Logic Synthesis for Switching Lattices by Decomposition with P-Circuits

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

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

Luca Frontini
Dipartimento di Informatica
Università degli Studi di Milano, Italy
luca.frontini@unimi.it

Valentino Liberali
Dipartimento di Fisica
Università degli Studi di Milano, Italy
valentino.liberali@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*—In this paper we propose a novel approach to the synthesis of minimal-sized lattices, based on the decomposition of logic functions. Since the decomposition allows to obtain circuits with a smaller area, our idea is to decompose Boolean functions with separate lattices, according to the P-circuits decomposition scheme, and then to implement the decomposed blocks with physically separated regions in a single lattice. Experimental results show that about 35% of the considered benchmarks achieve a smaller area when implemented using the proposed decomposition for switching lattices, with an average gain of at least 24%.**

I. Introduction

A switching lattice is a two-dimensional lattice of four-terminal switches linked to the four neighbors of a lattice cell, so that these are either all connected, or disconnected. A Boolean function can be implemented by a lattice associating each four-terminal switch to a Boolean literal, so that if the literal takes the value 1 the corresponding switch is ON and connected to its four neighbors, otherwise it is not connected. The function evaluates to 1 if and only if there exists a connected path between two opposing edges of the lattice, e.g., the top and the bottom edges (see Figure 1 for an example). The synthesis problem on a lattice thus consists in finding an assignment of literals to switches in order to implement a given target function with a lattice of minimal size.

The idea of using regular two-dimensional arrays of switches to implement Boolean functions is old and dates back to a seminal paper by Akers in 1972 [1]. Recently, with the advent of a variety of emerging nanoscale technologies based on regular arrays of switches, synthesis methods targeting lattices of multi-terminal switches have found a renewed interest [2], [3], [9]. Consider for instance a nanowire array, where each crosspoint is controlled by an input voltage. Each such crosspoint behaves like a four-terminal switch, and therefore the nanowire crossbar array can be modeled as a lattice of four-terminal switches. Nanowire crossbar arrays may offer substantial advantages over conventional CMOS when used to implement programmable architectures. Conventional implementations typically employ SRAMs for programming crosspoints; other techniques have been suggested for implementing programmable crosspoints such as bistable switches that form memory cores, molecular switches and solid-electrolyte nanoswitches (see [3] for more details and bibliographic references).

In this paper we will show how the cost of implementing a switching lattice could be mitigated by exploiting Boolean function decomposition techniques in lattice-based implementations. Decomposition of logic functions is a widely studied field in multi-level logic; here we focus on the particular decomposition method that gives rise to the bounded-level logic networks called *P-circuits* [4], [5], [6], [7]. The basic idea of the proposed approach is to first decompose a function into some subfunctions, according to the functional P-circuits decomposition scheme, and then to implement the decomposed blocks with physically separated regions in a single lattice. Since the decomposed blocks correspond to functions depending on fewer variables and/or with a smaller on-set, their synthesis should be easier and should produce lattice implementations of smaller area.

In the framework of switching lattices synthesis, where the available minimization tools are not yet as developed and mature as those available for CMOS technology, reducing the synthesis of a target Boolean function to the synthesis of smaller functions could represent a very beneficial approach. This expectation has been confirmed by our experimental results, which demonstrate that in about 35% of the analyzed cases the synthesis of switching lattices based on a decomposition of the logic function into smaller sub-functions allows to obtain a smaller area in the final resulting lattice.

The paper is organized as follows. Preliminaries on switching lattices and P-circuits are described in Section II. Section III shows how the P-circuit decomposition scheme can be exploited for the synthesis of switching lattices. Section IV provides the experimental results and Section V concludes the paper.

II. Preliminaries

In this section we briefly review some basic notions and results on switching lattices [1], [3], [9] and P-circuits [4], [5], [6], [7].

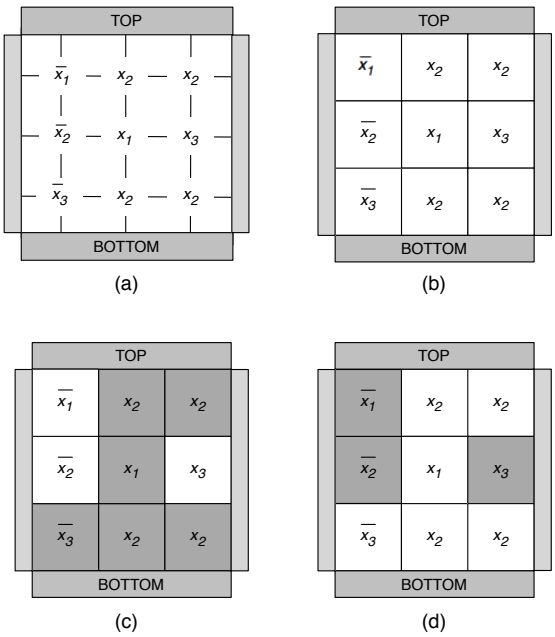

Fig. 1. A four terminal switching network implementing the function $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$ (a); its corresponding lattice form (b); the lattice evaluated on the assignments 1,1,0 (c) and 0, 0, 1 (d), with grey and white squares representing ON and OFF switches, respectively.

A. Switching Lattices

A switching lattice is a two-dimensional array of four-terminal switches. The four terminals of the switch link to the four neightbours of a lattice cell, so that these are either all connected (when the switch is ON), or disconnected (when the switch is OFF).

A Boolean function can be implemented by a lattice in terms of connectivity across it:

- each four-terminal switch is controlled by a Boolean literal;
- each switch may be also labelled with the constant 0, or 1;
- if the literal takes the value 1, the corresponding switch is connected to its four neightbours, else it is not connected;
- the function evaluates to 1 if and only if there exists a connected path between two opposing edges of the lattice, e.g., the top and the bottom edges;
- input assignments that leave the edges unconnected correspond to output 0.

For instance, the 3×3 network of switches in Figure 1 (a) corresponds to the lattice form depicted in Figure 1 (b), which implements the function $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$. If we assign the values 1, 1, 0 to the variables x_1, x_2, x_3, respectively, we obtain paths of gray square connecting the top and the bottom edges of the lattices (Figure 1 (c)), indeed on this assignment f evaluates to 1. On the contrary, the assignment $x_1 = 0, x_2 = 0, x_3 = 1$, on which f evaluates to 0, does not produce any path from the top to the bottom edge (Figure 1 (d)).

