Computing the full quotient in bi-decomposition by approximation

Anna Bernasconi
Department of Computer Science
Università di Pisa, Italy
anna.bernasconi@unipi.it

Valentina Ciriani
Department of Computer Science "Giovanni Degli Antoni"
Università degli Studi di Milano, Italy
valentina.ciriani@unimi.it

Jordi Cortadella
Department of Computer Science
Universitat Politècnica de Catalunya, Barcelona, Spain
jordi.cortadella@upc.edu

Tiziano Villa
Department of Computer Science
Università degli Studi di Verona, Italy
tiziano.villa@univr.it

***Abstract*—Bi-decomposition is a design technique widely used to realize logic functions by the composition of simpler components. It can be seen as a form of Boolean division, where a given function is split into a divisor and quotient (and a remainder, if needed). The key questions are how to find a good divisor and then how to compute the quotient. In this paper we choose as divisor an approximation of the given function, and characterize the incompletely specified function which describes the full flexibility for the quotient. We report at the end preliminary experiments for bi-decomposition based on two AND-like operators with a divisor approximation from 1 to 0, and discuss the impact of the approximation error rate on the final area of the components in the case of synthesis by three-level XOR-AND-OR forms.**

I. Introduction

Decomposition is a design paradigm to partition logic functions into a composition of simpler components. Bi-decomposition [6], [8], [11] is a well-known form of decomposition, where a Boolean function f is rewritten as $f = g \,\mathtt{op}\, h$, and $\mathtt{op}$ is a two-input binary operator. When the operator is conjunction, bi-decomposition can be seen as a form of Boolean division [10]: $f = g \cdot h$, where f, g, h are respectively the dividend, divisor and quotient function (a reminder function r may be introduced to compensate the two sides). Given f, in general the problem is to find a good divisor g and then to determine the quotient h, where $f \subseteq g$ and $f \subseteq h$ are necessary conditions on g and h. In the case of conjunctive decomposition, it was pointed out in [6] that one could choose as divisors over-approximations g of f, and derive the associated conjuncts h by Boolean minimization using the don't care set (dc-set) derived from the divisors g. Therefore, one could obtain a sequence of decompositions $f = g_i \cdot h_i, i = 0, \ldots, n$, by pairs of divisors and quotients, in which the logic is shifted between g_i and h_i, from $g_0 = f, h_0 = 1$ to $g_n = 1, h_n = f$, choosing the best trade-off according to some objective function.

In this paper, we address the general case of bi-decomposition by approximation, $f = g \,\mathtt{op}\, h$, where g is an approximation of f. The binary operation $\mathtt{op}$ used in the decomposition determines whether we can use over-approximations or under-approximations of f as divisors g. In fact, depending on $\mathtt{op}$, g must be an approximation for f or for its complement $\overline{f}$, and it can introduce only one kind of errors: only 0 to 1 complementations of the output bits, or only 1 to 0 complementations. Both kinds of errors can be introduced only in bi-decompositions based on the XOR operation.

Given a function f, an operator $\mathtt{op}$ and a divisor g which is an approximation of f, here we solve the question to find the complete flexibility of the quotient function h, by expressing it as an incompletely specified function h with the smallest on-set and the largest dc-set such that $f = g \,\mathtt{op}\, h$.

Depending on the operation used in the bi-decomposition of f, the on-set or the off-set of g become the dc-set for the function h, together with the dc-set of f, giving a flexibility that could be exploited to get a more compact representation of f. The larger is the on (or off)-set of g, the higher is the flexibility in the implementation of h. We could say that we use h to correct the errors introduced by the approximation g, so that $g \,\mathtt{op}\, h$ is an exact alternative representation of f.

This result can be used in optimization frameworks whose objective is to explore different realizations of a function f by bi-decomposition when the divisor is an under (or over)-approximation of f. Moreover, it could be used also when the original f could be implemented approximately within a tolerable error rate (as mentioned in the conclusions). As a proof-of-concept, we report preliminary experiments for bi-decomposition based on two AND-like operators with a divisor approximation from 1 to 0, and discuss the impact of the approximation error rate on the final area of the components in the case of synthesis by three-level XOR-AND-OR forms.

In Section II we introduce some basic notation, in Section III we state and prove the key results on the full quotient flexibility for all 10 non-trivial two-input Boolean operators (divided in three classes: AND-like, OR-like, XOR-like). We discuss the preliminary experiments in Section IV, and sum up conclusions and future work in Section V.

978-3-9819263-4-7/DATE20/©2020 EDAA

TABLE I
THE TEN BINARY OPERATION DEPENDING ON BOTH INPUT VARIABLES.

