XOR-AND-XOR Logic Forms for Autosymmetric Functions and Applications to Quantum Computing

Anna Bernasconi, Alessandro Berti, Valentina Ciriani, *Senior Member, IEEE*, Gianna M. Del Corso, and Innocenzo Fulginiti

***Abstract*—We propose a new three-level XOR-AND-XOR form for autosymmetric functions, called XORAX expression. In general, a Boolean function f over n variables is k-autosymmetric if it can be projected onto a smaller function f_k, which depends on $n-k$ variables only. We show that XORAX expressions can ease the reversible synthesis of autosymmetric functions, producing compact reversible networks, without inserting additional new input lines. Autosymmetry occurs especially for functions that exhibit a regular structure, as for instance arithmetic functions. For this reason, compact reversible networks for autosymmetric functions might be interesting for quantum computing. Experimental results validate the proposed approach.**

***Index Terms*—Autosymmetric function, quantum circuits, reversible logic.**

I. Introduction

AUTOSYMMETRIC Boolean functions are functions that exhibit a structural regularity based on the notion of affine spaces and are easily expressed using XORs. They were introduced in [20] and further studied in [4], [8], [9], and [18], where it was shown how this regularity can be exploited to derive, in shorter synthesis time, compact logic representations for the CMOS technology. More recently, the autosymmetry property has been applied in logic synthesis for emerging technologies, i.e., switching nano-crossbars [7], and to better estimate the multiplicative complexity of functions for security protocols [5], [41].

The aim of this article is to establish whether the concept of autosymmetry can ease the synthesis of Boolean functions also in the context of reversible and quantum circuits. Indeed, since autosymmetry often occurs for functions presenting a regular structure, as for instance arithmetic functions, compact reversible networks for autosymmetric functions might be interesting to ease the quantum circuit synthesis of subroutines given as classical functions, an issue occurring in many contexts (e.g., the Shor's algorithm for factoring and discrete logarithm and the Grover's search algorithm).

For this reason, we first propose a new three-level XOR-AND-XOR form, called XORAX, for autosymmetric functions, and then, we analyze theoretically and experimentally the applications of this new logic form for the synthesis of reversible and quantum circuits. Note that an XORAX form can be described as a generalization of exclusive-or sum-of-products (ESOPs) expressions, that is an XOR of ANDs of literals. Indeed, an XORAX is an XOR of ANDs of XORs of literals.

Intuitively, a Boolean function f over n variables is *k-autosymmetric* if it can be projected onto a smaller function f_k that depends on $n-k$ variables and contains a reduced number of minterms. The regularity of a Boolean function f is then measured computing its *autosymmetry degree* k, with $0 \leq k \leq n$, where $k = 0$ means no regularity. In this article, we prove that the reduction from f to f_k can be efficiently implemented onto reversible and quantum circuits using only a limited number of controlled NOT (CNOT) gates and without adding additional input lines/qubits to the networks. Moreover, since f_k is smaller than the original function f, both in terms of input variables and number of on-set minterms, its reversible synthesis should be easier and should lead to more compact quantum circuits.

Observe that in an exact synthesis scenario the autosymmetry preprocessing would not be necessary. However, in more realistic scenarios, where heuristic procedures are applied, the autosimmetry property can actually ease the synthesis process by simplifying the input instances. Indeed, the experimental results of this article show that the heuristic procedures are not in general sensitive to autosymmetry, and therefore, a preprocessing exploiting this regularity, often guarantees better synthesis results.

Autosymmetric functions are just a subset of all Boolean functions. Nevertheless, a considerable amount of standard functions of practical interest falls in this class. For instance, about 24% of the functions in the classical ESPRESSO benchmark suite [44] has at least one non degenerate autosymmetric output [8], [9], [10]. Notice that on the other hand, the total number of classical symmetric functions, i.e., the Boolean functions invariant under any permutation of their variables is much lower [8].

We have experimentally evaluated the approach proposed for the reversible synthesis of autosymmetric functions starting from their XORAX expressions, with interesting results.

Manuscript received 17 May 2022; revised 10 August 2022; accepted 27 September 2022. Date of publication 10 October 2022; date of current version 19 May 2023. This work was supported by the Università di Pisa through the "PRA Progetti di Ricerca di Ateneo" (Institutional Research Grants) under Project PRA 2020-2021 92 (Quantum Computing, Technologies and Applications). This article was recommended by Associate Editor M. Soeken. *(Corresponding author: Valentina Ciriani.)*

Anna Bernasconi, Alessandro Berti, Gianna M. Del Corso, and Innocenzo Fulginiti are with the Department of Computer Science, Università di Pisa, 56126 Pisa, Italy (e-mail: anna.bernasconi@unipi.it; alessandro.berti@phd.unipi.it; gianna.delcorso@unipi.it).

Valentina Ciriani is with the Department of Computer Science, Università degli Studi di Milano, 26913 Crema, Italy (e-mail: valentina.ciriani@unimi.it).

Digital Object Identifier 10.1109/TCAD.2022.3213214

The main goal of our experiments is first to verify whether ESOP standard minimization is sensitive to the autosymmetry regularity and it is able to exploit it. The experimental results clarify that this regularity is not detected, therefore, a second set of experiments aims at analyzing the gain of using a preprocessing phase, detecting autosymmetry, and representing functions in XORAX form, before standard reversible synthesis.

To better evaluate the quality of the circuits obtained, we study their cost in terms of elementary quantum gates, after the technology mapping using the *Clifford+T* library [1], [19], [27]. We consider two different methods for quantum compilation: one based on standard ESOP minimization, and a more recent one presented in [23] that finds a quantum circuit starting from XOR-AND graph (XAG) representations of logic functions. In both cases, our experiments show that the autosymmetry-based strategy guarantees gains in area (measured in terms of the number of elementary quantum gates) of about 40%–45% and requires 41%–43% fewer ancillary qubits, on average, for the subset of nondegenerate autosymmetric functions.

We finally observe that any other quantum compilation method could be applied, with the only requirement of adding (before and after the quantum circuit for f_k) a small number of elementary quantum gates (CNOTs) needed to implement the projection of the original function f onto f_k.

This article is organized as follows. Motivations and previous work on quantum logic synthesis are reviewed in Section II. Preliminaries on autosymmetric functions, ESOP forms, quantum computing, and reversible circuits are given in Section III. Sections IV introduces the new three-level XORAX form, and Section V discusses the strategy for constructing compact reversible circuits starting from XORAX forms. Section VI reports the experimental results. Finally, Section VII concludes the work.

II. Motivations and Related Work

In this section, we briefly discuss the motivations of our work and review some representative recent work on logic synthesis for quantum computing.

In the last few years, we have witnessed a growing interest in quantum computing motivated by the technological enhancement in quantum architectures. The new architectures will be able to both solve new problems and increase the performance of algorithms in many different application domains by exploiting the inherent parallelism provided by quantum computers. At the same time, there still are several issues that should be addressed, for instance, regarding the quantum circuit implementations of subroutines given as classical functions. In this regard, we recall that many quantum algorithms, including Shor's and Grover's algorithms, usually require to compute some classical logic functions, the so-called *oracles* [27]. According to the postulates of quantum Mechanics, the evolution of quantum systems is described by unitary operators, which are reversible. Thus, this implies that logic functions and oracles must be first realized in terms of reversible logic networks, whose gates are then translated into unitary quantum gates.

Moreover, today's quantum computers only support a very limited set of quantum operations natively. Thus, non-native operations have to be decomposed into sequences of simpler native operations of the targeted quantum hardware. This decomposition should be performed with the goal of minimizing the overall gate count of the quantum circuits to be executed.

This is precisely the reference context of our current work. We intend to establish whether the concept of regularity (in particular, autosymmetry) can ease the synthesis of Boolean functions in the context of reversible and quantum circuit synthesis, and can lead to circuits with a reduced number of elementary quantum gates, thus mitigating the impact of combinational logic on the cost of quantum algorithms.

Standard approaches to synthesize quantum oracles generally consist of two steps. The logic function is first embedded into a classical reversible circuit containing only reversible gates, as for instance multiple-controlled (MC) Toffoli gates derived from an ESOP expression [12], [14] (these notions are reviewed in Section III-D). The second step consists of mapping each reversible gate into a sequence of elementary quantum gates. Indeed, since quantum oracles are typically formulated in an abstract way that does not take into account the restrictions imposed by the physical hardware, one or more compilation steps are always required in order to execute them on a given target hardware.

