/3	0.00	17	0.00	17	0.16	**6**	0.00	9	0.24	8	0.00	9	312.00	79	25.50	83
rd84	8/4	14.74	309	0.00	299	14.16	310	0.00	303	10.53	194	0.00	247			61.10	**192**
ryy6	16/1	0.52	8	0.01	8	0.74	**7**	0.01	**7**	0.68	**7**	0.00	113	379.00	**7**		
sao2	10/4	0.04	30	0.00	12	0.46	13	0.00	14	0.55	**8**	0.00	**8**			3.70	38
sqn	7/3	1.18	47	0.00	51	1.23	41	0.00	48	1.25	43	0.00	53	6.10	**34**		
squar5	5/8	0.69	36	0.00	37	0.61	39	0.00	39	0.85	38	0.00	43			4.40	**22**
t2	17/16	2.69	94	0.02	126	2.42	124	0.01	123	3.06	110	0.00	116	44.00	**46**		
t481	16/1	14.69	361	0.01	361	12.24	360	0.00	360	43.92	441	0.01	482			557.20	**113**
table3	14/14	62.89	751	0.58	1385	59.19	861	0.31	1094	57.80	838	0.46	1140			79.30	**176**
table5	17/15	68.56	862	0.48	1427	72.28	977	0.35	797	67.98	800	0.41	1178			100.70	**158**
x1dn	27/6	2.27	78	0.11	83	2.58	**76**	0.10	81	5.83	76	0.19	81	237.00	81		
x9dn	27/7	3.33	111	0.08	244	4.90	103	0.12	158	4.68	110	0.09	297	307.00	**81**		
xor5	5/1	0.29	16	0.00	16	0.30	16	0.00	16	0.18	**9**	0.00	**9**			10.30	10
vg2	25/8	0.61	22	0.13	42	0.63	**18**	0.15	22	0.71	**18**	0.21	21	366.00	88	43.40	102
z4	7/4	1.09	55	0.00	61	1.12	62	0.00	60	0.95	41	0.00	41	2.10	**33**		
Z5xp1	7/10	1.69	86	0.00	82	1.76	79	0.00	83	1.53	74	0.00	80	112.00	**56**		

TABLE VIII. TIME, AREA AND DELAY: *op* CIRCUITS VS ESPRESSO WITH OUTPUT PHASE ASSIGNMENT.