Operator	Bi-decomposed form
AND	$f = g \cdot h$
$\not\Leftarrow$	$f = \overline{g} \cdot h$
$\not\Rightarrow$	$f = g \cdot \overline{h}$
NOR	$f = \overline{g + h} = \overline{g} \cdot \overline{h}$
OR	$f = g + h$
$\Rightarrow$	$f = \overline{g} + h$
$\Leftarrow$	$f = g + \overline{h}$
NAND	$f = \overline{g \cdot h} = \overline{g} + \overline{h}$
XOR	$f = g \oplus h$
XNOR	$f = \overline{g \oplus h} = g \overline{\oplus} h$

II. PRELIMINARIES

Let f be an incompletely specified function, depending on n binary variables, and let g be a completely specified approximation of f. We can classify the approximation g depending on the errors introduced, as follows.

Definition 1: The function g is a $0 \rightarrow 1$ *approximation* (*over-approximation*) of f if it is derived by a 0 to 1 complementation of some output bits of f, i.e., by moving some off-set minterms of f to the on-set, while there are no restrictions on the don't cares of f. In this case, it holds that $f^{on} \subseteq g^{on}$.

Definition 2: The function g is a $1 \rightarrow 0$ *approximation* (*under-approximation*) of f, if it is derived by a 1 to 0 complementation of some output bits of f, i.e., by moving some on-set minterms of f to the off-set, while there are no restrictions on the don't cares of f. In this case, it holds $g^{on} \subseteq f^{on}$.

Definition 3: The function g is a $0 \leftrightarrow 1$ *approximation* of f, if it is derived by both 0 to 1 and 1 to 0 complementations of some output bits of f.

III. BI-DECOMPOSITION WITH APPROXIMATION

Let $f : \{0,1\}^n \rightarrow \{0,1,-\}$ be the function that must be synthesized. Let g be a (completely specified) approximation of f or of its complement $\overline{f}$. Given f and g, and a two-input Boolean operator $\circ\!\!\mathrm{p}$, we want to compute an incompletely specified Boolean function h such that f can be represented in bi-decomposed form as $f = g \circ\!\!\mathrm{p}\ h$.

In our analysis, we only consider the ten (out of sixteen) binary operations depending on both input variables, described in Table I. Thus, we will not consider the two constant operations, as well as the four degenerate operations depending on only one input.

Applying the De Morgan's laws as shown in the table, we can note how the ten operations can be naturally divided into three sets:

- the set of the four operations based on the binary AND applied to g, h or to their complements (AND, $\not\Leftarrow$, $\not\Rightarrow$, NOR);
- the set of the four operations based on the binary OR applied to g, h or to their complements (OR, $\Rightarrow$, $\Leftarrow$, NAND);
- the set containing the two operations based on the exclusive OR (XOR, XNOR).

We now describe how to approximate f with g, and how to derive the quotient function h for each set of operations.

In the following exposition, we use f^{on}, g^{on}, and h^{on} to denote the on-sets of the three functions f, g, and h, f^{off}, g^{off}, and h^{off} to denote their off-sets, and f^{dc} and h^{dc} to denote the dc-sets of the incompletely specified functions f and h (observe that g is completely specified).

A. Decompositions based on AND, $\not\Leftarrow$, $\not\Rightarrow$, NOR

Let us first consider the AND binary operation. In order to represent f as $g \cdot h$ it is necessary that

- $f \subseteq g$,
- $f \subseteq h \subseteq f + \overline{g}$.

In fact, both g and h must be equal to 1 on the on-set minterms of f. Moreover, h can be equal to 1 on all off-set minterms of g, while it must get the value 0 where f evaluates to 0 and g to 1. Finally, h can get any value on the dc-set minterms of f, independently of the value of g. This, in turns, implies that

(i) g must be a $0 \rightarrow 1$ approximation of f, so that $f^{on} \subseteq g^{on}$, while there are no restrictions on the don't cares of f.

(ii) h is the incompletely specified function whose on-set is equal to the on-set of f, while the dc-set contains all off-set minterms of g and all dc-set minterms of f, i.e.,

$$h^{on} = f^{on}$$
$$h^{dc} = g^{off} \cup f^{dc}$$
$$h^{off} = g^{on} \cap f^{off} = g^{on} \setminus (f^{on} \cup f^{dc}).$$

Note that h^{off} coincides precisely with the set of minterms on which f^{on} and g^{on} differ, i.e., it describes the error introduced by the approximation. This implies that the more accurate is the approximation g, the smaller is the off-set of the function h and the largest is h^{dc}, thus providing a high flexibility that can be exploited to get a compact representation for the function h, and consequently, for the target function f.

We prove the correctness of our analysis in the following lemma.

Lemma 1: Let f be an incompletely specified function depending on n binary variables, let g be a completely specified $0 \rightarrow 1$ approximation of f, and let h be an incompletely specified function satisfying $h^{on} = f^{on}$ and $h^{dc} = g^{off} \cup f^{dc}$. Then $f = g \cdot h$.