Recently, many new methods for reversible circuit synthesis and quantum compilation have been proposed in the literature. In [39], a hierarchical reversible synthesis approach based on k-feasible Boolean logic networks is proposed. These are logic networks in which every gate has at most k inputs, often referred to as k-LUT (lookup table) networks. This approach to reversible circuit synthesis outperforms other existing state-of-the-art hierarchical methods [37], [38], [42].

In [24], an efficient method to map reversible single-target gates into a universal set of quantum gates (the *Clifford + T library* [1], [19], [27]) is discussed. The proposed mapping method aims at reducing the cost of the resulting quantum network. This article also proposes a post-synthesis optimization method to further reduce the cost of the final quantum network.

Finally, Meuli et al. [23] presented a constructive compilation algorithm that finds a quantum circuit starting from XAG representations of Boolean functions, i.e., Boolean networks consisting only of two input AND gates, two input XOR gates, and that can have constant one inputs. A very interesting result of this work is that the number of elementary quantum T gates in the final quantum circuit for a given function f depends on the number of AND gates used to represent the function f in XAG form. Thus, this result suggests a new upper bound on the number of T gates proportional to the *multiplicative complexity* of the function f, i.e., to the minimum number of AND gates required to realize f in XAG form [11]. Since, as proved in [5], the autosymmetry property can be exploited to better estimate the multiplicative complexity of functions, this approach to quantum compilation should be particularly convenient for the class of autosymmetric functions.

Other approaches have been proposed and discussed in the recent literature. In contrast to standard approaches, Zulehner

and Wille [45] proposed a new strategy for designing reversible circuits based on the idea of combining the embedding step, in which additional variables are added to the circuit, and the synthesis steps that produce the reversible circuit, into a single one, thus avoiding the growth in complexity of the function representation.

In [2], a post-synthesis optimization technique based on templates is proposed. The authors introduce several templates to merge two or more consecutive multiple-Toffoli gates and then replace the high-cost subroutines with equivalent low-cost solutions. Along with the templates, they also provide an algorithm for template-matching and thus for optimizing reversible circuits.

Finally, we refer the reader to [31] for more details on reversible circuits, and to [43] for an overview of efficient quantum compilation methods.

III. Preliminaries

A. Autosymmetric Functions

Autosymmetric Boolean functions [9], [10], [20] exhibit a special type of regularity based on the notion of affine spaces and are easily expressed using XORs. A function f over n variables is *autosymmetric* if it can be projected onto a function depending on a smaller number of variables and containing a reduced number of minterms. More precisely, the autosymmetry of a Boolean function f is measured computing an *autosymmetry degree* k, with $0 \leq k \leq n$: $k = 0$ means no regularity, while for $k \geq 1$, f is said *autosymmetric*, and a new function f_k depending on $n - k$ variables only and called the *restriction* of f, is identified in time polynomial in the dimension of some standard representations of the original function f, for instance, the dimension of a reduced ordered binary decision diagram (ROBDD) for f [9]. In particular, Bernasconi and Ciriani [4] and Bernasconi et al. [9] described implicit procedures for the test, working on BDD representations of incompletely and completely specified Boolean functions, respectively.

An autosymmetric function f can be expressed in terms of its restriction f_k as follows:

$$f(x_1, x_2, \ldots, x_n) = f_k(y_1, y_2, \ldots, y_{n-k})$$

where the $n - k$ variables y_1, y_2, …, y_{n-k} are given by XOR combinations of subsets of the original input variables x_i's. These combinations are denoted $\mathrm{XOR}(X_i)$, where $X_i \subseteq \{x_1, x_2, \ldots, x_n\}$, and the equations

$$y_i = \mathrm{XOR}(X_i) \qquad i = 1, \ldots, n - k$$

are called *reduction equations*. Thus, the autosymmetry test consists of finding the value of k, the restriction f_k, and every single XOR with its input variables (i.e., reduction equations).

The autosymmetry of a function can be exploited for any logic minimization problem, according to the overall structure shown in Fig. 1. In fact, the reduction from f to f_k can be obtained with an additional logic level of XOR gates, whose inputs are the original variables $x_1, \ldots, x_n$ and the outputs are the new variables $y_1, \ldots, y_{n-k}$, that become the inputs to a circuit for f_k. For example, in [5] and [6], autosymmetry has been exploited to ease the XAG synthesis and better evaluate the multiplicative complexity. In this work, we exploit this regularity in order to derive compact reversible and quantum circuits.

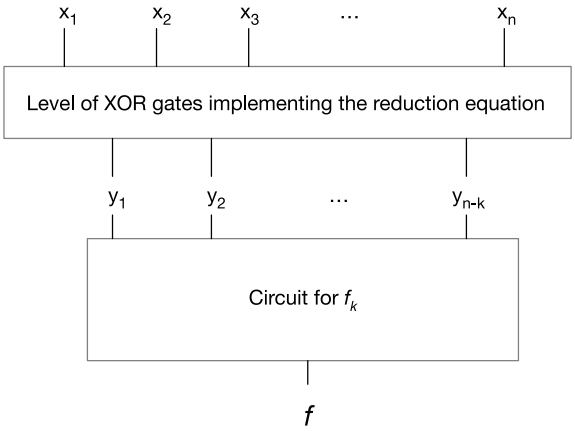

Fig. 1. General network for an autosymmetric function f, defined using a network for the restriction f_k, and an XOR layer implementing the reduction equations [9]. The restriction f_k can be synthesized in any framework of logic minimization, e.g., in SOP form, ESOP form, as an And-Inverter graph, as an XAG, etc. In particular, if f_k is represented in ESOP form, the overall circuit corresponds to an XORAX expression.

As shown in [9] and [10], any k-autosymmetric function f is associated to a k-dimensional vector space L_f, defined as the set of all minterms α s.t.

$$f(x) = f(x \oplus \alpha)$$

for all $x \in \{0, 1\}^n$. The k variables that are truly independent onto L_f are called *canonical variables*, while the other variables are called *noncanonical*. Informally, the canonical variables are the ones that assume all the possible combinations of $\{0, 1\}$ values in the vectors of the vector space L_f, meanwhile, the noncanonical variables are the variables that on L_f, have a constant value or are a linear combination of the canonical ones. Moreover, the restriction f_k corresponds to the projection of f onto the subspace $\{0, 1\}^{n-k}$ where all the canonical variables assume value 0, while the reduction equations correspond to the linear combinations that define each noncanonical variable in terms of the canonical ones.

The restriction f_k is "equivalent" to, but smaller than f, and has $|S(f)|/2^k$ minterms only, where $S(f)$ denotes the support of f, and thus, $|S(f)|$ is the number of minterms of f. As we will discuss in this article, the reversible synthesis of f can be reduced to the synthesis of its restriction f_k, which can be identified in polynomial time, as already mentioned.