The synthesis problem on a lattice consists in finding an assignment of literals to switches in order to implement a given target function with a lattice of minimal size. The size is measured in terms of the number of switches in the lattice.

A switching lattice can similarly be equipped with left edge to right edge connectivity, so that a single lattice can implement two different functions. This fact is exploited in [2], [3] where the authors propose a synthesis method for switching lattices simultaneously implementing a function f according to the connectivity between the top and the bottom plates, and its dual function f^D according to the connectivity between the left and the right plates. Recall that the dual of a Boolean function f depending on n binary variables is the function f^D such that $f(x_1, x_2, \ldots, x_n) = \overline{f^D}(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n)$. This method produces lattices with a size that grows linearly with the number of products in an irredundant sum of product (SOP) representation of f, and consists of the following steps:

1) find an irredundant, or a minimal, SOP representation for f and f^D: $SOP(f) = p_1 + p_2 + \cdots + p_s$ and $SOP(f^D) = q_1 + q_2 + \cdots + q_r$;
2) form a $r \times s$ switching lattice and assign each product p_j $(1 \leq j \leq s)$ of $SOP(f)$ to a column and each product q_i $(1 \leq i \leq r)$ of $SOP(f^D)$ to a row;
3) for all $1 \leq i \leq r$ and all $1 \leq j \leq s$, assign to the switch on the lattice site (i, j) one literal which is shared by q_i and p_j (the fact that f and f^D are duals guarantees that such a shared literal exists for all i and j).

This synthesis algorithm thus produces a lattice for f whose size depends on the number of products in the irredundant SOP representations of f and f^D, and it comes with the dual function implemented for free. For instance, the lattice depicted in Figure 1 has been built according to this algorithm, and it implements both the function $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$ and its dual $f^D = x_1\overline{x}_2x_3 + \overline{x}_1x_2 + x_2\overline{x}_3$.

The time complexity of the algorithm is polynomial in the number of products. However, the method does not always build lattices of minimal size for every target function, since it ties the dimensions of the lattices to the number of products in the SOP forms. In particular this method is not effective for Boolean functions whose duals have a a very large number of products. Another reason that could explain the non-minimality of the lattices produced in this way is that the algorithm does not use Boolean constants as input, i.e., each switch in the lattice is always controlled by a Boolean literal.

In [9], the authors proposed a different approach to the synthesis of minimal-sized lattices, which is formulated as a satisfiability problem in quantified Boolean logic and solved by quantified Boolean formula solvers. This method uses the previous algorithm to find an upper bound on the dimensions of the lattice. It then searches for successively better implementations until either an optimal solution is found, or else a preset time limit has been exhausted. Experimental results show how this alternative method can decrease lattice sizes considerably. In this approach the use of fixed inputs is allowed.

B. P-circuits

We now review the bounded-level logic networks called *P-circuits*, a special logic architecture which realizes a Boolean

function by projecting it onto overlapping subsets. They were introduced in [4], [5], [6] and further studied in [7].

We first give some preliminary definitions. A *completely specified Boolean function* f is a function $f : \{0,1\}^n \rightarrow \{0,1\}$. A completely specified Boolean function can also be interpreted as the set of points $x \in \{0,1\}^n$ such that $f(x) = 1$. An *incompletely specified Boolean function* is a function $f : \{0,1\}^n \rightarrow \{0,1,-\}$, where $-$ is called the don't care value of the function. An incompletely specified function can be described by three sets of points: the *on-set*, the *off-set* and the *don't care set*, which characterize the points in $\{0,1\}^n$ with images 0, 1, and $-$, respectively. Given the Boolean space $\{0,1\}^n$ described by the set $\{x_1, \ldots, x_n\}$ of n binary variables, a *literal* is a variable or its complement; a *cube* is conjunction (or product) of a set of literals, and a *minterm* is a cube when it represents only one point, i.e., when it is a conjunction of n distinct literals. Finally, a *multiple-output Boolean function* f is a function $f : \{0,1\}^n \rightarrow \{0,1,-\}^m$; it can be considered also as a vector of Boolean functions $\{f_1, f_2, \ldots, f_m\}$.

P-circuits are extended forms of Shannon cofactoring, where the expansion is with respect to an orthogonal basis $\overline{x}_i \oplus p$ (i.e., $x_i = p$), and $x_i \oplus p$ (i.e., $x_i \neq p$), where p is a function defined over all variables except for a critical variable x_i (e.g., the variable with more switching activity or with higher delay that should be projected away from the rest of the circuit).

More precisely, let f be a completely specified Boolean function depending on the set $\{x_1, \ldots, x_n\}$ of n binary variables. The classical Shannon decomposition $f = \overline{x}_i f|_{\overline{x}_i} + x_i f|_{x_i}$, and the more general EXOR-based decomposition [8] [10]

$$f = (\overline{x}_i \oplus p) f|_{x_i=p} + (x_i \oplus p) f|_{x_i \neq p}$$

(as before, p is a function non-depending on x_i) are suitable for keeping x_i disjoint from the rest of the circuit, but are not oriented to area minimization. In fact, $f|_{x_i=p}$, and $f|_{x_i \neq p}$ do not depend on the variable x_i, but the cubes of f intersecting both subsets $x_i = p$ and $x_i \neq p$ may be split into two smaller subcubes when they are projected onto $f|_{x_i=p}$, and $f|_{x_i \neq p}$, respectively. To overcome this problem, in P-circuits synthesis, we keep unprojected some of the points of the original function. For this purpose, let $I = f|_{x_i=p} \cap f|_{x_i \neq p}$ be the intersection of the two cofactors $f|_{x_i=p}$ and $f|_{x_i \neq p}$. Note that the intersection I contains the cubes whose products do not contain x_i and that cross the two sets. Therefore, in order to overcome the splitting of these crossing cubes, we could keep I unprojected, and project only the minterms in $f|_{x_i=p} \setminus I$ and $f|_{x_i \neq p} \setminus I$, obtaining the expression

$$f = (\overline{x}_i \oplus p)(f|_{x_i=p} \setminus I) + (x_i \oplus p)(f|_{x_i \neq p} \setminus I) + I\,.$$

Note that p, $f|_{x_i=p} \setminus I$, $f|_{x_i \neq p} \setminus I$ and I do not depend on x_i. However, the points that are in I could be exploited to form bigger cubes in the projected sets. Therefore, if a point is in I and is useful for a better minimization of the projected parts, it can be kept both in the projection and in the intersection. Moreover, if a point is covered in both the projected sets, it is not necessary to cover it in the intersection. From these observations, we can infer that the projected subcircuits should cover at least $f|_{x_i=p} \setminus I$ and $f|_{x_i \neq p} \setminus I$, and must be contained in $f|_{x_i=p}$ and $f|_{x_i \neq p}$, respectively. Moreover, the part of the circuit that is not projected should be contained in the intersection I.