Proof. We first show that for any minterm $w \in f^{on}$, $g(w) \cdot h(w) = 1$. Observe that $f^{on} \subseteq g^{on}$, as g is a $0 \rightarrow 1$ approximation of f. Moreover $h^{on} = f^{on}$ by hypothesis. Thus, $w \in g^{on}$ and $w \in h^{on}$, and we immediately have $g(w) \cdot h(w) = 1$.

Now suppose that $w \in f^{off}$. If $w \in g^{on}$, then w belongs to the off-set of h by construction (as it can be neither in h^{on} nor in h^{dc}), and we have $g(w) \cdot h(w) = 0$. Otherwise, if $w \in g^{off}$, then, independently of the value of h on w, we have $g(w) \cdot h(w) = 0$, and the thesis follows. ■

From the proof of this lemma it follows that h is the quotient function that guarantees the maximum flexibility in the decomposition:

Corollary 1: The function h with on-set $h^{on} = f^{on}$ and dc-set $h^{dc} = g^{off} \cup f^{dc}$ is the quotient function with the smallest on-set and the biggest dc-set satisfying $f = g \cdot h$.

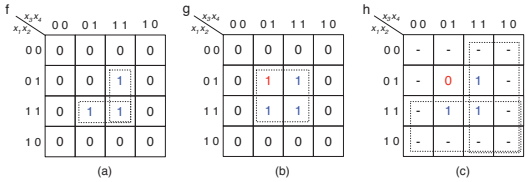

Fig. 1. Karnough map for the function $f = x_1x_2x_4 + x_2x_3x_4$ (a); for its approximation $g = x_2x_4$ (b); and for the function $h = x_1 + x_3$ such that $f = g \cdot h$.

Example 1: To provide more intuition for the proposed approach, we discuss a simple example for the AND bi-decomposition $f = g \cdot h$ of the function f represented by the Karnaugh map in Figure 1 (a). A minimal SOP representation of f is given by $f^{SOP} = x_1x_2x_4 + x_2x_3x_4$, that has 6 literals. A $0 \rightarrow 1$ approximation of f can be obtained by simply adding the minterm $\overline{x}_1x_2\overline{x}_3x_4$ to the on-set of f. In this way we obtain the function g depicted in Figure 1 (b), that has a SOP representation $g^{SOP} = x_2x_4$ containing only two literals. Applying Lemma 1, we then derive the function h, described in Figure 1 (c). Thanks to the large dc-set of h, we can obtain a very compact SOP representation for h, with only two literals: $h^{SOP} = x_1 + x_3$. The overall bi-decomposed form is given by $f = g \cdot h = x_2x_4 \cdot (x_1 + x_3)$, with 4 literals.

The function h for the bi-decomposition of f with respect to the $\not\Leftarrow$, $\not\Rightarrow$, and NOR operations can be derived in a similar way, applying the previous considerations to $\overline{g}$ and h, g and $\overline{h}$, and to $\overline{g}$ and $\overline{h}$, respectively. The definitions of the on-, off-, and dc-set of the function h for all three cases are reported in Table II, and their correctness is proved in the following lemma.

Lemma 2: Let f be an incompletely specified function depending on n binary variables, and let g be a completely specified approximation of f.

1) If g is a $1 \rightarrow 0$ approximation of $\overline{f}$ and h is an incompletely specified function satisfying $h^{on} = f^{on}$ and $h^{dc} = g^{on} \cup f^{dc}$, then $f = \overline{g} \cdot h$, i.e., $f = (g \not\Leftarrow h)$.
2) If g is a $0 \rightarrow 1$ approximation of f and h is an incompletely specified function satisfying $h^{on} = f^{off} \setminus g^{off}$ and $h^{dc} = g^{off} \cup f^{dc}$, then $f = g \cdot \overline{h}$, i.e., $f = (g \not\Rightarrow h)$.
3) If g is a $0 \rightarrow 1$ approximation of $\overline{f}$ and h is an incompletely specified function satisfying $h^{on} = f^{off} \setminus g^{on}$ and $h^{dc} = g^{on} \cup f^{dc}$, then $f = \overline{g} \cdot \overline{h}$, i.e., $f = g$ NOR h.

Proof. We only prove the correctness of the decomposition based on $\not\Leftarrow$. The correctness of the other two decompositions can be proved in a similar way.

First, suppose that $w \in f^{on}$. The fact that g is a $1 \rightarrow 0$ approximation of $\overline{f}$ implies that some on-set minterms of $\overline{f}$ have been moved to its off-set, so that $g^{on} \subseteq \overline{f}^{on}$, i.e., $g^{on} \subseteq f^{off}$. This in turns implies that $w \notin g^{on}$. Moreover, since $h^{on} = f^{on}$, we have that $w \in h^{on}$, and we immediately derive that $\overline{g(w)} \cdot h(w) = 1$.