Example 1: Consider the function f, depending on $n = 6$ variables, depicted in Fig. 2(a). It is possible to verify that f is autosymmetric, with vector space $L_f = \{000000, 001100, 110000, 111100\}$. Since the dimension of L_f is $k = 2$, f is 2-autosymmetric. The canonical variables are x_1 and x_3. Indeed, x_1 and x_3 assume all possible combinations of 0 and 1 values onto L_f, meanwhile x_2 and x_4 are always equal to x_1 and x_3, respectively, and x_5 and x_6 are constant and equal to 0. The reduction equations are $y_1 = x_1 \oplus x_2$, $y_2 = x_3 \oplus x_4$, $y_3 = x_5$, and $y_4 = x_6$ (for details on the computation see [9]). Thus, the restriction f_2 depends on four variables only and corresponds

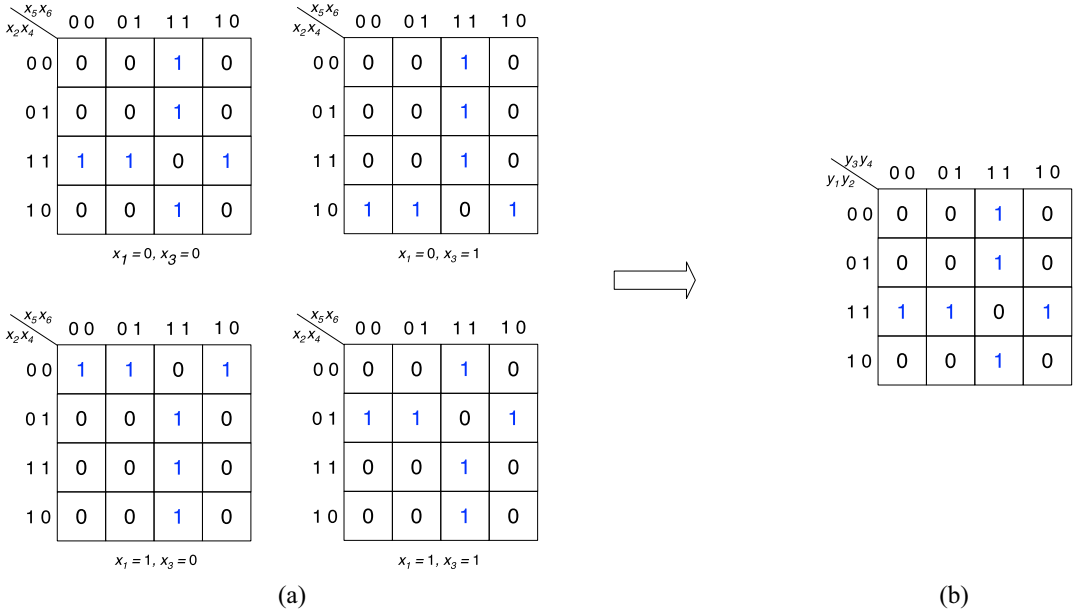

Fig. 2. (a) 2-autosymmetric Boolean function f in a Karnaugh map of six variables and (b) its restriction f_2 that depends on the new input variables $y_1 = x_1 \oplus x_2$, $y_2 = x_3 \oplus x_4$, $y_3 = x_5$, and $y_4 = x_6$.

to the projection of the function f onto the space where $x_1 = 0$ and $x_3 = 0$, as it can be noted from Fig. 2(a) and (b). Finally, we can reconstruct f from f_2 in the following way: $f(x_1, \ldots, x_6) = f_2(x_1 \oplus x_2, x_3 \oplus x_4, x_5, x_6)$.

Autosymmetric functions are a subset of the total number of Boolean functions. Nevertheless, as already mentioned, a considerable amount of functions of practical interest falls in the class of autosymmetric functions, i.e., about 24% of the functions in the classical ESPRESSO benchmark suite [44] has at least one truly (i.e., non degenerate) autosymmetric output [9], [10]. Moreover, autosymmetry occurs especially for functions that exhibit a regular structure, as for instance arithmetic functions. For this reason, compact reversible networks for autosymmetric functions might be interesting for quantum computing.

B. ESOP Forms

An ESOPs form is a two-level Boolean expression consisting of one level of AND gates followed by one XOR gate on the second level (i.e., is an XOR of ANDs of literals). The problem of synthesizing an ESOP form for a given Boolean function f thus consists of identifying a set of product terms over the Boolean input variables of f such that each minterm in the off-set of f is covered by the product terms an even number of times, or never, while each minterm in the on-set is covered an odd number of times. Such a representation is not unique. A Boolean function can be expressed by different, but semantically equivalent, ESOP forms.

Example 2: Consider the function f depicted in Fig. 2(a). f can be represented in ESOP form with five products and ten literals as follows:

$$\text{ESOP}(f) = x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_2x_4 \oplus x_5x_6. \quad (1)$$

Observe that the on-set minterm 000011 is covered only once by the product x_5x_6, the on-set minterm 111111 is covered by all the five products in the ESOP, the off-set minterm 111100 is covered four times, and so on. An alternative and functionally equivalent ESOP representation for f is given by the following expression:

$$\text{ESOP}(f) = x_1\bar{x}_2x_3 \oplus x_1x_4 \oplus \bar{x}_1x_2x_3 \oplus x_2x_4 \oplus x_5x_6.$$

ESOP forms play an important role in logic synthesis, design for tests, and other areas of the computer technology. Indeed, they require fewer products than standard SOP forms to realize randomly generated functions [34], [35]. They are more compact than other two-level representations, especially for arithmetic or communication circuits [33]. They have excellent testability properties [15], [17]. Finally, thanks to the reversibility of the XOR operation, ESOP forms represent the backbone of synthesis schemes for reversible logic circuits [12], [14], [16] and are therefore important for quantum computing.

For all these reasons, in the last decades, several algorithms have been proposed for exact and heuristic minimization of ESOP forms [26], [28], [29], [30], [32], [36], with the aim of reducing the overall costs of their hardware realizations and software implementations. Heuristic methods focus on finding small (but not necessarily minimum) ESOP forms; they are fast, but only examine a subset of the possible search space. Heuristic methods, e.g., the Exorcism approach [26], usually operate in two phases. In the first phase, an ESOP form with a suboptimal number of product terms is derived, e.g., by translating each minterm of the function into one product term. In the second phase, the ESOP form is iteratively optimized and reshaped using cube transformations with the overall goal of merging as many product terms as possible. The second phase terminates when, after several iterations, no further size reduction is achieved. For this reason, heuristic methods produce

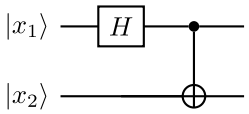

Fig. 3. Simple quantum circuit.

small ESOP forms in reasonable time, but suffer from local minima that cannot be easily escaped. On the other hand, exact methods try to find an ESOP form with a minimum number of product terms, but they hardly can deal with more than eight Boolean variables and a few product terms. Recently, an exact synthesis method based on Boolean satisfiability (SAT) has been proposed in [30]. This novel approach is hardly affected by the number of Boolean variables and turns out to be particularly fast if the Boolean function can be expressed by using only a reduced number of product terms.

C. Qubits and Quantum Operations

The unit of information of classical computation is the bit which can be either in state 0 or 1. Instead, the unit of information of quantum computation is the *qubit* and its state is a superposition of two basis states, $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ and $|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$ corresponding to the states 0 and 1 for a classical bit. The state $|\psi\rangle$ of a qubit is described mathematically by a linear combination of the two basis states

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix}\alpha\\\beta\end{bmatrix}, \quad \alpha, \beta \in \mathbb{C}, \quad |\alpha|^2 + |\beta|^2 = 1.$$

The evolution of a n-qubit quantum system is described by the multiplication of the state-vector, a vector of 2^n complex parameters, by $2^n \times 2^n$ unitary matrices. Each quantum computation is then described as the composition of elementary quantum operations, or quantum gates, that act on one or two qubits and implement the abstract unitary matrices. This article uses the so-called *Clifford+T library*, a universal and fault-tolerant library. The generator gates in the Clifford+T group are

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}, \; S = \begin{bmatrix}1 & 0\\0 & i\end{bmatrix}, \; T = \begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$$

and

$$\text{CNOT} = \begin{bmatrix}I & 0\\0 & X\end{bmatrix}, \quad \text{where } X = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}.$$

This group includes also the Pauli matrices, which can be obtained by multiplying H and S gates, for example, for the NOT gate denoted by X, we have $X = HS^2H$.

Quantum circuits are diagrams used to visualize the flux of computation. Each qubit is associated with a horizontal line (or a wire) and each operation (gate) with a box. Controlled gates are important gates acting on more than a qubit. Among them, the CNOT gate is the simplest and most important example. The NOT operation on the target (second) qubit is performed only when the control (first) qubit is $|1\rangle$; otherwise, it is left unchanged. Fig. 3 is depicted a simple circuit composed of a Hadamard gate (H gate) on the qubit $|x_1\rangle$ followed by a CNOT.

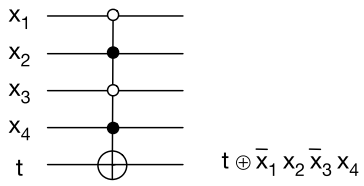

Fig. 4. MCMP Toffoli gate with four control lines, with polarities 0, 1, 0, and 1, respectively, where polarity 0 is represented by an empty bullet.