In summary, we can define a P-circuit as follows, where $S(f)$ indicates a SOP circuit implementing a Boolean function f.

Definition 1 ([7]): A *P-circuit* of a completely specified function f is the circuit $P(f)$ denoted by the expression:

$$P(f) = (\overline{x}_i \oplus S(p))\, S(f^{=}) + (x_i \oplus S(p))\, S(f^{\neq}) + S(f^I)$$

where

1) $(f|_{x_i=p} \setminus I) \subseteq f^{=} \subseteq f|_{x_i=p}$
2) $(f|_{x_i \neq p} \setminus I) \subseteq f^{\neq} \subseteq f|_{x_i \neq p}$
3) $\emptyset \subseteq f^I \subseteq I$
4) $P(f) = f\,.$

This definition can be easily generalized to incompletely specified Boolean functions $f = \{f^{on}, f^{dc}\}$. For the sake of simplicity, suppose that $f^{on} \cap f^{dc} = \emptyset$; otherwise, following the usual semantics, we consider $f^{on} \setminus f^{dc}$ as the on-set of f. Let I be the intersection of the projections of f onto the two sets $x_i = p$ and $x_i \neq p$:

$$I = (f^{on}|_{x_i=p} \cup f^{dc}|_{x_i=p}) \cap (f^{on}|_{x_i \neq p} \cup f^{dc}|_{x_i \neq p}).$$

Definition 2 ([7]): A *P-circuit* of an incompletely specified function $f = \{f^{on}, f^{dc}\}$ is the circuit $P(f)$ denoted by the expression:

$$P(f) = (\overline{x}_i \oplus S(p))\, S(f^{=}) + (x_i \oplus S(p))\, S(f^{\neq}) + S(f^I)$$

where

1) $(f^{on}|_{x_i=p} \setminus I) \subseteq f^{=} \subseteq f^{on}|_{x_i=p} \cup f^{dc}|_{x_i=p}$
2) $(f^{on}|_{x_i \neq p} \setminus I) \subseteq f^{\neq} \subseteq f^{on}|_{x_i \neq p} \cup f^{dc}|_{x_i \neq p}$
3) $\emptyset \subseteq f^I \subseteq I$
4) $f^{on} \subseteq P(f) \subseteq f^{on} \cup f^{dc}$.

An example of P-circuit decomposition is presented in Section III-A.

As we have seen, the idea for synthesis of P-circuits is to construct a network for f by appropriately choosing the sets $f^{=}$, $f^{\neq}$, and f^I as building blocks. Several algorithms for performing such a choice have been studied and proposed in the literature [4], [5], [6], [7]. In particular, in [7], it is shown how the structural flexibility of P-circuits can be completely characterized by using Boolean relations, and how the associated optimal P-circuit decomposition problem can be efficiently solved using a Boolean relation minimizer.

III. Decomposition with lattices

In this section we will discuss how the decomposition of Boolean functions can be exploited in the synthesis of switching lattices. The basic idea of this approach is to first decompose a function into some subfunctions, according to a given functional decomposition scheme, and then to implement the decomposed blocks with separate lattices, or physically separated regions in a single lattice. Since the decomposed blocks usually correspond to functions depending on fewer variables and/or with a smaller on-set, their synthesis should be

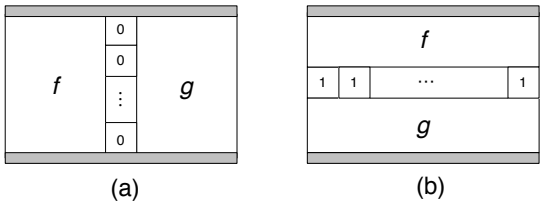

Fig. 2. Lattice implementation of $f + g$ (a) and of $f \cdot g$ (b).

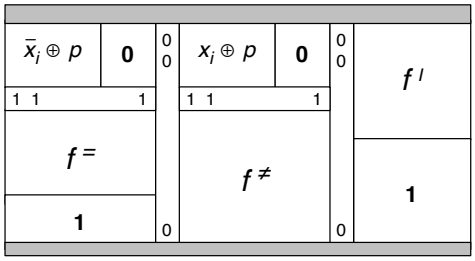

Fig. 3. Lattice implementation of a P-circuit.

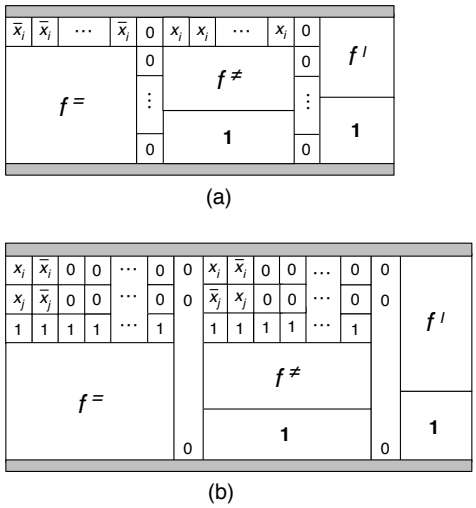

Fig. 4. Lattice implementation of a P-circuit with projection function $p = 0$ (a) and with projection function $p = x_j$ (b).

easier and should produce lattice implementations of smaller size.

Decomposition of logic functions is a widely studied field in multi-level logic, see [11] for a review on this subject. Here we focus on the decomposition method that gives rise to the bounded-level logic networks called P-circuits, described in Section II-B.

First of all, we recall from [9] that given the switching lattices implementing two functions f and g, we can easily construct the lattices representing their disjunction $f + g$ and their conjunction $f \cdot g$ using a padding column of 0s and a padding row of 1s, respectively, as shown in Figure 2. Indeed, the column of 0s separates all top-to-bottom paths in the lattices for f and g, so that the accepting paths for the two functions never intersect. This, in turn, implies that there exists a top-to-bottom connected path in the lattice for $f + g$ if and only if there is at least one connected path for f or for g. Of course, if the lattices for f and g have a different number of rows, we add some rows of 1s to the lattice with fewer rows, so that each accepting path can reach the bottom edge.