Now suppose that $w \in f^{off}$. Since $g^{on} \subseteq f^{off}$, w might belong either to g^{on} or to g^{off}. In the first case, we immediately have that $\overline{g(w)} \cdot h(w) = 0$, independently of the value of h on w. Otherwise, if $w \in g^{off}$, then w belongs to h^{off} by construction. In fact, w can be neither in h^{on}, which is equal to f^{on}, nor in h^{dc}, which is equal to $g^{on} \cup f^{dc}$. Thus, we have $\overline{g(w)} \cdot h(w) = 0$, and the thesis follows. ■

As before, the functions h used in the bi-decompositions guarantee the maximum flexibility thanks to the definition of their dc-sets:

Corollary 2: The functions h defined as in Lemma 2 are the functions with the smallest on-set and the biggest dc-set satisfying the bi-decompositions with respect to the $\not\Leftarrow$, $\not\Rightarrow$, and NOR operations.

From Table II we can observe that, depending on the specific operation used in the decomposition, h^{on} or h^{off} describe the differences between f or $\overline{f}$ and their approximation g. Thus, the more accurate is the approximation g, the smaller will be h^{on} or h^{off}.

B. Decompositions based on OR, $\Rightarrow$, $\Leftarrow$, NAND

We first consider the OR binary operation. In order to represent f as $g+h$ it is necessary that $g \subseteq f$ and $f \setminus g \subseteq h \subseteq f$. In fact, both g and h must be equal to 0 on the off-set minterms of f, otherwise $g + h$ would be equal to 1. Moreover, h must be equal to 1 on all on-set minterms of f where the approximation g is equal to 0, while it can get any value on the dc-set minterms of f, independently of the value of g. This implies that

(i) g is a $1 \rightarrow 0$ approximation of f, so that $g^{on} \subseteq f^{on}$. As before, g can take either the value 0 or the value 1 on the don't cares of f.
(ii) h is the incompletely specified function whose on-set is equal to set difference between the on-sets of f and g, while the dc-set contains all on-set minterms of g and all dc-set minterms of f, i.e.,

$$h^{on} = f^{on} \setminus g^{on}$$
$$h^{dc} = g^{on} \cup f^{dc}$$
$$h^{off} = f^{off}.$$

Observe that, in this case, h^{on} describes the error introduced by the approximation g, while h^{off} coincides with the off-set of the target function f. So, if the quality of the approximation is good, h will have a limited number of on-set minterms, and a dc-set bigger than f^{dc}.

We prove the correctness of our analysis in the following lemma.

Lemma 3: Let f be an incompletely specified function depending on n binary variables, let g be a completely specified $1 \rightarrow 0$ approximation of f, and let h be an incompletely specified function satisfying $h^{on} = f^{on} \setminus g^{on}$ and $h^{dc} = g^{on} \cup f^{dc}$. Then $f = g + h$.

Proof. The proof is similar to the proof of correctness of the decomposition with the AND operator. A complete proof will be added in the extended version of the paper. ■

Even in this case, from the proof of the lemma it follows that h is the function that guarantees the maximum flexibility in the decomposition:

TABLE II
FUNCTIONS g AND h OCCURRING IN THE BI-DECOMPOSED FORMS BASED ON THE TEN BINARY OPERATIONS DEPENDING ON BOTH INPUTS.

Operator	Bi-decomposed form	Approximation function g	$\mathbf{h}^{on}$	$\mathbf{h}^{dc}$	$\mathbf{h}^{off}$
AND	$f = g \cdot h$	$0 \to 1$ approximation of f (i.e., $f^{on} \subseteq g^{on}$)	f^{on}	$g^{off} \cup f^{dc}$	$g^{on} \setminus f^{on}$
$\not\Leftarrow$	$f = \overline{g} \cdot h$	$1 \to 0$ approximation of $\overline{f}$ (i.e., $g^{on} \subseteq f^{off}$)	f^{on}	$g^{on} \cup f^{dc}$	$g^{off} \setminus f^{on}$
$\not\Rightarrow$	$f = g \cdot \overline{h}$	$0 \to 1$ approximation of f (i.e., $f^{on} \subseteq g^{on}$)	$f^{off} \setminus g^{off}$	$g^{off} \cup f^{dc}$	f^{on}
NOR	$f = \overline{g} \cdot \overline{h}$	$1 \to 0$ approximation of $\overline{f}$ (i.e., $g^{on} \subseteq f^{off}$)	$f^{off} \setminus g^{on}$	$g^{on} \cup f^{dc}$	f^{on}
OR	$f = g + h$	$1 \to 0$ approximation of f (i.e., $g^{on} \subseteq f^{on}$)	$f^{on} \setminus g^{on}$	$g^{on} \cup f^{dc}$	f^{off}
$\Rightarrow$	$f = \overline{g} + h$	$0 \to 1$ approximation of $\overline{f}$ (i.e., $f^{off} \subseteq g^{on}$)	$f^{on} \setminus g^{off}$	$g^{off} \cup f^{dc}$	f^{off}
$\Leftarrow$	$f = g + \overline{h}$	$1 \to 0$ approximation of f (i.e., $g^{on} \subseteq f^{on}$)	f^{off}	$g^{on} \cup f^{dc}$	$f^{on} \setminus g^{on}$
NAND	$f = \overline{g} + \overline{h}$	$0 \to 1$ approximation of $\overline{f}$ (i.e., $f^{off} \subseteq g^{on}$)	f^{off}	$g^{off} \cup f^{dc}$	$g^{on} \setminus f^{off}$
XOR	$f = g \oplus h$	$0 \leftrightarrow 1$ approximation of f	$f^{on} \oplus g^{on}$	f^{dc}	$f^{on} \oplus g^{off}$
XNOR	$f = g \overline{\oplus} h$	$0 \leftrightarrow 1$ approximation of $\overline{f}$	$f^{off} \oplus g^{on}$	f^{dc}	$f^{off} \oplus g^{off}$