D. Reversible Circuits

Reversible circuits are circuits with the same number of input and output signals that implement bijections, i.e., each input assignment maps to a unique output assignment. As a consequence, reversible computations can not only be performed from the inputs to the outputs but also in the other direction [13]. A reversible circuit computes a *reversible Boolean function*, i.e., a bijective function that uniquely maps each input assignment to a unique output assignment and vice versa. Recall that an irreversible function can always be converted into a reversible one by adding extra outputs such that the input–output mapping is unique.

Reversible logic networks are usually composed as cascades of *Toffoli gates* [22], [25]. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ be the set of input variables in the reversible circuit. An *MC mixed-polarity (MCMP) Toffoli* gate $T(C, t)$ has a (possibly empty) set $C \subset \mathcal{X}$ of control lines, which are literals over $\mathcal{X}$, and one target line t which is a variable in $\mathcal{X} \setminus C$.

The MCMP Toffoli gate inverts the value of the variable assigned to the target line, if and only if the polarity of all inputs to the control lines match their polarity, or if $C = \emptyset$. All remaining values are passed through unaltered. More precisely, given $C = \{x_{c_1}, x_{c_2}, \ldots, x_{c_k}\} \subset \mathcal{X}$, the gate maps the target line t to

$$t \oplus \left(x_{c_1}^{p_1} x_{c_2}^{p_2} \cdots x_{c_k}^{p_k}\right)$$

where $p_1, p_2, \ldots, p_k$ denote the polarities, 0 or 1, of the control variables (i.e., $x_{c_i}^{p_i}$ corresponds to x_{c_i} if $p_i = 1$, and to $\bar{x}_{c_i}$ otherwise). Fig. 4 illustrates one MCMP Toffoli gate with four control qubits.

An MCMP Toffoli gate is called an *MC Toffoli* gate if all the control variables are positive literals, i.e., their polarities are all equal to 1.

Note that an MC Toffoli gate corresponds to the *NOT (X)* quantum gate if $C = \emptyset$, to the CNOT quantum gate if C contains only one variable, and to the standard *Toffoli* gate (used for computing the logic AND between two variables) if C contains two variables [27].

We can observe that there is a natural correspondence between product terms in an ESOP expression and MCMP Toffoli gates, and this is the reason why ESOP-based logic synthesis is widely used in reversible logic synthesis [12], [14], [16]. Indeed, given an ESOP expression, one can extract a sequence of MCMP Toffoli gates whose control lines correspond to the literals in the product terms of the ESOP form. Observe that, when considering multiple-output functions, the resulting reversible circuit will contain $n+m$ lines, where n is

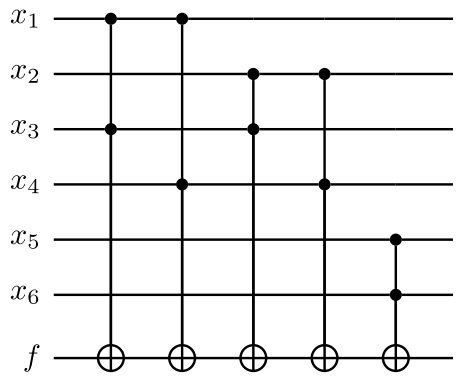

Fig. 5. Reversible circuit for the function f, derived from the ESOP representation (1).

the number of input variables and m is the number of outputs of the function.

Example 3: Consider again the function f in our running example. From its ESOP representation (1), we can build the reversible circuit depicted in Fig. 5. The circuit contains five standard Toffoli gates computing the AND of their two control variables.

Reversible logic synthesis, i.e., the ability to generate a reversible circuit for a given target function, has many applications, especially in the area of quantum computing. Indeed, as discussed in Section II, quantum algorithms consist of quantum and classical operations, and classical operations are first realized in terms of reversible logic networks. Such networks must then be mapped into quantum circuits using an additional synthesis step, whose goal is to transform reversible circuits of MCMP Toffoli gates into functionally equivalent quantum circuit implementations: input lines are translated into qubits and reversible gates, as MCMP Toffoli gates are implemented into quantum circuits referring to a chosen quantum gate library [22], [25]. This process often requires additional support qubits, that are called *ancillary qubits*. In this work, as already mentioned, we will refer to the Clifford+T library for this mapping step.

IV. XORAX Synthesis

For any given function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, the algorithms for computing the autosymmetry degree k, the restriction f_k and the reduction equations are polynomial in the dimension of an ROBDD for f [9]. Obviously, note that the dimension of the ROBDD of f might be exponential in the number n of input variables.

Thus, in the synthesis process for $f(x_1, \ldots, x_n)$, we could first obtain and implement the reduction from f to f_k, through the construction of the new variables $y_1, \ldots, y_{n-k}$ from the original ones $x_1, \ldots, x_n$ by an additional level of XOR gates, and then synthesize the restriction $f_k(y_1, \ldots, y_{n-k})$ in any two-level logic framework. This gives rise to a three-level form for autosymmetric functions.

In general, we expect that the time required by the minimization of f_k will be obviously less than the time required for the minimization of f, since f_k is defined on a smaller space: it depends only on $n - k$ variables and has $|S(f)|/2^k$ minterms only, where $|S(f)|$ is the number of on-set minterms of f. A similar approach has been studied and proven to be particularly convenient within standard sum of products (SOP) minimization. A three-level OR-AND-XOR form (shortly, *ORAX form*) has been defined and experimentally evaluated in [10].

Unfortunately, the OR operator is not suited for reversible logic synthesis. For this reason, here, we focus on ESOP synthesis and introduce a new three-level XOR-AND-XOR form for autosymmetric functions, which we call XORAX expression. In particular, since several methods in quantum synthesis start from ESOP (XOR-AND) forms instead of SOP (OR-AND) expressions, we consider here XORAX forms, that are a natural extension of ESOP forms, instead of ORAX (OR-AND-XOR) expressions.

As already mentioned, this expression can be derived by first running the autosymmetry test to detect the reduction equations and the restriction f_k, and then, by synthesizing in ESOP form the restriction f_k only.

The reduction from f to f_k is obtained through an initial logic level composed of XOR gates that implement the reduction equations. The inputs are the original variables $x_1, \ldots, x_n$ and the outputs are the new variables $y_1, \ldots, y_{n-k}$, that become the inputs to a minimal ESOP form for f_k, according to the network decomposition depicted in Fig. 1. Notice that if the function f is not autosymmetric, XORAX and ESOP expressions coincide.

Example 4: Consider again the function f and its restriction f_2 depicted in Fig. 2. Recall from Example 1 that the reduction equations are $y_1 = x_1 \oplus x_2$, $y_2 = x_3 \oplus x_4$, $y_3 = x_5$, and $y_4 = x_6$. A minimal ESOP for f_2 is given by $\text{ESOP}(f_2) = y_1 y_2 \oplus y_3 y_4$, and the XORAX of f is then obtained by adding a level of XORs

$$\text{XORAX}(f) = (x_1 \oplus x_2)(x_3 \oplus x_4) \oplus x_5 x_6. \tag{2}$$

Note how this expression is more compact than the ESOP representation (1) for f: it contains only six literals and two products, while $\text{ESOP}(f)$ contains five products and ten literals.

As shown also by this simple example, in general, a minimal XORAX form has a smaller size than a standard minimal ESOP form for the same function, thanks to the extra level of XOR gates. Moreover, the gain in size can also become exponential, as proved in the following proposition.

Proposition 1: An XORAX form for a function f can be exponentially smaller than a minimal ESOP expression for f.