	ESPRESSO - exact			ESPRESSO - heuristic			$\not\Leftarrow$ exact			$\not\Leftarrow$ heuristic			$\Rightarrow$ exact			$\Rightarrow$ heuristic			XNOR exact			XNOR heuristic		
benchmark	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**	**time**	**area**	**delay**
add6	0.96	292	25.20	0.33	288	23.50	7.60	297	25.30	0.08	290	25.40	6.97	294	22.70	0.09	**287**	24.10	5.32	**292**	26.60	0.07	327	28.10
al2	1.09	340	15.10	0.05	370	15.20	3.76	332	14.00	0.01	**335**	14.20	0.58	**331**	13.10	0.00	443	13.80	4.87	505	16.10	0.01	425	15.10
alcom	0.38	222	12.30	0.00	231	12.30	2.08	**210**	12.20	0.02	**210**	11.20	0.21	**210**	11.20	0.01	216	11.10	3.25	345	16.50	0.01	304	15.40
alu1	0.03	53	6.80	0.00	55	6.80	0.21	**53**	6.80	0.00	**53**	6.80	0.06	**53**	6.80	0.00	**53**	6.80	0.62	59	7.00	0.00	85	12.00
b10	0.10	876	42.80	0.09	881	45.20	10.40	**721**	31.90	0.13	**827**	34.80	11.47	723	32.00	0.11	857	37.00	10.00	745	37.00	0.15	909	37.30
b3	1.12	1126	45.00	1.03	1095	44.70	281.90	877	34.60	0.51	1250	36.20	279.14	**853**	35.80	0.47	1337	39.20	192.95	903	36.80	0.45	**1048**	34.70
b4	1.62	649	31.20	1.64	649	31.20	7.39	596	26.60	0.75	**657**	28.00	7.19	**594**	26.60	0.69	731	30.40	8.18	616	28.80	0.62	697	29.70
b7	0.00	234	16.30	0.01	219	15.90	1.93	**195**	15.00	0.01	212	16.60	0.22	196	15.30	0.00	**203**	16.10	2.72	294	18.10	0.00	236	17.10
b9	0.20	225	21.20	0.04	225	21.20	3.14	194	24.90	0.04	**231**	20.10	2.80	**193**	19.70	0.03	238	20.60	3.36	219	23.00	0.05	251	24.00
bench	0.01	172	15.00	0.00	218	20.50	0.97	**98**	12.10	0.00	120	14.00	0.62	**98**	11.30	0.00	**110**	11.70	1.03	116	15.00	0.00	141	15.10
dist	0.01	735	36.70	0.05	790	40.40	5.12	596	33.40	0.01	646	35.20	5.08	611	32.50	0.01	625	29.90	4.64	**543**	33.70	0.01	**600**	37.90
fout	0.06	535	30.40	0.02	519	29.40	2.82	339	23.10	0.00	**398**	25.00	2.80	336	21.90	0.00	418	26.20	2.40	**325**	23.10	0.00	402	28.00
in0	0.03	1005	47.80	0.07	1032	54.20	12.44	**835**	36.60	0.17	**979**	41.40	13.36	841	38.30	0.13	988	39.40	12.03	864	38.40	0.14	1161	46.50
in1	0.09	3984	76.90	0.16	3957	73.00	167.55	3219	69.10	1.28	3597	69.70	170.23	3207	66.30	1.11	**3492**	68.70	151.62	**3156**	65.20	1.56	3567	76.30
in2	0.02	939	36.70	0.12	997	40.80	88.78	831	34.30	0.45	**953**	37.60	104.00	**820**	34.90	0.38	1019	37.50	86.56	847	40.50	0.38	982	37.10
in3	0.16	1111	34.10	0.28	1095	34.60	8.71	928	32.80	0.16	985	32.80	7.79	**913**	32.90	0.22	**970**	33.60	8.95	971	36.20	0.08	974	37.80
in4	0.84	1127	44.90	1.04	1128	44.00	286.03	924	35.60	0.59	1297	36.60	280.34	**900**	36.80	0.51	1384	39.30	194.52	950	37.80	0.46	**1114**	35.60
in5	0.06	865	38.50	0.28	865	38.50	8.47	**686**	31.90	0.15	937	34.90	8.47	696	31.40	0.18	**831**	31.70	11.35	720	31.60	0.21	954	40.70
in6	0.99	662	31.30	0.48	662	31.30	7.06	609	26.70	0.23	**669**	28.30	6.64	**607**	26.70	0.14	744	30.50	7.67	629	28.80	0.08	710	29.80
in7	0.32	319	23.60	0.03	321	23.90	4.80	305	23.30	0.13	340	26.90	4.81	**304**	23.10	0.09	**325**	28.40	4.99	338	27.10	0.09	379	33.50
lin	3.06	2849	89.40	0.24	3289	98.50	15.26	1755	53.90	0.02	1909	58.50	14.11	1757	52.30	0.01	**1899**	54.80	13.47	**1645**	54.30	0.01	1921	54.40
m181	0.30	234	20.50	0.02	132	12.90	0.92	114	11.10	0.00	141	18.10	1.08	**113**	12.00	0.00	**127**	11.50	1.34	134	14.80	0.00	169	19.30
m4	0.04	1904	66.50	0.09	1166	44.80	5.91	**799**	35.20	0.00	**863**	36.50	5.80	807	34.80	0.02	867	36.90	5.64	812	38.50	0.01	901	38.70
max128	0.03	2149	67.70	0.07	1588	55.80	5.08	632	30.90	0.01	701	34.50	4.89	**624**	31.40	0.00	698	33.00	4.71	625	30.50	0.00	**683**	38.00
max512	0.04	919	41.00	0.11	793	38.80	6.74	664	36.00	0.02	780	44.50	6.59	669	35.20	0.02	772	43.10	6.00	**659**	33.20	0.01	**753**	43.50
mlp4	0.12	702	34.50	0.08	709	35.70	4.69	**569**	30.50	0.01	629	30.10	4.59	579	31.10	0.00	**595**	29.30	4.64	573	35.10	0.01	630	36.90
mp2d	0.10	268	20.60	0.04	227	17.40	1.72	224	17.00	0.00	269	20.30	1.36	**222**	19.00	0.00	**233**	17.20	2.17	239	17.10	0.00	238	15.20
spla	0.96	2422	57.40	0.96	2	0.00	14.56	**1678**	44.70	0.05	**1859**	48.40	14.38	1688	46.30	0.09	1862	47.40	16.68	1763	40.60	0.04	1932	49.00
sym10	1.66	519	29.90	0.06	592	31.70	59.06	344	35.70	0.00	379	31.50	54.97	**301**	32.40	0.01	**350**	27.50	32.66	**301**	30.40	0.01	379	31.50
t1	8.86	546	22.10	0.24	487	21.70	3.66	**347**	18.00	0.00	**378**	19.30	3.46	353	17.90	0.03	388	20.50	4.49	374	19.30	0.00	412	22.90
t2	0.00	352	24.10	0.08	351	26.90	2.69	**281**	21.00	0.02	**300**	20.40	2.42	283	20.00	0.01	**300**	20.70	3.06	314	19.00	0.00	322	20.40
t4	0.01	102	14.10	0.03	92	12.20	0.91	96	14.90	0.00	**97**	14.80	0.63	**93**	13.00	0.00	100	9.40	0.97	114	11.90	0.00	116	19.30
test1	50.63	1392	47.50	0.18	1474	49.70	7.95	**935**	39.40	0.10	1351	44.90	8.37	**935**	35.30	0.07	1369	47.00	8.00	944	39.00	0.10	**1302**	46.30
tial	1.68	2376	68.70	1.01	1928	49.60	123.41	1516	51.60	0.51	**1654**	50.40	104.32	**1489**	50.80	0.31	1715	49.30	142.05	1537	53.10	0.64	2428	60.00
vg2	0.14	46	10.70	0.01	46	10.70	3.80	287	22.00	0.10	495	21.70	4.24	**284**	20.00	0.09	**369**	18.60	5.16	292	22.00	0.06	376	18.60
vtx1	0.08	324	21.30	0.43	324	21.30	3.19	**238**	17.90	0.07	424	28.30	3.46	259	20.50	0.10	**383**	25.30	4.13	257	19.70	0.05	429	27.20
x6dn	0.06	789	34.40	0.12	762	31.20	6.91	**725**	36.40	0.13	772	32.20	6.91	738	32.50	0.13	**770**	33.50	6.75	734	32.90	0.08	777	34.80
x9dn	0.08	384	23.00	0.64	545	37.70	3.33	262	22.70	0.08	530	28.00	4.90	**254**	20.90	0.12	**464**	24.60	4.68	258	19.40	0.09	590	32.70