Corollary 3: The function h with on-set $h^{on} = f^{on}$ and dc-set $h^{dc} = g^{off} \cup f^{dc}$ is the function with the smallest on-set and the biggest dc-set satisfying $f = g \cdot h$.

The function h for the bi-decomposition of f with respect to the $\Rightarrow$, $\Leftarrow$, and NAND operators can be derived in a similar way, applying the previous considerations to $\overline{g}$ and h, g and $\overline{h}$, and to $\overline{g}$ and $\overline{h}$, respectively. The definitions of the on-, off-, and dc-set of the function h for these cases are shown in Table II, and the correctness is proved in the following lemma.

Lemma 4: Let f be an incompletely specified function depending on n binary variables, and let g be a completely specified approximation of f.

1) If g is a $0 \to 1$ approximation of $\overline{f}$ and h is an incompletely specified function satisfying $h^{on} = f^{on} \setminus g^{off}$ and $h^{dc} = g^{off} \cup f^{dc}$, then $f = \overline{g} + h$, i.e., $f = (g \Rightarrow h)$.
2) If g is a $1 \to 0$ approximation of f and h is an incompletely specified function satisfying $h^{on} = f^{off}$ and $h^{dc} = g^{on} \cup f^{dc}$, then $f = g + \overline{h}$, i.e., $f = (g \Leftarrow h)$.
3) If g is a $0 \to 1$ approximation of $\overline{f}$ and h is an incompletely specified function satisfying $h^{on} = f^{off}$ and $h^{dc} = g^{off} \cup f^{dc}$, then $f = \overline{g} + \overline{h}$, i.e., $f = g$ NAND h.

Proof. The proof is similar to the proof of correctness of the AND-based decompositions. A complete proof will be provided in the extended version of the paper. ■

As before, the functions h guarantee the maximum flexibility thanks to the definition of their dc-sets:

Corollary 4: The functions h defined as in Lemma 4 are the functions with the smallest on-set and the biggest dc-set satisfying the bi-decompositions with respect to the $\Rightarrow$, $\Leftarrow$, and NAND operations.

C. Decompositions based on XOR and XNOR

If we want to represent f as the XOR between its approximation g and the function h, so that $f = g \oplus h$, by the linearity of the XOR operator we immediately derive that h must be defined as $h = f \oplus g$. Indeed, where f and g assume the same (specified) value, h must evaluate to 0, while h must be equal to 1 where f and g differs. Finally, h can get any value on the dc-set set of f, independently of the value of g. Thus,

(i) g can be any approximation of f, derived by both $0 \to 1$ and $1 \to 0$ complementations of some output bits of f.

(ii) h^{on} describes the error introduced by the approximation:

$$h^{on} = f^{on} \oplus g^{on}$$
$$h^{dc} = f^{dc}$$
$$h^{off} = f^{on} \,\overline{\oplus}\, g^{on} = f^{on} \oplus g^{off}.$$

Analogously, we can observe that in order to represent f as f XNOR g, it is necessary that

$$h^{on} = f^{on} \,\overline{\oplus}\, g^{on} = f^{off} \oplus g^{on}$$
$$h^{dc} = f^{dc}$$
$$h^{off} = f^{off} \,\overline{\oplus}\, g^{on} = f^{on} \oplus g^{on}.$$

In this case, g is a $0 \leftrightarrow 1$ approximation of $\overline{f}$, whose errors are described by h^{off}. We conclude proving the correctness of our approach for these last two binary operations.