Proof: To prove the proposition, we show an example of a function whose XORAX contains only one product of XOR combinations, while a minimal ESOP contains an exponential number of products. Consider the autosymmetric function f, that depends on $n =$ 2-m binary variables and whose XORAX form is

$$\text{XORAX}(f) = (x_1 \oplus x_2)(x_3 \oplus x_4) \cdots (x_{n-1} \oplus x_n).$$

This form contains n literals and just one product of $m = n/2$ XOR combinations of two variables each. This is a consequence of the fact that f is m-autosymmetric, with reduction

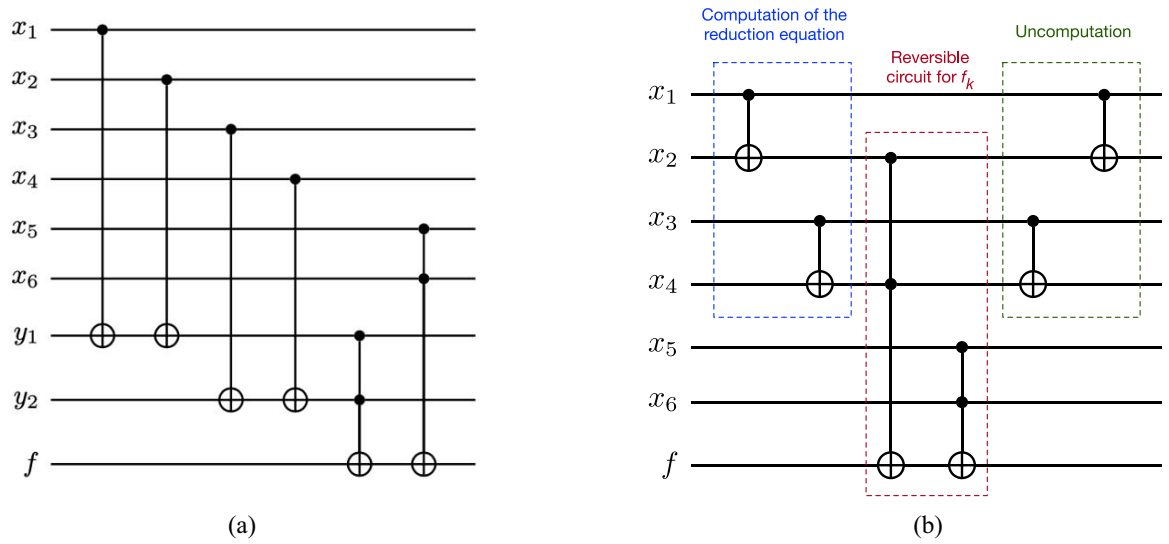

Fig. 6. Reversible circuits, (a) with and (b) without new input lines, for the function f of the running example, derived from its XORAX representation (2). The last two CNOTs of the circuit (b) are used for restoring the two noncanonical variables x_2 and x_4 to their initial values.

equations $y_i = x_{2i-1} \oplus x_{2i}$, $1 \leq i \leq m$, and with restriction $f_m = y_1\, y_2\, \ldots,\, y_n$.

A minimal ESOP for f instead contains $2^m = 2^{n/2}$ products, each containing m variables. This can be proved by induction on n, exploiting the fact that, when we add a new factor $(x_{n+1} \oplus x_{n+2})$, the resulting function $f' = f \cdot (x_{n+1} \oplus x_{n+2})$ is such that the cofactors obtained assigning a value to x_{n+1} and x_{n+2}, satisfy $f'_{00} = f'_{11} = 0$ and $f'_{01} = f'_{10} = f$. This, in turn, implies that any ESOP for f' must contain an ESOP for f multiplied by x_{n+1}, and one multiplied by x_{n+2}. Thus, the number of products doubles each time leading to an exponential growth. ■

V. Construction of Reversible Circuits From XORAX Forms

As reviewed in Sections III-B and III-D, ESOP-based logic synthesis is widely used in reversible logic synthesis because of the natural correspondence between product terms in an ESOP expression and MCMP Toffoli gates.

In particular, given an ESOP expression, one can easily extract a sequence of MCMP Toffoli gates whose control lines correspond to the literals in the product terms of the ESOP form, and whose target lines correspond to the output of the function. Notice that all MCMP Toffoli gates act on the same target line, thus realizing the exclusive OR sum of all product terms.

In this section, we propose a method for deriving compact reversible circuits computing k-autosymmetric functions, starting from their representation in XORAX form. The idea of the approach is to concatenate two circuits: 1) a circuit implementing the transformation from the old input variables $X = \{x_1, x_2, \ldots, x_n\}$ to the new $n-k$ variables and 2) a circuit implementing the restriction f_k.

Observe that this approach will require $n-k$ new lines (and therefore $n-k$ new qubits in the quantum implementation of the reversible circuit for f), corresponding to the new variables $y_1, \ldots, y_{n-k}$.

Each reduction equation $y_i = \text{XOR}(X_i)$, where $X_i \subseteq X$ and $i = 1, \ldots, n-k$, is then implemented through CNOT gates, one for any variable in X_i. The control line of each CNOT corresponds to a different variable in X_i, eventually complemented using a NOT gate, while the target line corresponds to the new variable y_i. Then, the new variables $y_1, \ldots, y_{n-k}$ are used as control variables in the cascade of MCMP Toffoli gates implementing a reversible circuit for the restriction f_k, as shown in Fig. 6(a).

The number of lines, and therefore of qubits, grows up to a factor 2 with respect to a standard reversible circuit build from an ESOP of f, but the reduced number of products and literals in the ESOP for f_k should compensate the cost of the additional input lines, trading-off the number of qubits for the number of quantum gates.

However, due to the current technological limitations of quantum circuits (current machines contain just a few dozen qubits), it is very important to keep the number of input qubits as low as possible. Fortunately, the structural properties of autosymmetric functions can be better exploited, so that no additional input qubits must be added to the network for the new variables, as proved in the following proposition.

Proposition 2: Let $f : \{0,1\}^n \to \{0,1\}$ be a k autosymmetric function, with vector space L_f and restriction f_k. Then, a reversible circuit for f can be derived from a reversible circuit for the restriction f_k without adding new input lines, nor new MCMP Toffoli gates.

Proof: Recall from Section III-A that any k-autosymmetric function f is associated to a k-dimensional vector space L_f. Onto this space, only the k canonical variables are truly independent, i.e., can assume all the possible combinations of $\{0,1\}$ values. The remaining noncanonical variables have a constant value or are defined by linear combinations of the canonical ones.

These linear combinations can be used to derive the reduction equations, that precisely define each noncanonical variable in terms of the canonical ones. As proved in [9], each noncanonical variable appears in one and only one reduction equation, alone or in XOR combination with a subset of canonical variables. Thus, instead of adding new inputs, we can implement each equations $y_i = \text{XOR}(X_i)$, simply modifying the only noncanonical variable in X_i.

In particular, the noncanonical variable is used as target variable, while the remaining canonical variables in X_i, if any, are used as control variables in the CNOTs used to implement the linear combination defining the noncanonical one. The procedure is correct since the noncanonical variable that has been modified does not occur in any other reduction equation. Finally, we can use the $n-k$ modified noncanonical variables as input variables for the reversible circuit implementing f_k. ■

Once f has been computed on the output line, we can restore the noncanonical variables to their initial values applying the so-called *uncomputing procedure* [3], [27]. Basically, we uncompute the linear combinations stored in the noncanonical variables by reapplying the CNOT gates used to implement the corresponding reduction equations in reverse order. This disentangles the variables, reverting them to their initial values.

Example 5: Let us consider the function f in our running example, and its XORAX representation (2). The reduction equations $y_1 = x_1 \oplus x_2$, $y_2 = x_3 \oplus x_4$, $y_3 = x_5$, and $y_4 = x_6$ can be translated into the first two *CNOT* gates in Fig. 6(b) (blue box). The ESOP representation of the restriction f_2, $ESOP(f_2) = y_1y_2 \oplus y_3y_4$, is represented by the two *CNOT*s in the red box. Finally, the last two CNOT gates (green box) are used for the uncomputing procedure on the noncanonical variables.

We finally observe that the methodology that we have proposed to derive a reversible circuit for an autosymmetric function is very flexible. Here, we have considered the standard technique based on ESOP minimization, however, we could apply any other reversible synthesis technique to the restriction f_k, with the only requirement of adding (before and after the reversible circuit for f_k) the CNOT gates needed to implement and then to uncompute the reduction equations.

VI. Experimental Results

In this section, we report our experimental results. First of all, we consider the practical gain, in terms of area, of XORAX forms with respect to ESOP expressions. (Recall that, by theoretical results, the area gain can be exponential.) Second, we evaluate the proposed decomposition in the context of reversible and quantum circuit synthesis, considering two different quantum compilation strategies.

Note that as several methods in quantum synthesis start from ESOP (XOR-AND) forms instead of SOP (OR-AND) expressions, we consider XORAX (XOR-AND-XOR) forms instead of ORAX (OR-AND-XOR) expressions.