circuits have at least one complemented input to distinguish them from existing published methods. We report the result for the *op* gates $\not\Leftarrow$, $\Rightarrow$, and XNOR.

The algorithms have been implemented in C, using the CUDD library for OBDDs to represent Boolean functions. We used BREL [1] for the synthesis of Boolean relations, because it finds better solutions in shorter runtimes than previous methods. The exact minimization in BREL is obtained using SIS; the heuristic minimization is based on the BDD size; even if the resulting number of literals is higher, the second version is useful since it is much more efficient w.r.t. synthesis time. The experiments were run on a Linux Intel Core i7, 3.40 GHz CPU with 8 GB of main memory. The benchmarks are taken from LGSynth93 [24]. Multi-output benchmarks were synthesized by minimizing each output independently from the others. To show the performance in area of the circuits derived by using Boolean relations, we generated SOP forms of (u, v) using ESPRESSO both in exact and heuristic mode. To evaluate the obtained circuits, the SIS system was used with the MCNC technology library for mapping and the SIS command `map -W -f 3 -s` to estimate area and delay.

In Table IV we compare synthesis time (in seconds), mapped area and delay of circuits and SOP forms for a significant subset of the benchmarks. The first column reports the names of the benchmarks. The following columns report, by groups of three, the synthesis times in seconds, the areas and delays estimated by SIS. The first two groups, labeled SOP exact and SOP heuristic, refer to plain SOPs. The next group refers to circuits synthesized with different *op* gates ($\not\Leftarrow$, $\Rightarrow$, and XNOR).