Lemma 5: Let f be an incompletely specified function depending on n binary variables, and let g be a completely specified approximation of f.

1) If g is a $0 \leftrightarrow 1$ approximation of f, and h is an incompletely specified function s.t. $h^{on} = f^{on} \oplus g^{on}$ and $h^{dc} = f^{dc}$, then $f = g \oplus h$, i.e., $f = g$ XOR h.
2) If g is a $0 \leftrightarrow 1$ approximation of $\overline{f}$, and h is an incompletely specified function satisfying $h^{on} = f^{off} \oplus g^{on}$ and $h^{dc} = f^{dc}$, then $f = g \,\overline{\oplus}\, h$, i.e., $f = g$ XNOR h.

Proof. The correctness of these decomposition follows immediately from the linearity of the XOR and XNOR operations, and from the hypothesis on h. Indeed, for any minterm w in f^{on} or in f^{off} we have:

$$g(w) \oplus h(w) = g(w) \oplus (f(w) \oplus g(w)) = f(w)$$
$$g(w) \,\overline{\oplus}\, h(w) = g(w) \,\overline{\oplus}\, (f(w) \,\overline{\oplus}\, g(w)) = f(w).$$

■

The overall results of this analysis, together with the definitions of the on-, off-, and dc-set of the function h are summarized in Table II.

IV. EXPERIMENTAL RESULTS

The method described in the previous sections is very general and can be applied to any approximation technique. In order to experimentally study the proposed approach, we discuss its application to a $0 \to 1$ known approximation method [2]. Thus, in this section we discuss the experimental results obtained by applying the techniques that require a $0 \to 1$ approximation of f (i.e., AND and $\not\Rightarrow$, see Table II).

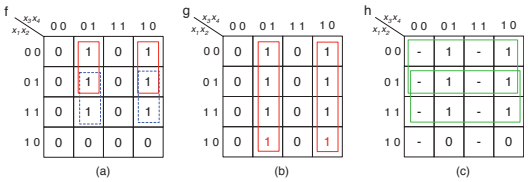

Fig. 2. Karnough maps for the function $f = \overline{x}_1(x_3 \oplus x_4) + x_2(x_3 \oplus x_4)$ (a), its approximation $g = (x_3 \oplus x_4)$ (b), and the quotient function $h = \overline{x}_1 + x_2$. The pseudoproduct $\overline{x}_1(x_3 \oplus x_4)$ in (a) and its expansion in (b) are represented with a solid red line. The pseudoproduct $x_2(x_3 \oplus x_4)$ is represented in (a) with a dashed blue line. Finally, a cover for h is represented in (c) with a solid green line.

A. Approximation method: computation of g

We applied our decomposition method to the synthesis of Boolean functions in three-level *Sum of Pseudoproducts* or SPP forms. SPP networks are three-level XOR-AND-OR forms introduced in [7] as a direct generalization of SOP forms. They are obtained generalizing cubes to *pseudocubes* where literals in cubes may be replaced by *XOR factors* (i.e., exclusive or of literals) in pseudocubes. Since XOR gates are sensitive to some structural regularities of Boolean functions that are difficult to express using just AND and OR gates, they help when minimizing to derive more compact expressions. For technological reasons, SPP forms are further restricted to XOR factors with at most k literals [4], [5]; in particular, the approximation heuristic proposed in [2] and applied in this paper refers to SPP forms with $k = 2$, called 2-SPP forms [5], which exhibit a compact area, reduced delay (due to bounded number of levels), and reasonable synthesis time [1]. As an example, the function f represented by the Karnaugh map in Figure 2 (a) has a minimal SOP $f^{SOP} = \overline{x}_1\overline{x}_3x_4 + \overline{x}_1x_3\overline{x}_4 + x_2\overline{x}_3x_4 + x_2x_3\overline{x}_4$ with 4 products and 12 literals, vs. a minimal 2-SPP expression $f^{2-SPP} = \overline{x}_1(x_3 \oplus x_4) + x_2(x_3 \oplus x_4)$, with only 2 pseudoproducts and 6 literals.