Similarly, the padding row of 1s allows to join any top-to-bottom accepting path for the function f with any top-to-bottom accepting path for g, so that the overall lattice evaluates to 1 if and only if both f and g evaluate to 1. As before, if the lattices for f and g have a different number of columns, we add some columns of 0s to the lattice with fewer columns, so that an accepting path for one of the two functions can never reach the opposite edge of the lattice if the other function evaluates to 0.

We can use these simple composition rules to derive a lattice describing a P-circuit expression $P(f) = (\overline{x}_i \oplus S(p))\, S(f^=) + (x_i \oplus S(p))\, S(f^{\neq}) + S(f^I)$ for a given function f, using lattices for the three sets $f^=$, $f^{\neq}$, and f^I and for the projection functions $(\overline{x}_i \oplus p)$ and $(x_i \oplus p)$ as building blocks, as depicted in Figure 3.

We now formally prove that the lattice implementation of a P-circuit $P(f)$ described in Figure 3 correctly implements the function f.

Theorem 1: Let f be a Boolean function depending on n binary variables, and let $P(f) = (\overline{x}_i \oplus S(p))\, S(f^=) + (x_i \oplus S(p))\, S(f^{\neq}) + S(f^I)$ be a P-circuit representing f. The lattice obtained composing the lattices for the three sets $f^=$, $f^{\neq}$, and f^I and for the projection functions $(\overline{x}_i \oplus p)$ and $(x_i \oplus p)$, as shown in Figure 3, implements the function f.

Proof: In order to prove the theorem, we need to show that the function f evaluates to 1 on a given input assignment if and only if there exists a connected path between the top and the bottom edge of the lattice in Figure 3.

First suppose that f evaluates to 1 on a given input $(x_1, x_2, \ldots, x_n)$. Then, at least one of the three terms in the expression for $P(f)$ must evaluate to 1. Suppose that the first term $(\overline{x}_i \oplus S(p))\, S(f^=)$ takes the value 1 on $(x_1, x_2, \ldots, x_n)$. Then, both $(\overline{x}_i \oplus S(p))$ and $S(f^=)$ are equal to 1, giving rise to a top-to-bottom connected path in the left side of the lattice. An analogous situation arises if the second or the third term are equal to 1.

Now suppose that a given input assignment $(x_1, x_2, \ldots, x_n)$ induces some connected top-to-bottom paths on the lattice. Due to the presence of the two columns of 0s, separating the left, center, and right portions of the lattice corresponding to the implementations of the functions $(\overline{x}_i \oplus p)\, f^=$, $(x_i \oplus p)\, f^{\neq}$, and f^I, respectively, each connected path is entirely contained in one of the three portions. This implies that at least one of the three terms in input to the final OR gate of the P-circuit representing f is equal to 1, and the thesis immediately follows. ■

The lattice description of the P-circuit can be simplified depending on the chosen projection function. For instance, if we choose the P-circuit $P(f) = \overline{x}_i\, S(f^=) + x_i\, S(f^{\neq}) + S(f^I)$ based on the generalization of the classical Shannon decomposition with projection function $p = 0$, which experimentally represents a very efficient and effective solution, we get the lattice shown in Figure 4 (a), where the two padding rows of

1s have been substituted with one row of cells assigned to the literal $\overline{x}_i$ and one row of cells assigned to x_i. Figure 4 (b) shows the lattice implementation of the P-circuit $P(f) = (\overline{x}_i \oplus x_j)\, S(f^{=}) + (x_i \oplus x_j)\, S(f^{\neq}) + S(f^I)$ corresponding to the choice $p = x_j$. Both lattices correctly implement the function f, as proved in the following corollary.

Corollary 1: The two lattices in Figure 4 implement the function f through its P-circuit representations with projection functions $p = 0$ and $p = x_j$, respectively.

Proof: Let us consider the first lattice, corresponding to the P-circuit representation of f with projection function $p = 0$. As before, observe that each top-to-bottom connected path must be entirely contained in one of the three portions of the lattice, because of the two padding columns of 0s. Moreover, the row of cells assigned to $\overline{x}_i$ on top of the lattice for $f^{=}$ allows to derive a top-to-bottom connected path for f from a connected path for $f^{=}$ if and only if $x_i = 0$; analogously, the row of cell assigned to x_i on top of the lattice for $f^{\neq}$ allows to derive a top-to-bottom connected path for f from a connected path for $f^{\neq}$ if and only if $x_i = 1$. Finally, any connected path for f^I can be extended to a connected path for f. Thus the thesis immediately follows from the definition of P-circuit and of the terms $f^{=}$, $f^{/=}$, and f^I (see Definitions 1 and 2).

Now consider the lattice in Figure 4 (b), corresponding to the P-circuit for f with projection function $p = x_j$. Any connected path for f^I can be extended to a connected path for f. Moreover, any connected path for $f^{=}$ can be extended to a connected path for f if and only if both variables x_i and x_j assume the same value, i.e., $x_i = x_j$, while any connected path for $f^{\neq}$ can be extended to a connected path for f if and only if the two variables assume different values, i.e., $x_i = \overline{x}_j$. Since the presence of the two columns of 0s prevents the existence of top-to-bottom connected paths intersecting different regions of the lattice, the thesis immediately follows from Definitions 1 and 2. ■

As already observed, the main idea behind this approach is that lattice synthesis of the subfunctions $f^{=}$, $f^{\neq}$, and f^I, which depend on $n-1$ variables instead of n and have a smaller on-set than f, should be an easier task, and should produce lattices of reduced size, so that the overall lattice for f - derived using minimal lattices for $f^{=}$, $f^{\neq}$, and f^I as building blocks - could be smaller than the one derived for f without exploiting its decomposition in P-circuits. This expectation has been confirmed by our experimental results (Section IV).