Table V summarizes the comparison between *op* circuits and SOP forms for both exact and heuristic minimization. In more detail, we report the percentages of benchmarks where we obtain better results with respect to the corresponding SOP forms. We observe that a high percentage of *op* circuits synthesized with Boolean relations turned out to be more compact and yielded lower delays than the corresponding SOP forms. This is true for both the exact and heuristic mode.

Table VI reports the average gains obtained with *op* circuits with respect to SOP forms for both exact and heuristic minimization. We observe that the synthesis of the *op* circuits yields positive gains for both area and delay, at the expense of more computation times. Note that *op* circuits with an XNOR *op* gate exhibit a lower gain w.r.t. the corresponding benchmarks with an AND or OR *op* gate. On average, we obtain higher gains in the case of exact minimization. In particular, the maximum percentage of area gain is 71% (max128) in the exact case and 67% (Z9sym) in the exhaustive case (in these two examples, the percentage is the same for all *op* gates).

In Table VII we compare against approaches presented in [9] and in [10]. In [9] the authors describe a three-level logic synthesis procedure based on a novel strategy for pairing cubes. They ran the experiments on a Sun Ultra 60 operating with two 360 MHz CPU and with 1024 MB RAM main storage. In [10] the authors describe a three-level heuristic AND-OR-XOR minimization strategy for incompletely specified Boolean functions. They ran their experiments on a Sun SPARC 20 operating at 50 MHz with 64 MB RAM main storage. The first and the second columns of the table report the names of the benchmarks and the number of inputs/outputs, respectively. The following six columns report both execution time (in seconds) and number of literals of minimized *op* circuits for each case ($\not\Leftarrow$, $\Rightarrow$, and XNOR) and two different cost functions: the first (SIS) minimizes the number of literals, and the second (BDD) minimizes the size of the BDDs used for representing the relations. Finally, the last columns report results presented in [9] (columns 15 and 16) and in [10] (columns 17 and 18). The results show that the *op* circuits synthesized with Boolean relations turned out to be more compact than the corresponding circuits proposed in [9] and in [10] in about 29% of our experiments. Finally, we compare computation time, area, delay and number of literals of *op* circuits with respect to the results of ESPRESSO when run

with output phase assignment (to choose the best realization between the positive and negative phase of each output). After choosing a phase assignment for each output, the function is minimized (in heuristic or exact mode). In about 15% of tested benchmarks, we stopped the computation of ESPRESSO with output phase assignment (after about 30 minutes), without obtaining minimization results.

In Tab.VIII we compare synthesis time (in seconds), mapped area and delay of circuits synthesized with ESPRESSO after the phase optimization (ESPRESSO command `-Dopo`) against *op* circuits. The first column reports the names of the benchmarks. The following columns report, by groups of three, the synthesis time in seconds, the area and delay estimated by SIS. The first two groups refer to ESPRESSO synthesis. The next group refers to *op* circuits synthesized with different *op* gates ($\not\Leftarrow$, $\Rightarrow$, and XNOR). In the exact case, the percentages of *op* circuits with lower computation time, area and delay w.r.t. the corresponding circuits minimized with ESPRESSO are 5%, 94%, and 89%, respectively. In the heuristic case, the percentages of *op* circuits with lower time, area and delay w.r.t. the corresponding circuits minimized with ESPRESSO are 87%, 75%, and 77%, respectively.

VII. Conclusions and future work

We considered the bi-decomposition of ISFs, which have the form $u\ op\ v$, where *op* can be any two-input logic function. Then we characterized all the correct implementations in such a form in terms of logic functions compatible with a Boolean relation, depending on the operator *op*. We studied the taxonomy of such circuits, and classified them into three groups according to the chosen *op* gate. Any member of a group can be transformed into any other member of the same group by complementing one or more of the inputs. Then we experimented with one example *op* from each group, namely $op \in \{\not\Leftarrow, \Rightarrow, \text{and XNOR}\}$. These were chosen to differ from other forms already studied in the literature. Finally, we reported experiments to compare such realizations vs. SOPs as well as other published three-level forms, in term of area and delay, evaluated by synthesizing and mapping the circuits with SIS. This showed good gains in a majority of benchmarks against affordable increases in synthesis runtime.