2-SPP forms can be over-approximated, as studied in [2], by expanding heuristically pseudoproducts by applying the following two steps: *(i)* evaluate the cost (i.e., the number of $0 \rightarrow 1$ complementations) and the gain of the expansion of each pseudoproduct in an initial 2-SPP cover of f; and *(ii)* select the set of pseudoproducts whose expansion guarantees the maximum gain within a given error rate. As a fact, the expansion of pseudoproducts implies a reduction of literals in the algebraic expression; moreover, expanded pseudoproducts might cover some other pseudoproducts, which are removed from the cover. Consider again the function f in Figure 2 (a), and suppose to expand the first pseudoproduct $\overline{x}_1(x_3 \oplus x_4)$ removing the literal $\overline{x}_1$. This introduces two $0 \rightarrow 1$ complementations, as the two off-set minterms $x_1\overline{x}_2\overline{x}_3x_4$ and $x_1\overline{x}_2x_3\overline{x}_4$ are moved to the on-set (see the red minterms in Figure 2 (b)). The new pseudoproduct covers entirely the other pseudoproduct $x_2(x_3 \oplus x_4)$ that can be then removed from the initial cover. In this way, we derive an over-approximation g of f whose 2-SPP form consists in just one pseudoproduct: $g^{2-SPP} = x_3 \oplus x_4$.The quotient h has the same on-set of f, it is 0 on the minterms $x_1\overline{x}_2\overline{x}_3x_4$ and $x_1\overline{x}_2x_3\overline{x}_4$, and it is don't care otherwise, yielding $h^{2-SPP} = \overline{x}_1 + x_2$, as shown in Figure 2 (c).

Since in the present study we use an approximation of f to derive an *exact* bi-decomposed representation, we computed g in a slightly different way. Instead of setting an error rate, which is not crucial for our purposes, we allowed more freedom to the 2-SPP minimizer: we expanded all pseudoproducts in an initial 2-SPP cover, moving off-set minterms involved in the expansion to the dc-set. Then, we re-synthesized the function with the extended dc-set. By so doing, the actual error rate of the approximation g depends on the benchmark.

B. Computation of h and bi-decomposition's analysis

We study the performance of the proposed approach by comparing the 2-SPP form of the given function f to the AND (and $\not\Rightarrow$) of the 2-SPP forms for g and h. We considered several benchmarks with different error rates from the collection [12], described in Tables III and IV, where due to space limits we report only a significant subset of the functions as representative indicators of our experiments. The algorithms were implemented in C, using the CUDD library for BDDs to represent Boolean functions. The computational experiments were performed on a Linux Intel Core i7-7700 CPU with 8 GB of RAM.

In the experiments, the bi-decomposition of f is derived as follows: 1) The 2-SPP over-approximation (g) of f is computed following the strategy described in Section IV-A, then g is minimized in 2-SPP form; 2) the on-set and dc-set of h are computed using f and g by the formulas in Table II for AND (resp., $\not\Rightarrow$) with OBDD operations; 3) h is minimized in 2-SPP form; 4) the bi-decomposition of f is computed as AND (resp., $\not\Rightarrow$) of the two 2-SPP forms for g and h.

The theoretical results in Section III suggest that this method should be applied when the approximation g of f is particularly compact and the error rate is not too high. The experimental results confirm this observation: in fact, we obtained interesting results when the error rate is less than 10% as shown in Table III. Moreover, for another subset of benchmarks, also very high error rates between 40% and 50% yielded good results, reported in Table IV. In some cases the resulting gain is high, probably due to the fact that the area of g becomes very small, and the number of don't cares of h is enough to guarantee its compact representation.

The first column in Tables III and IV reports the name of the instance and its input and output size. The following column shows the time (in seconds) required for the construction of g and h. The next two columns report the area of the 2-SPP forms for f and g estimated with the SIS tool [9] after technology mapping (mcnc.genlib). The following column describes the effective error rate of the approximated function g. The next column reports the gain in area of g with respect to f. The following couples of columns report the area and gain in area of (g AND h) and ($g \not\Rightarrow h$), respectively. The area of h can be directly inferred as the difference between the area of the bi-decompositions and of g. Finally, we note that the

TABLE III
EXPERIMENTAL COMPARISON OF BENCHMARKS WITH ERROR RATE LESS THAN 10%.

Benchmark	Time (s)	Area f	Area g	%Errors	%(Area f -Area g)/Area f	AreaAND	Gain AND (%)	Area $\not\Rightarrow$	Gain $\not\Rightarrow$ (%)
bcb (26/39)	1.20	**4662**	4154	0.1	10.90	4855	-4.14	4800	-2.96
br1 (12/8)	0.04	384	356	0.35	7.29	**370**	3.65	**370**	3.65
br2 (12/8)	0.04	275	250	0.38	9.09	**263**	4.36	**263**	4.36
mp2d (14/14)	0.09	**204**	65	3.73	68.14	210	-2.94	210	-2.94
alcom (15/38)	0.19	**210**	140	4.93	33.33	**210**	0.00	**210**	0.00
spla (16/46)	0.39	**1792**	1394	5.01	22.21	1919	-7.09	1931	-7.76
al2 (16/47)	0.59	**328**	226	5.03	31.10	340	-3.66	342	-4.27
ex5 (8/63)	0.12	935	206	5.52	77.97	925	1.07	**907**	2.99
newtpla2 (10/4)	0.01	56	19	5.62	66.07	**55**	1.79	**55**	1.79
ts10 (22/16)	0.67	**901**	609	5.76	32.41	1153	-27.97	1173	-30.19
chkn (29/7)	0.25	**744**	370	5.78	50.27	995	-33.74	971	-30.51
opa (17/69)	0.49	**1566**	1482	8.09	5.36	1578	-0.77	1578	-0.77
b7 (8/31)	0.10	198	146	8.52	26.26	197	0.51	**194**	2.02
risc (8/31)	0.08	204	150	8.62	26.47	203	0.49	**200**	1.96

TABLE IV
EXPERIMENTAL COMPARISON OF BENCHMARKS WITH ERROR RATE MORE THAN 40%.