A. An example of lattice decomposition

In order to better describe the synthesis of lattices based on the P-circuit decomposition scheme, we describe here a simple example. Consider the benchmark $z4$ taken from LGSynth93 [12], and in particular its third output $z4(2)$, simply denoted by z. This function, represented by the Karnaugh map in Figure 5, depends on seven variables $(x_1, x_2, \ldots, x_7)$, two of which (x_2 and x_5) are nonessential. Consider the P-circuit decomposition with respect to the first variable x_1 and to the projection function $p = 0$. The three sets $z^{=}$, $z^{\neq}$, and z^I that define an optimal P-circuit representation of $z = z4(2)$ are depicted in Figure 6. Observe that the two minterms 0010 and 1000 in the intersection between the cofactors $z|_{x_1=0}$ and $z|_{x_1\neq 0}$ have been removed from z^I since they have been kept

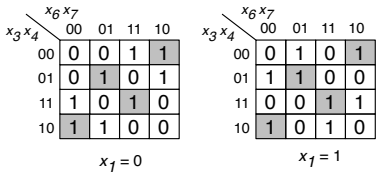

Fig. 5. Karnaugh map of the benchmark $z = z4(2)$. The cells in grey show the minterms that belong to the intersection between the cofactors $z|_{x_1=0}$ and $z|_{x_1\neq 0}$.

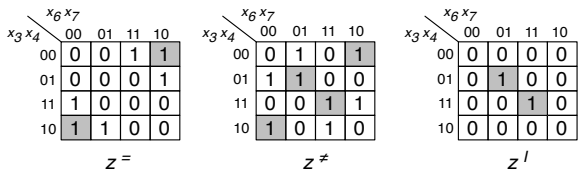

Fig. 6. Karnaugh maps of the sets $z^{=}$, $z^{\neq}$, and z^I defining an optimal P-circuit for $z = z4(2)$ with respect to x_1 and to the function $p = 0$. The minterms in the intersection between $z|_{x_1=0}$ and $z|_{x_1\neq 0}$, now distributed among $z^{=}$, $z^{\neq}$, and z^I, are highlighted in grey.

in both sets $z^{=}$ and $z^{\neq}$ in order to get smaller SOP forms. Moreover, the other two points of the intersection, 0101 and 1111, are kept both in z^I and in the projection $z^{\neq}$, where they are used to form bigger cubes. The P-circuit representation of $z = z4(2)$ is then given by the expression

$$P(z) = \overline{x}_1\, S(z^{=}) + x_1\, S(z^{\neq}) + S(z^I)$$

where $S(z^{=}) = \overline{x}_3\overline{x}_4x_6 + x_3\overline{x}_4\overline{x}_6 + \overline{x}_3x_6\overline{x}_7 + x_3\overline{x}_6\overline{x}_7$, $S(z^{\neq}) = x_3x_4x_6 + \overline{x}_3x_4\overline{x}_6 + x_3x_6x_7 + \overline{x}_3\overline{x}_6x_7 + \overline{x}_3\overline{x}_4x_6\overline{x}_7 + x_3\overline{x}_4\overline{x}_6\overline{x}_7$, and $S(z^I) = x_3x_4x_6x_7 + \overline{x}_3x_4\overline{x}_6x_7$ are the SOP representations of $z^{=}$, $z^{\neq}$, and z^I, respectively.

We can now derive a lattice implementation of z using lattices for the three subfunctions $z^{=}$, $z^{\neq}$, and z^I as building blocks, as depicted in Figure 4 (a). Computing the sublattices for $z^{=}$, $z^{\neq}$, and z^I using the method described in [3], we get an overall lattice of dimension 7×14, as shown in Figure 7. Note that the synthesis method in [3] applied directly to z, without exploiting its P-circuit decomposition, would produce a lattice of dimension 12×12.

$\overline{x}_1$	$\overline{x}_1$	$\overline{x}_1$	$\overline{x}_1$	0	x_1	x_1	x_1	x_1	x_1	x_1	0	x_4	x_4
x_6	x_3	x_6	x_3	0	x_6	$\overline{x}_3$	x_6	$\overline{x}_3$	$\overline{x}_3$	$\overline{x}_4$	0	x_7	x_7
$\overline{x}_3$	$\overline{x}_6$	$\overline{x}_3$	$\overline{x}_6$	0	x_3	$\overline{x}_6$	x_3	$\overline{x}_6$	$\overline{x}_4$	$\overline{x}_4$	0	x_6	$\overline{x}_3$
$\overline{x}_4$	$\overline{x}_4$	$\overline{x}_7$	$\overline{x}_7$	0	x_6	$\overline{x}_3$	x_6	$\overline{x}_3$	$\overline{x}_3$	$\overline{x}_7$	0	x_3	$\overline{x}_6$
1	1	1	1	0	x_3	$\overline{x}_6$	x_3	$\overline{x}_6$	$\overline{x}_7$	$\overline{x}_7$	0	1	1
1	1	1	1	0	x_3	x_4	x_3	x_7	x_6	x_3	0	1	1
1	1	1	1	0	x_4	x_4	x_7	x_7	$\overline{x}_3$	$\overline{x}_6$	0	1	1

Fig. 7. Lattice for the benchmark $z = z4(2)$ based on the P-circuit decomposition. The sublattices for $z^{=}$, $z^{\neq}$, and z^I are highlighted in grey.

IV. Experimental results

In this section we report the experimental results obtained by applying the decomposition with lattices described in Section III. The algorithms have been implemented in C. The experiments have been run on a machine with two AMD Opteron 4274HE for a total of 16 CPUs at 2.5 GHz and 128 GByte of main memory, running Linux CentOS 6.6. The benchmarks are taken from LGSynth93 [12]. We considered each output as a separate Boolean function, for a total of 1886 functions. Due to the limited space available, we report in the following only a significant subset of the functions as representative indicators of our experiments. Since the experimental results in [4], [5] show that the best choice for the projection function p is often $p = 0$, we evaluate and report the results just for $p = 0$, using $x_i = x_1$, i.e. we decompose with respect to the first input variable of each benchmark.

In order to evaluate the utility of our approach, we compare our results with the ones obtained by the methods presented in [3] and in [9]. To simulate the results reported in [9], we used a collection of Python scripts for computing minimum-area switching lattices, using transformation to a series of SAT problems.

To improve the readability, we divided in alphabetical order the reported benchmarks in two tables, Table I and Table II. The first column of each table reports the name and the number of the considered output of each instance. The following columns report, by groups of four columns, the lattice dimensions (X, Y), the area ($Area = X \cdot Y$) and the time (in seconds) to synthesize the lattice of each considered method. In particular, the first two groups refer to the synthesis of the lattice, as described in [3], without any decomposition (columns 2, 3, 4 and 5), and with the decomposition proposed in this paper (columns 6, 7, 8 and 9); the last two groups refer to the synthesis of the lattice, as described in [9], without any decomposition (columns 10, 11, 12 and 13), and with the decomposition proposed in this paper (columns 14, 15, 16 and 17). The time for the synthesis of the lattice decomposed as described in this paper includes the time for the decomposition plus the time for the synthesis of lattices used to compose the final lattice. Each row of a table lists the numbers for a separate output function of the benchmark circuit. The last row of a table reports the sum of the values of the corresponding column. For each function, we bolded the best area. By comparing the results of the first two groups, the values show that we obtain a smaller area in about 36% of cases, with an average gain of about 25%, at the expense of a very limited increase in simulation time needed to decompose the input function.