Future work includes completing an exhaustive study of all ten non-trivial *op*s, and in finding a way to choose the best *op* for a particular ISF. This might be based on solving a phase assignment problem within each of the three group classifications. A variant of BREL might be developed to examine, at each of its steps, choosing a function or its complement implementation. Moreover, since in general we have multi-output ISFs to implement, treating them all at once using a single Boolean relation would be of great interest. Another interesting direction would be to iterate the construction of the blocks u and v recursively to obtain multi-level bi-decompositions with higher depths. Also, inclusion of a MUX as an *op* would be interesting. The question of taking better advantage of logic sharing among the 2*m* outputs is another potential direction of investigation.

References

[1] D. Bañeres, J. Cortadella, and M. Kishinevsky, "A Recursive Paradigm to Solve Boolean Relations," *IEEE Transactions on Computers*, vol. 58, no. 4, pp. 512–527, April 2009.

[2] A. Bernasconi, V. Ciriani, R. Drechsler, and T. Villa, "Logic Minimization and Testability of 2-SPP Networks," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 7, pp. 1190–1202, July 2008.

[3] R. Brayton and F. Somenzi, "An Exact Minimizer for Boolean Relations," in *Computer-Aided Design, 1989. ICCAD-89. Digest of Technical Papers., 1989 IEEE International Conference on*, 1989, pp. 316–319.

[4] V. Ciriani, "Synthesis of SPP Three-Level Logic Networks using Affine Spaces," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 22, no. 10, pp. 1310–1323, 2003.

[5] V. Ciriani and A. Bernasconi, "2-SPP: a Practical Trade-Off between SP and SPP Synthesis," in *5th International Workshop on Boolean Problems (IWSBP2002)*, 2002, pp. 133–140.

[6] D. Debnath and T. Sasao, "Minimization of AND-OR-EXOR Three-Level Networks with AND Gate Sharing," *IEICE Trans. Information and Systems*, vol. E80-D, no. 10, pp. 1001–1008, 1997.

[7] ——, "Multiple–Valued Minimization to Optimize PLAs with Output EXOR Gates," in *IEEE International Symposium on Multiple-Valued Logic*, 1999, pp. 99–104.

[8] 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.

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

[10] E. Dubrova, D. Miller, and J. Muzio, "AOXMIN: A Three-Level Heuristic AND-OR-XOR Minimizer for Boolean Functions," in *3rd International Workshop on the Applications of the Reed-Muller Expansion in Circuit Design*, 1997, pp. 209–218.

[11] R.-R. Lee, J.-H. R. Jiang, and W.-L. Hung, "Bi-decomposing large boolean functions via interpolation and satisfiability solving," in *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*. IEEE, 2008, pp. 636–641.

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

[13] A. Malik, D. Harrison, and R. Brayton, "Three-Level Decomposition with Application to PLDs," in *IEEE International Conference on Computer Design: VLSI in Computer & Processors, ICCD '91*, 1991, pp. 628–633.

[14] A. Mishchenko, B. Steinbach, and M. Perkowski, "An algorithm for bi-decomposition of logic functions," in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 103–108.

[15] A. Mishchenko, R. Brayton, and S. Chatterjee, "Boolean factoring and decomposition of logic networks," in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2008, pp. 38–44.

[16] M. Perkowski and S. Grygiel, "A Survey of Literature on Function Decomposition," PSU Electrical Engineering Department, Technical Report November 20, 1995, 1995.

[17] R. Rudell and A. Sangiovanni-Vincentelli, "Multiple Valued Minimization for PLA Optimization," *IEEE Transactions on CAD*, vol. 6, no. 5, pp. 727–750, 1987.

[18] T. Sasao, "Input Variable Assignment and Output Phase Optimization of PLA's," *Computers, IEEE Transactions on*, vol. C-33, no. 10, pp. 879–894, Oct 1984.

[19] ——, "On the Complexity of Three-Level Logic Circuits," in *Int. Workshop on Logic Synthesis*, 1989.

[20] ——, "OR-AND-OR Three-Level Networks," in *Representation of Discrete Functions*, T. Sasao and M. Fujita, Eds. Kluwier Academic, 1996.

[21] T. Sasao and J. Butler, "On bi-decomposition of logic functions," in *Int. Workshop on Logic Synthesis*, 1997.

[22] C. Scholl, *Functional Decomposition with Applications to FPGA Synthesis*. Springer, 2002.

[23] C. Yang and M. Ciesielski, "Bds: a bdd-based logic optimization system," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 21, no. 7, pp. 866–876, 2002.

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