Benchmark	Time (s)	Area f	Area g	%Errors	%(Area f -Area g)/Area f	AreaAND	Gain AND (%)	Area $\not\Rightarrow$	Gain $\not\Rightarrow$ (%)
dist (8/5)	0.03	**669**	77	40.62	88.49	736	-10.01	718	-7.32
max512 (9/6)	0.01	817	3	43.23	99.63	769	5.88	**745**	8.81
ex7 (16/5)	0.05	**192**	32	43.51	83.33	338	-76.04	386	-101.04
z4 (7/4)	0.01	140	3	43.75	97.86	**135**	3.57	136	2.86
clip (9/5)	0.03	430	24	44.65	94.42	142	66.98	**47**	89.07
max1024 (10/6)	0.03	1362	48	44.79	96.48	946	30.54	**838**	38.47
adr4 (8/5)	0.02	**180**	27	45.00	85.00	223	-23.89	215	-19.44
radd (8/5)	0.00	**119**	3	45.62	97.48	144	-21.01	141	-18.49
add6 (12/7)	0.05	**292**	3	46.54	98.97	402	-37.67	401	-37.33
log8mod (8/5)	0.01	237	11	47.50	95.36	**219**	7.59	221	6.75
Z5xp1 (7/10)	0.01	273	10	48.91	96.34	271	0.73	**265**	2.93

behavior of the two considered bi-decomposed forms is quite similar, i.e., we can observe how the gain in area occurs almost always for the same benchmarks.

V. CONCLUSIONS

In this paper we described and analyzed an approach to Boolean function exact bi-decomposition based on approximation, and we reported its application to a $0 \rightarrow 1$ known approximation method, for the two operators AND and $\not\Rightarrow$. Future work includes a wider experimental investigation, considering all binary operators and all approximations: $0 \rightarrow 1$, $1 \rightarrow 0$, and $0 \leftrightarrow 1$; we plan also to extend it to other approximation frameworks, e.g., approximated SOP forms.

We are also interested in the investigation of bi-decomposition combined with approximation methods that derive the *closest approximate regular* version of a given Boolean function, as proposed for instance in [3]. Indeed, the *approximation toward regularity* approach presents a drawback: it may introduce an unbounded number of errors, producing a low-quality approximation g. Our approach, however, can overcome this problem, thanks to the presence of the function h, whose aim is precisely to correct the errors introduced by the function g. Moreover, we could derive h, and then exploit it to correct partially instead than totally the errors introduced by g. In other words, we could use h to correct the unbounded 'regular' approximation g, and then approximate h, in any bounded-error approximation framework, in order to derive an overall compact approximation of the original function f, with a bounded number of errors.

REFERENCES

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

[2] A. Bernasconi and V. Ciriani, "2-SPP Approximate Synthesis for Error Tolerant Applications," in *17th Euromicro Conference on Digital System Design, DSD 2014, Verona, Italy, August 27-29, 2014*, 2014, pp. 411–418.

[3] A. Bernasconi, V. Ciriani, and T. Villa, "Approximate logic synthesis by symmetrization," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2019, Florence, Italy, March 25-29, 2019*, 2019, pp. 1655–1660.

[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] J. Cortadella, "Timing-Driven Logic Bi-Decomposition," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 675–685, 2003.

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

[8] A. Mishchenko, B. Steinbach, and M. Perkowski, "An Algorithm for Bi-Decomposition of Logic Functions," in *ACM/IEEE 38th Design Automation Conference (DAC)*, 2001, pp. 103–108.

[9] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," Tech. Rep. No. UCB/ERL M92/41, Berkeley, CA, Tech. Rep., 1992.

[10] T. Villa, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Synthesis of Multi-Level Boolean Networks," in *Boolean Methods and Models in Mathematics, Computer Science and Engineering, Encyclopedia of Mathematics and its Applications 134*, Y. Crama and P. L. Hammer, Eds. Cambridge University Press, 2010, pp. 675–722.

[11] S. Yamashita, H. Sawada, and A. Nagoya, "New methods to find optimal non-disjoint bi-decompositions," in *Proc. of 1998 Asia and South Pacific Design Automation Conference*, Feb 1998, pp. 59–68.

[12] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0," Microelectronic Center, User Guide, 1991.