Our method has been tested on autosymmetric functions taken from the LGSynth'89 benchmark suite [44] and the EPFL benchmark suite [40], [41]. Observe that autosymmetry is a property of single outputs, i.e., different outputs of the same benchmark can have different autosymmetry degrees. Therefore, we perform the autosymmetry test on single outputs of the benchmark functions.

TABLE I
Cost of k-Controlled Toffoli Gates in Number of T, H, and CNOT Gates

k	T	H	*CNOT*	ancillary qubits
2	7	2	6	0
3	16	6	14	1
≥ 4	8k-8	8k-12	4k-6	$\lceil \frac{k-2}{2} \rceil$

We consider only *proper* autosymmetric functions, i.e., autosymmetric functions whose reduction equations contain at least one XOR operator. Notice that this choice excludes all functions whose autosymmetry is only due to the fact that they are degenerate.

The experiments have been run on a Pentium Intel core i7-6700HQ, 4-core, 2.60-GHz processor with 16-GB RAM.

The first aim of the experiments is to compare the XORAX expression of an autosymmetric function and the corresponding ESOP form. For this purpose, we perform the autosymmetry test [4] (for noncompletely specified functions) on the benchmark outputs. The functions f and f_k are then minimized in ESOP form, using EXORCISM-4 [26]. The average gain, in using XORAX expressions with respect to ESOP ones, is about 40% of literals in the form. Due to the heuristic nature of the ESOP minimizer, the synthesis times for the function f and the corresponding restriction f_k are quite similar and very short (of order of 10^{-2} s on average). Therefore, in our experiments, the gain in synthesis time is negligible.

This set of experiments verifies that the standard ESOP minimization is not sensitive to autosymmetry. Therefore, we propose a preprocessing phase detecting autosymmetry before standard reversible synthesis, as shown in the following set of experiments.

A. ESOP- and XORAX-Based Quantum Circuit Synthesis

We now evaluate the impact of using XORAX forms for autosymmetric functions in quantum circuit synthesis. We emphasized in Section III-D the correspondence between ESOP expressions and multicontrolled Toffoli gates. However, controlled Toffoli gates are only an intermediate representation that needs to be mapped into elementary quantum gates. To perform this additional step, we select the universal fault-tolerant *Clifford+T* quantum gate library [1], [19], [27] as low-level gate library.

The cost model for this last synthesis step is nontrivial. In general, heuristics try to minimize the number of MCMP Toffoli gates and the number of control lines inside each gate in the reversible circuit that must be transformed into a quantum one. This in turn corresponds to minimize the number of products in the ESOP expression and the number of literals in each product, respectively, as smaller ESOP expressions naturally lead to fewer and smaller MCMP Toffoli gates.

TABLE II
COMPARISON BETWEEN QUANTUM CIRCUITS COMPUTED WITHOUT AND WITH AUTOSYMMETRY TEST, STARTING FROM ESOPS

		without autosymmetry test				with autosymmetry test				
Benchmark_output (autosymmetry degree)	**input**	**T**	**H**	**CNOT**	**ancillary qubits**	**T**	**H**	**CNOT**	**ancillary qubits**	**T gain**
add6_5 (1)	12	1031	454	916	46	1031	454	117	54	0%
apla_5 (1)	10	128	60	120	6	24	10	23	1	81%
dk48_9 (1)	15	208	100	200	12	16	6	58	1	92%
in2_7 (2)	19	976	454	912	48	912	424	1526	55	7%
in5_3 (5)	24	904	424	848	47	880	414	37	51	3%
m181_6 (7)	15	247	110	222	11	151	66	3	8	39%
max1024_2 (1)	10	1124	516	1041	55	504	226	753	29	55%
max1024_3 (1)	10	1767	808	1625	88	808	366	1311	47	54%
max1024_5 (1)	10	4383	2016	4041	208	1999	908	38	111	54%
opa_16 (3)	17	448	208	418	22	344	158	22	21	23%
pdc_22 (4)	16	384	184	368	20	24	10	30	1	94%
pdc_26 (2)	16	976	468	936	52	64	28	154	3	93%
pdc_30 (1)	16	1784	852	1706	95	95	40	320	4	95%
spla_15 (5)	16	232	110	220	12	120	56	120	7	48%
spla_39 (3)	16	560	268	536	32	208	98	2314	13	63%
voting_N_2_M_2.pla_dbb_orig_1 (1)	8	832	366	734	47	471	204	416	24	43%
voting_N_2_M_3.pla_dbb_orig_1 (1)	16	203272	96804	193611	12077	119749	56880	113780	7105	41%
Average Benchmark Suite		**1210**	**571**	**1143**	**67**	**653**	**305**	**623**	**38**	**46%**

As reviewed in Section III-C, the Clifford+T library is composed of the Pauli, Hadamard, and CNOT gates and of the additional non-Clifford T gate. Since the T gate is considered the most expensive gate in the library, usually the cost of a Toffoli gate is expressed in the number of T gates needed for its realization and the algorithms for the mapping of k-controlled Toffoli gates minimize respect to this measure. In Table I, we report the cost in terms of Hadamard, CNOTs, and T gates of the realization of k-controlled Toffoli gates with the algorithm described in [21]. We report also the number of ancillary qubits used. Ancillas indeed impact the efficiency of connections between primary qubits. Both kinds of qubits share the same physical space and they need to be close to each other to ensure high efficiency.

Table II reports a significant subset of benchmarks as representative indicators of our experiments. The first column reports the name, the number of the considered output of each benchmark, and the autosymmetry degree. The second column reports the number of inputs of the benchmark. The following group of four columns reports the costs, in terms of elementary quantum gates, of the quantum circuits derived from the ESOP representation of the functions, while the last group of four columns reports the costs of the circuits derived from the XORAX representation. The last column reports the gain in the number of T gates. The last row reports the average costs for all the benchmarks considered in our experiments.

The gain obtained synthesizing a quantum circuit starting from XORAX forms instead of ESOP ones is quite interesting. Indeed, the cost gain for T gates is about 46%, the cost gain for H gates is about 47%, the cost gain for CNOTs is about 45%, and the gain in ancillary qubits is about 43%, as shown in Table III.

Some particular benchmarks highly benefit from the proposed strategy. For example, the benchmark *dk48_9* can be represented, exploiting the autosymmetry of the function, with a gain of 92%, in T gates. We can notice that benchmarks with a high autosymmetry degree often have a very high gain in T gates. See, for example, *pdc_22* that has an autosymmetry degree 4 and has a gain of about 94%.

TABLE III
GAIN FOR T, H, CNOT GATES, AND ANCILLARY QUBITS FOR THE ENTIRE BENCHMARK SUITE

T	H	*CNOT*	ancillary qubits
46%	47%	45%	43%

TABLE IV
GAIN FOR T AND CNOT GATES, T-DEPTH, AND ANCILLARY QUBITS FOR THE ENTIRE CONSIDERED BENCHMARK SUITE

T	*CNOT*	T-depth	ancillary qubits
43%	39%	23%	41%

For other benchmarks we do not obtain any significant gain, for example, *add6_5* and *in5_3*. We can notice that there are cases, e.g., *in5_3*, where even if the autosymmetry degree is high, e.g., 5, the gain in T nodes is not relevant. In the specific case of the benchmark *in5_3*, we can notice that the proposed method gives a very high gain in terms of CNOT gates. This is due to the fact that, using our method, the XORs of the circuit have been factorized in the reduction equations.

Finally, we can observe that even if, in general, arithmetic functions are autosymmetric, this does not directly imply that arithmetic functions always benefit from this strategy. In fact, the arithmetic benchmark *add6_5* is autosymmetric, but the proposed method does not give any gain in terms of T gates. Also in this case, the main advantage is the gain in CNOT gates.