The values of the last two groups show that we obtain a smaller area in about 33% of cases, with an average gain of about 24%. The instances not reported in Table I and Table II follow the same distributions.

For the method [9], the simulation times exhibit significant variations. In many cases we obtain a smaller area, at the expense of a much increased simulation time (for example, addm4(0)), as reasonably expected. But in other cases we do better with the same time (addm4(3)), or even we do better or the same with much less time (addm4(2)). In the overall we save area and time.

In [9], in all cases where the result of the proposed method is worse than the result of the initial naive construction, the authors report the result computed with the method proposed in [3]. In Table I and Table II we adopt the same choice. So there are some functions (for example addm4(3)) where the reported results (for both area and simulation time) are the same as the ones computed with the method proposed in [3]. Moreover, in many cases the method proposed in [9] fails in computing a result in reasonable simulation time. For this reason, we set a time limit (equal to ten minutes) for each SAT execution; if we do not find a solution within the time limit, the synthesis is stopped and, as before, we report the result computed with the method proposed in [3].

V. Conclusions

In this paper we proposed a new method for the synthesis of minimal-sized lattices, preprocessing the logic functions with a decomposition based on P-circuits. The results demonstrate that lattice synthesis benefits from this type of Boolean decomposition, yielding smaller circuits with an affordable computation time (even less in some cases). Future work includes assessing the impact of more complex types of decompositions, both within the class of P-circuits (with more expressive projection functions) and beyond.

VI. Acknowledgments

This project has received funding from the European Union Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 691178.

References

[1] S. B. Akers, "A Rectangular Logic Array," *IEEE Trans. Comput.*, vol. 21, no. 8, pp. 848–857, Aug. 1972.

[2] M. Altun and M. D. Riedel, "Lattice-Based Computation of Boolean Functions," in *Proceedings of the 47th Design Automation Conference, DAC 2010, Anaheim, California, USA, July 13-18, 2010*, 2010, pp. 609–612.

[3] ——, "Logic Synthesis for Switching Lattices," *IEEE Trans. Computers*, vol. 61, no. 11, pp. 1588–1600, 2012.

[4] A. Bernasconi, V. Ciriani, G. Trucco, and T. Villa, "On Decomposing Boolean Functions via Extended Cofactoring," in *Design Automation and Test in Europe (DATE)*, 2009, pp. 1464–1469.

[5] ——, "Logic Synthesis by Signal-Driven Decomposition," in *Advanced Techniques in Logic Synthesis, Optimizations and Applications*, K. Gulati, Ed. Springer New York, 2011, pp. 9–29.

[6] A. Bernasconi, V. Ciriani, V. Liberali, G. Trucco, and T. Villa, "Synthesis of P-Circuits for Logic Restructuring," *Integration*, vol. 45, no. 3, pp. 282–293, 2012.

[7] A. Bernasconi, V. Ciriani, G. Trucco, and T. Villa, "Using Flexibility in P-Circuits by Boolean Relations," *IEEE Trans. Computers*, vol. 64, no. 12, pp. 3605–3618, 2015.

[8] J. C. Bioch, "The Complexity of Modular Decomposition of Boolean Functions," *Discrete Applied Mathematics*, vol. 149, no. 1-3, pp. 1–13, 2005.

[9] G. Gange, H. Søndergaard, and P. J. Stuckey, "Synthesizing Optimal Switching Lattices," *ACM Trans. Design Autom. Electr. Syst.*, vol. 20, no. 1, pp. 6:1–6:14, 2014.

[10] V. Kravets, "Constructive Multi-Level Synthesis by Way of Functional Properties," Ph.D. dissertation, Computer Science Engineering, University of Michigan, 2001.

[11] T. Sasao, *Switching Theory for Logic Synthesis*. Kluwer Academic Publishers, 1999.

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

TABLE I. PROPOSED LATTICE SIZES FOR STANDARD BENCHMARK CIRCUITS: A COMPARISON OF THE PROPOSED METHOD WITH THE RESULTS PRESENTED IN [3] AND IN [9]. RESULTS ARE MARKED WITH $^\circ$ WHEN THE METHOD IN [3] GIVES BETTER RESULTS AND WITH * WHEN SAT IS STOPPED AFTER 10 MINUTES.

