Formal Methods for Emerging Technologies

(Invited Paper)

Robert Wille Rolf Drechsler
Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
Cyber Physical Systems, DFKI GmbH, 28359 Bremen, Germany
{rwille,drechsle}@informatik.uni-bremen.de

***Abstract*—Formal methods advanced to an important core technique in Computer-Aided Design (CAD). At the same time, researchers and engineers also started the investigation of so-called emerging technologies such as reversible computation, quantum computation, or optical circuits. Although most of these technologies are still in a rather "academic" state, first physical realizations have already been presented. This motivates a more detailed consideration of how to design circuits for these technologies. As for conventional circuits, formal methods do play an important role here. In this tutorial paper[1], we are aiming to address the current momentum caused by the recent accomplishments and provide an overview of these emerging technologies as well as their corresponding CAD methods. This includes a special focus on how formal methods may help in the design and verification of circuits for those technologies.**

I. Introduction

Formal methods advanced to an important core technique in *Computer-Aided Design* (CAD). Prominent examples include representations for Boolean functions such as *Decision Diagrams* (DDs, [1]) or solvers for satisfiability problems such as SAT- or SMT-solvers [2], [3], [4]. They are heavily utilized not only in obvious application areas like verification but can also be exploited for other CAD-tasks including synthesis, optimization, test pattern generation, etc. This allowed for impressive improvements in the design for conventional circuit technologies.

At the same time, mainly caused by the expected physical boundaries and cost restrictions of conventional CMOS-based circuitry, researchers and engineers also started the investigation of so-called emerging technologies such as:

- Reversible circuits, in which all operations are assumed to be bijective. This could be beneficial e.g. for certain low-power applications or the design of encoders/decoders. At the same time, reversible logic provides a basis for quantum computation.
- Quantum circuits, in which quantum-mechanical effects (e.g. superposition or entanglement) are exploited in order to represent multiple (Boolean) states at the same time and, thus, allow for massive parallelism.
- Optical Circuits, which rely on optical rather than electrical signals and, hence, find useful applications for ultra-high-speed networks and optical interconnects.

Although most of these technologies are still in a rather "academic" state, first physical realizations have already been presented. This motivates a more detailed consideration of how to design circuits for these technologies. As for conventional circuits, formal methods do play an important role here.

In this paper, we are aiming to address the current momentum caused by the recent accomplishments by providing an overview of these emerging technologies as well as their corresponding CAD methods. This includes a special focus on how formal methods may help in the design of circuits for those technologies and what adjustments would be necessary for this purpose. More precisely, the application of the following formal methods will be addressed:

- *Decision Diagrams:* Decision diagrams allow for an efficient representation of the desired (Boolean) functionality to be realized. By the application of various decomposition strategies, they are a core method for the realization of conventional logic. However, when the represented functionality is supposed to be realized in terms of an alternative emerging technology, new representations, objectives, or decomposition schemes are required.
- *SAT Solvers:* SAT solvers as well as their derivatives (e.g. SMT solvers, QBF solvers, PBO solvers, etc.) allow for an efficient traversal of the search space. This is of benefit e.g. for many CAD problems in which an (optimal) solution within a rather large space of non-optimal (or even invalid) solutions has to be determined. If emerging technologies are considered, additional constraints such as non-Boolean representations have to be additionally addressed.

In order to cover these issues, the remainder of this paper is structured as follows. First, the basics on the considered emerging technologies are summarized in Section II. Afterwards, it is discussed how formal methods can be utilized in the design of circuits for emerging technologies in Section III (using decision diagrams) and Section IV (using SAT solvers). Due to the amount of covered content, all descriptions are kept brief but references for a detailed treatment are provided. Finally, the paper is concluded in Section V.

II. Considered Emerging Technologies

For each emerging technology considered in this paper, this section first reviews its basics and the main application areas. Afterwards, the logic model which is applied in order to design the corresponding circuits is reviewed.

A. Reversible Circuits

1) Basics & Motivation: In reversible circuits, functionality is specified and, eventually, realized in a bijective fashion, i.e. a unique input/output mapping is enforced. In particular, applications in the domain of quantum computation (covered in the next section) profit from the corresponding functional descriptions. This is because every quantum operation inherently is reversible and, hence, Boolean components of quantum circuits can first be realized as reversible circuits, before they are mapped to quantum circuits [5], [6]. Besides that, certain aspects of low-power design may profit from the reversible computation paradigm according to the observations

[1]This paper is a summary (including a list of references) of a tutorial which has been given at ICCAD'2015.