B. Comparison With XAG-Based Quantum Compilation [23]

To further confirm the efficacy of the autosymmetry-based synthesis strategy, we run another set of experiments considering the recent quantum compilation method proposed in [23]. Indeed, as already pointed out, autosymmetry can be exploited by different optimization methods, simply adapting the general decomposition scheme shown in Fig. 1 to the chosen framework. In the context of quantum compilation, this task simply requires to add (before and after any quantum circuit for the restriction f_k) some CNOT gates, acting on the noncanonical input variables of f, in order to implement, and then to

TABLE V
COMPARISON BETWEEN QUANTUM CIRCUITS COMPUTED WITHOUT AND WITH AUTOSYMMETRY TEST, STARTING FROM XAGS (COMPUTATIONAL TIMES ARE OF A FEW MILLISECONDS ON AVERAGE AND ARE NOT REPORTED)

		without autosymmetry test				with autosymmetry test				
Benchmark_output (autosymetry degree)	**input**	**T**	**T_depth**	**CNOT**	**ancillary qubits**	**T**	**T_depth**	**CNOT**	**ancillary qubits**	**T gain**
add6_5 (1)	12	32	7	38	21	20	6	58	17	38%
apla_5 (1)	10	44	5	0	21	32	5	2	17	27%
dk48_9 (1)	15	52	6	4	28	52	5	2	27	0%
in2_7 (2)	19	136	11	102	53	136	14	106	51	0%
in5_3 (5)	24	100	13	68	45	108	13	38	46	-8%
m181_6 (7)	15	32	6	34	18	40	6	2	18	-25%
max1024_2 (1)	10	300	22	94	85	120	17	168	40	60%
max1024_3 (1)	10	436	42	290	119	236	27	226	68	46%
max1024_5 (1)	10	1220	70	512	315	624	36	260	166	49%
opa_16 (3)	17	100	13	60	40	96	12	34	38	4%
pdc_22 (4)	16	60	7	34	30	44	5	10	23	27%
pdc_26 (2)	16	96	10	16	40	48	6	18	26	50%
pdc_30 (1)	16	64	11	54	32	64	10	44	31	0%
spla_15 (5)	16	48	6	4	24	40	5	30	21	17%
spla_39 (3)	16	64	7	44	30	48	5	6	25	25%
voting_N_2_M_2_1 (1)	8	324	31	248	89	88	14	48	29	73%
voting_N_2_M_3_1 (1)	16	37044	478	6834	9277	20832	378	4168	5223	44%
Average Benchmark Suite		**480**	**13**	**117**	**129**	**275**	**10**	**72**	**76**	**43%**

uncompute, the reduction equations. Recall that, as proved in Proposition 2, this task does not require any additional ancillary qubit.

These last experiments were run on two AMD EPYC 7282 processors with 16 cores each, at 2.8 GHz, with 504 GB of RAM in total.

The quantum compilation heuristic discussed in [23] starts from an XAG representation of a Boolean function f (a network consisting only of two input AND and XOR gates, and that can have constant one inputs), and targets quantum circuits over the Clifford+T gate set.

The experimental results show that, also for this method, the autosymmetry-based approach guarantees an overall reduction not only in the number of T gates, but also in the number of CNOT gates, T-depth, ancillary qubits, and compilation time. In particular, compiling a quantum circuit starting from an XAG for f_k, instead of an XAG for f, we can obtain a cost gain in T gates of about 43%, a gain in CNOTs of about 39%, a gain in T-depth of 23%, and a gain in the number of ancillary qubits of about 41%, as shown in Table IV. The computational times required by this compilation step are very short (a few milliseconds on average), with autosymmetry-based compilation about 66% faster on average.

We report in Table V a subset of all the benchmarks that are considered for our experiments. The first column contains the name, the number of the output, and the autosymmetry degree of the considered benchmark. The second column reports the number of inputs of the benchmark. The following group of four columns reports the costs, in terms of elementary quantum gates, of the quantum circuit derived from the XAG representation of the function f, while the last group of four columns reports the costs of the quantum circuit for f derived from the XAG for f_k and the reduction equations. The last column reports the gain in the number of T gates. The last row reports the average costs for all the benchmarks considered in our experiments.

We first observe that the overall T cost of the XAG-based quantum compiler is much lower than the corresponding cost of the circuits derived from ESOP forms (see Tables II and V).

Moreover, from Table V, we observe that there are cases where the proposed strategy gives a circuit with a higher number of T gates (see for instance, *in5_3* and *m181_6*). This fact is due to the heuristic nature of the XAG optimization and of the quantum compiler. In any case, the overall T gain of the entire benchmark suite is very encouraging (i.e., 43%). Interesting enough comparing the two Tables II and V, we notice that the autosymmetry property gives very different T gains. For example, *add6_5* shows no T gain in Table II and presents a T gain of about 38% in Table V. Conversely, the benchmark *dk48_9* has no T gain for the XAG approach (Table V) and 92% of T gain in Table II. This is due to the very different approaches of the two quantum compilers considered.

VII. CONCLUSION

In this article, we have proposed and evaluated a methodology to derive compact reversible circuits for autosymmetric functions. This methodology is very flexible and can be applied to any quantum compilation method. We have experimentally tested two different compilation methods, the classic one based on ESOP minimization, and a more recent one, based on XAG representations of logic functions.

As future work, it would be interesting to verify whether other quantum synthesis techniques are sensitive to this structural regularities and are able to exploit it in the optimization process.

REFERENCES

[1] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 6, pp. 818–830, Jun. 2013.

[2] C. Bandyopadhyay, R. Wille, R. Drechsler, and H. Rahaman, "Post synthesis-optimization of reversible circuit using template matching," in *Proc. 24th Int. Symp. VLSI Design Test (VDAT)*, Jul. 2020, pp. 1–4.

[3] C. H. Bennett, "Logical reversibility of computation," *IBM J. Res. Develop.*, vol. 17, no. 6, pp. 525–532, 1973.

[4] A. Bernasconi and V. Ciriani, "Autosymmetry of incompletely specified functions," in *Proc. Design Autom. Test Europe (DATE)*, 2021, pp. 360–365.