	[3] vs proposed method								[9] vs proposed method							
	Standard solution				Decomposition-based solution				Standard solution				Decomposition-based solution			
Name	X	Y	Area	Time(s)	X	Y	Area	Time(s)	X	Y	Area	Time(s)	X	Y	Area	Time(s)
addm4(0)	9	12	108	0	9	12	108	0	9	12	108°	0,35	4	6	**24**	1969,37
addm4(1)	22	23	**506**	0	22	23	**506**	0	22	23	**506***	89,24	22	23	**506**$^\circ$	0,07
addm4(2)	33	36	1188	0	32	36	**1152**	0	33	36	1188*	89,55	32	36	**1152***	38,3
addm4(3)	49	55	2695	0	52	39	**2028**	0	49	55	2695°	0,05	52	39	**2028**$^\circ$	0,09
addm4(4)	62	63	3906	0	67	37	**2479**	0	62	63	3906*	206,78	67	37	**2479***	56,07
addm4(5)	25	26	650	0	26	17	442	0	25	26	650*	881,27	24	17	**408**	8549,6
addm4(6)	10	11	110	0	11	7	77	0	10	11	110*	12,41	10	4	**40**	3,23
adr4(1)	36	36	1296	0	37	19	**703**	0,01	36	36	1296°	0,04	37	19	**703***	2192,28
alu2(2)	8	7	56	0	11	5	55	0,05	3	5	**15**	13,44	8	5	40	0,94
alu2(5)	5	5	25	0	9	5	45	0,05	3	5	**15**	48,6	8	5	40	0,11
alu2(6)	14	13	182	0	17	10	170	0,05	14	13	182°	0,02	16	10	**160**	17,57
alu3(0)	5	4	20	0	8	4	32	0,01	3	3	**9**	0,47	7	4	28	0,07
alu3(1)	8	7	56	0	11	5	55	0,01	3	5	**15**	12,75	8	5	40	0,89
b12(0)	4	6	24	0	4	6	24	0,41	4	3	**12**	2,46	3	4	**12**	1,24
b12(1)	7	5	35	0	7	5	35	0,41	4	4	**16**	145,44	4	4	**16**	3,44
b12(2)	7	6	42	0	8	6	48	0,41	4	4	**16**	376,27	8	5	40	5,65
b12(3)	4	2	8	0	5	2	10	0,41	3	2	**6**	0,15	5	2	10°	0,47
b12(6)	9	6	54	0	10	6	60	0,41	5	4	**20**	1388,99	10	5	50	14,7
b12(7)	6	4	24	0	10	4	40	0,41	3	6	**18**	10,59	9	3	27	2,3
b12(8)	7	2	**14**	0	11	2	22	0,41	7	2	**14**$^\circ$	0,02	10	2	20	1,25
bcc(35)	24	38	912	0,02	25	24	**600**	0,03	24	38	912°	0,08	25	24	**600***	0,11
bcc(36)	4	17	**68**	0	5	15	75	0,03	4	17	**68**$^\circ$	0,02	5	15	75°	0,09
bcc(37)	11	34	374	0,02	12	22	**264**	0,03	11	34	374°	0,04	12	22	**264***	684,8
bcc(38)	3	16	**48**	0	4	15	60	0,03	3	16	**48**$^\circ$	0,02	4	15	60°	0,09
bcc(39)	2	15	**30**	0	2	15	**30**	0,03	2	15	**30**$^\circ$	0,02	2	15	**30**$^\circ$	0,09
bcc(43)	10	20	200	0	11	16	**176**	0,03	10	20	200°	0,03	11	16	**176***	726,05
bcc(44)	6	20	**120**	0	6	20	**120**	0,03	6	20	**120**$^\circ$	0,02	6	20	**120***	537,09
bcd,div3(0)	2	2	4	0	1	3	**3**	0,03	2	2	4°	0,02	1	3	**3**$^\circ$	0,06
bcd,div3(1)	3	4	12	0	6	4	24	0,03	3	3	**9**	0,31	5	4	20	0,09
bcd,div3(2)	3	4	12	0	6	4	24	0,03	3	3	**9**	0,3	5	4	20	0,09
bcd,div3(3)	3	5	15	0	4	4	16	0,03	4	3	**12**	0,55	4	4	16°	0,09
bench1(6)	21	35	735	0	26	24	624	3,72	21	35	735*	56,24	26	24	**624***	156,54
bench1(7)	21	43	903	0	28	20	560	3,72	21	43	903*	56,91	27	20	**540**	86,61
bench1(8)	24	44	1056	0	31	26	806	3,72	24	44	1056*	1210,65	31	26	**806***	3,8
bench(2)	6	7	42	0	7	5	35	0,34	4	4	**16**	85,08	6	4	24	1,29
bench(3)	4	6	24	0	7	4	28	0,34	4	3	**12**	1,7	6	4	24	0,4
bench(4)	3	5	15	0	4	5	20	0,34	4	3	**12**	0,85	4	5	20°	0,4
bench(5)	2	3	**6**	0	3	3	9	0,34	2	3	**6**$^\circ$	0,02	3	3	9	0,4
bench(6)	4	8	32	0	7	3	21	0,34	3	4	**12**	1,85	6	3	18	0,4
bench(7)	4	6	24	0	5	4	20	0,34	5	3	**15**	13,94	5	4	20°	0,4
br2(0)	5	14	**70**	0	5	14	**70**	0,01	5	14	**70**$^\circ$	0,02	5	14	**70**$^\circ$	0,07
br2(1)	3	11	33	0	3	11	33	0,01	3	11	33*	118,7	2	8	**16**	2411,03
br2(2)	2	12	**24**	0	2	12	**24**	0,01	2	12	**24***	52,79	2	12	**24**$^\circ$	0,07
br2(4)	8	18	**144**	0	8	18	**144**	0,01	8	18	**144***	140,6	8	18	**144**$^\circ$	0,07
br2(5)	4	14	**56**	0	4	14	**56**	0,01	4	14	**56***	68,53	4	14	**56***	122,7
br2(6)	5	16	**80**	0	5	16	**80**	0,01	5	16	**80***	171,65	5	16	**80***	41,37
br2(7)	4	12	**48**	0	4	12	**48**	0,01	4	12	**48***	858,6	4	12	**48***	720,41
clpl(0)	4	4	16	0	5	4	20	0,52	3	4	**12**	1,79	5	4	20°	0,58
clpl(1)	3	3	**9**	0	7	3	21	0,52	3	3	**9**$^\circ$	0,02	6	3	18	0,58
clpl(2)	2	2	**4**	0	3	2	6	0,52	2	2	**4**$^\circ$	0,02	3	2	6°	0,58
clpl(3)	6	6	**36**	0	10	6	60	0,52	6	6	**36***	2301,93	9	6	54	15,25
clpl(4)	5	5	25	0	9	5	45	0,52	3	5	**15**	53,51	8	5	40	0,58
co14(0)	14	92	1288	0	15	80	1200	1,03	14	14	**196**	0,21	15	80	1200°	1,21
			17464	0,04			13413	20,28			16077	8475,91			13068	18371,03

TABLE II. PROPOSED LATTICE SIZES FOR STANDARD BENCHMARK CIRCUITS: A COMPARISON OF THE PROPOSED METHOD WITH THE RESULTS PRESENTED IN [3] AND IN [9]. RESULTS ARE MARKED WITH $^{\circ}$ WHEN THE METHOD IN [3] GIVES BETTER RESULTS AND WITH * WHEN SAT IS STOPPED AFTER 10 MINUTES.