[5] A. Bernasconi, S. Cimato, V. Ciriani, and M. C. Molteni, "Multiplicative complexity of autosymmetric functions: Theory and applications to security," in *Proc. 57th ACM/IEEE Design Autom. Conf.*, Jul. 2020, pp. 1–6.
[6] A. Bernasconi, S. Cimato, V. Ciriani, and M. C. Molteni, "Multiplicative complexity of XOR based regular functions," *IEEE Trans. Comput.*, vol. 71, no. 11, pp. 2927–2939, Nov. 2022.
[7] A. Bernasconi, V. Ciriani, L. Frontini, and G. Trucco, "Composition of switching lattices for regular and for decomposed functions," *Microprocess. Microsyst.*, vol. 60, pp. 207–218, Jul. 2018.
[8] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Three-level logic minimization based on function regularities," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 8, pp. 1005–1016, Aug. 2003.
[9] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Exploiting regularities for Boolean function synthesis," *Theory Comput. Syst.*, vol. 39, no. 4, pp. 485–501, 2006.
[10] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Synthesis of autosymmetric functions in a new three-level form," *Theory Comput. Syst.*, vol. 42, no. 4, pp. 450–464, 2008.
[11] J. Boyar, R. Peralta, and D. Pochuev, "On the multiplicative complexity of boolean functions over the basis (cap, +, 1)," *Theor. Comput. Sci.*, vol. 235, no. 1, pp. 43–57, 2000.
[12] R. Drechsler, A. Finder, and R. Wille, "Improving ESOP-based synthesis of reversible logic using evolutionary algorithms," in *Applications of Evolutionary Computation* (Lecture Notes in Computer Science 6625), C. D. Chio et al., Eds. Heidelberg, Germany: Springer, 2011, pp. 151–161.
[13] R. Drechsler and R. Wille, "From truth tables to programming languages: Progress in the design of reversible circuits," in *Proc. 41st IEEE Int. Symp. Multiple-Valued Logic*, 2011, pp. 78–85.
[14] K. Fazel, M. A. Thornton, and J. E. Rice, "ESOP-based Toffoli gate cascade generation," in *Proc. IEEE Pacific Rim Conf. Commun., Comput. Signal Process.*, 2007, pp. 206–209.
[15] E. Goto and H. Takahasi, "Some theorems useful in threshold logic for enumerating Boolean functions," in *Proc. 2nd IFIP Congr. Inf. Process.*, Munich, Germany, 1962, pp. 747–752.
[16] P. Gupta, A. Agrawal, and N. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 11, pp. 2317–2330, Nov. 2006.
[17] U. Kalay, D. V. Hall, and M. A. Perkowski, "A minimal universal test set for self-test of EXOR-sum-of-products circuits," *IEEE Trans. Comput.*, vol. 49, no. 3, pp. 267–276, Mar. 2000.
[18] V. Kravets and K. Sakallah, "Generalized symmetries of Boolean functions," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2000, pp. 526–532.
[19] N. Linke et al., "Experimental comparison of two quantum computing architectures," *Proc. Nat. Acad. Sci.*, vol. 114, no. 13, pp. 3305–3310, Feb. 2017.
[20] F. Luccio and L. Pagli, "On a new Boolean function with applications," *IEEE Trans. Comput.*, vol. 48, no. 3, pp. 296–310, Mar. 1999.
[21] D. Maslov, "Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization," *Phys. Rev. A*, vol. 93, Feb. 2016, Art. no. 22311.
[22] D. Maslov, G. W. Dueck, and D. M. Miller, "Synthesis of Fredkin-Toffoli reversible networks," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 6, pp. 765–769, Jun. 2005.
[23] G. Meuli, M. Soeken, E. Campbell, M. Roetteler, and G. D. Micheli, "The role of multiplicative complexity in compiling low T-count oracle circuits," in *Proc. Int. Conf. Comput.-Aided Design*, 2019, pp. 1–8.
[24] G. Meuli, M. Soeken, M. Roetteler, N. Wiebe, and G. D. Micheli, "A best-fit mapping algorithm to facilitate ESOP-decomposition in clifford+T quantum network synthesis," in *Proc. 23rd Asia South Pacific Design Autom. Conf.*, 2018, pp. 664–669.
[25] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proc. 40th Design Autom. Conf.*, 2003, pp. 318–323.
[26] A. Mishchenko and M. Perkowski, "Fast heuristic minimization of exclusive-sums-of-products," in *Proc. 5th Int. Reed-Muller Workshop*, 2001, pp. 1–9.
[27] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2016.
[28] G. K. Papakonstantinou, "A parallel algorithm for minimizing ESOP expressions," *J. Circuits Syst. Comput.*, vol. 23, no. 1, 2014, Art. no. 1450015.
[29] K. G. Papakonstantinou and G. Papakonstantinou, "A nonlinear integer programming approach for the minimization of Boolean expressions," *J. Circuits Syst. Comput.*, vol. 27, no. 10, pp. 1–29, 2018.
[30] H. Riener, R. Ehlers, B. Schmitt, and G. De Micheli, "Exact synthesis of ESOP forms," in *Advanced Boolean Techniques*, R. S. M. Drechsler, Ed. Cham, Switzerland: Springer, 2020.
[31] M. Saeedi and I. L. Markov, "Synthesis and optimization of reversible circuits—A survey," *ACM Comput. Surveys*, vol. 45, no. 2, pp. 21:1–21:34, 2013.
[32] M. Sampson, M. Kalathas, D. Voudouris, and G. K. Papakonstantinou, "Exact ESOP expressions for incompletely specified functions," *Integration*, vol. 45, no. 2, pp. 197–204, 2012.
[33] T. Sasao, "Representation of logic functions using EXOR operators," in *Representation of Discrete Functions*, T. Sasao and M. Fujita, Eds. Boston, MA, USA: Springer, 1996.
[34] T. Sasao, "EXMIN: A simplification algorithm for exclusive-OR-sum-of-products expressions for multiple-valued input two-valued output functions," in *Proc. 20th Int. Symp. Multiple-Valued Logic*, 1990, pp. 128–135.
[35] T. Sasao, "Logic synthesis with EXOR gates," in *Logic Synthesis and Optimization*, T. Sasao, Ed. Boston, MA, USA: Kluwer Acad., 1993, pp. 259–286.
[36] B. Schmitt, M. Soeken, G. De Micheli, and A. Mishchenko, "Scaling-up ESOP synthesis for quantum compilation," in *Proc. IEEE 49th Int. Symp. Multiple-Valued Logic (ISMVL)*, 2019, pp. 13–18.
[37] M. Soeken and A. Chattopadhyay, "Unlocking efficiency and scalability of reversible logic synthesis using conventional logic synthesis," in *Proc. 53rd Annu. Design Autom. Conf.*, 2016, pp. 149:1–149:6.
[38] M. Soeken, M. Roetteler, N. Wiebe, and G. D. Micheli, "Design automation and design space exploration for quantum computers," in *Proc. Design, Autom. Test Europe Conf. Exhibit.*, 2017, pp. 470–475.
[39] M. Soeken, M. Roetteler, N. Wiebe, and G. D. Micheli, "LUT-based hierarchical reversible logic synthesis," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 38, no. 9, pp. 1675–1688, Sep. 2019.
[40] E. Testa, M. Soeken, H. Riener, L. Amaru, and G. D. Micheli, "A logic synthesis toolbox for reducing the multiplicative complexity in logic networks," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2020, pp. 568–573.
[41] E. Testa, M. Soeken, L. G. Amarù, and G. D. Micheli, "Reducing the multiplicative complexity in logic networks for cryptography and security applications," in *Proc. 56th Annu. Design Autom. Conf.*, 2019, p. 74.
[42] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic for large functions," in *Proc. 46th Design Autom. Conf.*, 2009, pp. 270–275.
[43] R. Wille, S. Hillmich, and L. Burgholzer, "Efficient and correct compilation of quantum circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2020, pp. 1–5.
[44] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide Version 3.0, User Guide*, Microelectron. Center, Hilliard, OH, USA, 1991.
[45] A. Zulehner and R. Wille, "One-pass design of reversible circuits: Combining embedding and synthesis for reversible logic," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 5, pp. 996–1008, May 2018.

Anna Bernasconi received the Laurea degree in physics from the University of Pavia, Pavia, Italy, in 1992, and the Ph.D. degree in computer science from the University of Pisa, Pisa, Italy, in 1998.

She is currently an Associate Professor with the Department of Computer Science, University of Pisa, where she teaches fundamental courses in the Computer Science Program. She has authored or coauthored more than 80 research papers, published in international journals, conference proceedings, books, and books chapters. Her research interests include Boolean function complexity, combinational logic synthesis, and logic synthesis for quantum computing.

Alessandro Berti received the B.S. and M.S. degrees in computer science from the University of Pisa, Pisa, Italy, in 2017 and 2020, respectively, where he is currently pursuing the Ph.D. degree.

His research topics are related to quantum algorithms.

Valentina Ciriani (Senior Member, IEEE) received the Laurea and the Ph.D. degrees in computer science from the University of Pisa, Pisa, Italy, in 1998 and 2003, respectively.

In 2003 and 2004, she was with the Department Computer Science, University of Pisa as a Ph.D. Fellow. From January 2005 to February 2015, she was an Assistant Professor with the Department Information Technologies and the Department of Computer Science, Università degli Studi di Milano, Crema, Italy, where she is currently an Associate Professor of Computer Science with the Department of Computer Science. She has authored or coauthored more than 100 research papers, published in international journals, conference proceedings, and books chapters. Her research interests include algorithms and data structures, as well as combinational logic synthesis for classical and emerging technologies, VLSI design of low power circuits and testing of Boolean circuits.

Gianna M. Del Corso received the Laurea degree in computer science form the University of Pisa, Pisa, Italy, and the Ph.D degree in applied mathematics from the University of Milano, Milano, Italy.

She is currently an Associate Professor of Numerical Analysis with the Computer Science Department, University of Pisa. During her Ph.D. degree with the University of Milano, she has been a Visiting Scholar with the Computer Science Department, Columbia University, New York, NY, USA, and a Visiting Researcher with the Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA. She joined the Computer Science Department, University of Pisa in June 2000. Her main scientific interests range from the study of algorithms for eigenvalue computation of low-rank perturbation of unitary matrices to the use of spectral techniques for data classification. In the quantum computing field, she is especially fascinated by the study of state preparation, quantum machine learning techniques, and discrete-time quantum walks.

Innocenzo Fulginiti received the B.S. degree in computer science from the University of Pisa, Pisa, Italy, in 2021, where he is currently pursuing the M.S. degree.

Open Access funding provided by 'Università degli Studi di Roma' within the CRUI CARE Agreement