	[3] vs proposed method								[9] vs proposed method							
	Standard solution				Decomposition-based solution				Standard solution				Decomposition-based solution			
Name	X	Y	Area	Time(s)	X	Y	Area	Time(s)	X	Y	Area	Time(s)	X	Y	Area	Time(s)
dc1(0)	4	4	16	0	5	4	20	0,01	3	3	**9**	0,38	4	4	16	0,24
dc1(1)	2	3	**6**	0	3	3	9	0,01	2	3	**6**$^{\circ}$	0,02	3	3	9°	0,07
dc1(4)	4	5	20	0	5	4	20	0,01	4	3	**12**	0,58	5	4	20°	0,07
dc1(6)	3	3	9	0	4	2	8	0,01	2	3	**6**	0,19	4	2	8°	0,07
dc2(4)	9	10	90	0	10	9	90	0	5	4	**20**	1241,2	8	5	40	11,55
dc2(5)	6	6	36	0	7	7	49	0	6	2	**12**	34,83	5	6	30	1,91
dk17(0)	2	8	16	0	4	4	16	0,04	6	2	**12**	59,54	4	4	16°	0,1
dk17(1)	2	8	16	0	4	4	16	0,04	6	2	**12**	78,71	4	4	16°	0,1
dk17(3)	3	11	33	0	4	7	28	0,04	7	2	**14**	1233,19	6	3	18	124,35
dk17(4)	3	9	27	0	7	4	28	0,04	6	3	**18**	980,36	6	4	24	0,1
dk17(8)	3	8	24	0	8	4	32	0,04	6	2	**12**	429,53	8	4	32	3,28
dk17(9)	4	5	20	0	7	4	28	0,04	3	4	**12**	1,75	6	4	24	0,1
dk27(6)	1	2	**2**	0	1	2	**2**	0	1	2	**2**$^{\circ}$	0,03	1	2	**2**$^{\circ}$	0,06
ex4(4)	6	17	**102**	0	6	17	**102**	0	6	17	**102***	3656,45	6	17	**102**$^{\circ}$	0,06
ex4(5)	45	35	**1575**	0,01	45	35	**1575**	0	45	35	**1575**$^{\circ}$	0,09	45	35	**1575***	0,14
ex5(31)	8	4	32	0	11	4	44	0,11	3	6	**18**	12,56	10	3	30	1,88
ex5(32)	10	4	40	0	14	3	42	0,11	4	6	**24**	2369,44	13	3	39	1,12
ex5(33)	7	3	**21**	0	7	3	**21**	0,11	7	3	**21***	201,48	7	3	**21***	1,98
ex5(36)	8	2	**16**	0	11	2	22	0,11	8	2	**16**$^{\circ}$	0,02	10	2	20	2,46
ex5(38)	9	4	36	0	14	3	42	0,11	4	6	**24**	940,79	13	3	39	5,88
ex5(39)	8	2	**16**	0	12	3	36	0,11	8	2	**16***	23,13	11	3	33	0,76
ex5(40)	12	6	72	0	16	5	80	0,11	12	6	72*	517,8	13	4	**52**	1,87
ex5(43)	14	8	112	0	17	6	102	0,11	14	8	112*	19,67	13	4	**52**	533,23
ex5(44)	7	2	**14**	0	10	2	20	0,11	7	2	**14**$^{\circ}$	0,02	9	2	18	0,31
ex5(46)	6	3	**18**	0	6	3	**18**	0,11	6	3	**18***	13,59	6	3	**18***	1,02
ex5(49)	6	2	**12**	0	6	3	18	0,11	6	2	**12**$^{\circ}$	0,02	6	3	18°	0,47
ex5(51)	10	7	70	0	13	5	65	0,11	10	7	70*	3784,24	11	3	**33**	216,87
ex5(52)	6	6	36	0	9	6	54	0,11	3	6	**18**	12,09	8	3	24	19,92
ex5(53)	12	10	120	0	15	6	90	0,11	12	10	120*	5,99	11	4	**44**	317,66
ex5(54)	14	8	112	0	18	8	144	0,11	14	8	112*	941,76	14	5	**70**	684,6
ex5(56)	8	5	40	0	7	5	35	0,11	3	7	21	219,67	6	3	**18**	25,9
ex5(57)	10	8	80	0	14	8	112	0,11	3	7	**21**	1502,89	13	3	39	278,33
ex5(60)	9	3	27	0	12	4	48	0,11	4	6	**24**	583,27	11	3	33	16,28
ex5(62)	5	2	**10**	0	5	2	**10**	0,11	5	2	**10**$^{\circ}$	0,02	5	2	**10***	0,15
exam(5)	6	11	66	0	7	6	42	4,62	6	11	66*	1008,68	6	5	**30**	93,77
exam(9)	30	59	1770	0	38	30	**1140**	4,62	30	59	1770°	0,04	38	30	**1140***	2298,26
max128(7)	6	5	30	0	8	5	40	0,77	3	6	**18**	3,5	8	5	40°	0,56
max128(8)	10	5	50	0	11	4	44	0,77	10	5	50*	39,76	10	4	**40**	2,34
max128(9)	10	9	90	0	14	7	98	0,77	4	6	**24**	2637,43	13	4	52	3,93
max128(10)	16	17	272	0	17	10	170	0,77	16	17	272*	2,5	15	10	**150**	2814,41
max128(11)	23	23	529	0	25	15	375	0,77	23	23	529*	168,22	20	15	**300**	810,01
mp2d(1)	8	6	48	0	12	6	72	0,1	8	6	48*	2454,43	11	4	**44**	0,42
mp2d(2)	10	5	**50**	0	14	5	70	0,1	10	5	**50**$^{\circ}$	0,01	13	4	52	0,22
mp2d(6)	6	10	**60**	0	6	10	**60**	0,1	6	10	**60**$^{\circ}$	0,01	6	10	**60***	2635,89
mp2d(9)	8	6	48	0	9	6	54	0,1	8	6	48*	2484,13	9	4	**36**	0,23
mp2d(10)	3	6	18	0	4	5	20	0,1	4	3	**12**	0,58	4	5	20°	0,17
z4(0)	15	15	225	0	16	11	176	0	5	4	**20**	1654,72	6	6	36	3,11
z4(1)	28	28	784	0	31	16	496	0	28	28	784*	4,34	18	16	**288**	1751,69
z4(2)	12	12	144	0	15	7	105	0	12	12	144*	579,29	11	5	**55**	1,34
z4(3)	4	4	16	0	5	3	15	0	3	3	**9**	0,11	5	3	15°	0,02
Z5xp1(0)	3	5	15	0	3	5	15	0	4	3	**12**	1,02	3	5	15°	1,19
Z5xp1(1)	7	7	49	0	8	5	40	0	4	5	**20**	2,02	8	4	32	84,6
Z5xp1(2)	11	12	132	0	14	7	98	0	11	12	132*	371	11	5	**55**	2447
Z5xp1(3)	18	18	324	0	19	11	209	0	18	18	324*	4,02	10	6	**60**$^{\circ}$	0
			7596	0,01			6365	15,96			6959	30281,11			5063	15202,05