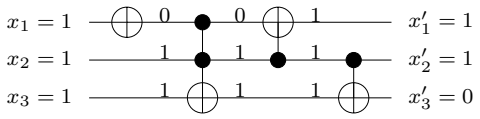


Fig. 1: A reversible circuit

by Landauer [7] and Bennett [8]. They respectively proved that, during a computation, each information loss causes a certain amount of power dissipation and, since reversible computations never lead to an information loss, every circuit technology which aims for a (theoretical) power dissipation of zero, indeed has to be reversible. This has been experimentally confirmed recently in [9]. Also in the domain of the design of encoders for on-chip interconnects, reversible circuits have successfully been applied [10].

2) Logic Model: A Boolean function $f : \mathbb{B}^n \to \mathbb{B}^m$ over the variables $X := \{x_1, \ldots, x_n\}$ is *reversible* iff (1) its number of inputs is equal to the number of outputs (i.e. $n = m$) and (2) it maps each input pattern to a unique output pattern. Since fanout and feedback are not directly allowed, reversible circuits realizing reversible functions are represented as a cascade of reversible gates [11]. Each variable of the function f is represented by a *circuit line*, i.e. a signal through the whole cascade structure on which the respective computations are performed. Computations are performed by *reversible gates*, whereby the *Toffoli gate* is the most frequently used gate type. A Toffoli gate is composed of a (possibly empty) set of *control lines* $C = \{x_{i_1}, \ldots, x_{i_k}\} \subset X$ and a single *target line* $x_j \in X \setminus C$. The Toffoli gate inverts the value on the target line if all values on the control lines are assigned to 1 or if $C = \emptyset$, respectively. All remaining values are passed through unaltered.

Example 1. *Fig. 1 shows a reversible circuit composed of several Toffoli gates. Control lines are denoted by ●, while the target lines are denoted by ⊕. This circuit maps e.g. the input* 111 *to the output* 110*. As can be seen, this computation can also been conducted in the reverse fashion, i.e. the output* 110 *is mapped to the input* 111*.*

B. Quantum Circuits

1) Basics & Motivation: Quantum computation [11] provides a new way of computation based on so called *qubits*. In contrast to conventional bits, qubits do not only allow to represent the (Boolean) basis states 0 and 1, but also superpositions of both. By this, qubits can represent multiple (Boolean) states at the same time which enables massive parallelism. Additionally exploiting further quantum mechanical phenomena such as phase shifts or entanglement enables asymptotic speed-ups for many relevant problems (e.g. database search [12] or integer factorization [13]), offers new methods for secure communication (e.g. quantum key distribution), and has several other appealing applications [11].

2) Logic Model: A *qubit* is a two-level quantum system, described by a two-dimensional complex Hilbert space. The two orthogonal quantum states $|0\rangle \equiv \binom{1}{0}$ and $|1\rangle \equiv \binom{0}{1}$ are used to represent the Boolean values 0 and 1. The state of a qubit may be written as $|x\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers with $|\alpha|^2 + |\beta|^2 = 1$. A *quantum operation* over n qubits can be represented by a *unitary matrix*, i.e. a $2^n \times 2^n$ matrix $\mathbf{U} = [u_{i,j}]_{2^n \times 2^n}$ with

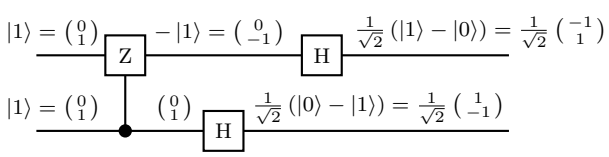


Fig. 2: A quantum circuit

- each entry $u_{i,j}$ assuming a complex value and
- the inverse $\mathbf{U}^{-1}$ of $\mathbf{U}$ being the conjugate transpose matrix (adjoint matrix) $\mathbf{U}^\dagger$ of $\mathbf{U}$ (i.e. $\mathbf{U}^{-1} = \mathbf{U}^\dagger$).

At the end of the computation, a qubit can be measured causing it to collapse to a basis state. Then, depending on the current state of the qubit, either a 0 (with probability of $|\alpha|^2$) or a 1 (with probability of $|\beta|^2$) results. The state of the qubit is destroyed by the act of measuring it. The corresponding quantum operations are usually represented in terms of quantum gates that are performed in a predetermined serial fashion – eventually leading to the representation of a quantum circuit. Similar to reversible gates, quantum gates may have control lines.

Example 2. *Fig. 2 shows a quantum circuit composed of several quantum gates. Again, control lines are denoted by ●, while the target line is denoted by a box indicating the respectively applied unitary operation, namely*

$$H = \tfrac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \text{ and } Z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}.$$

This circuit transforms the basis states shown at the inputs to the superposed states shown at the outputs. Measuring the qubits at the outputs would either lead to a Boolean 0 or a Boolean 1 with a probability of $|\pm\tfrac{1}{\sqrt{2}}|^2 = 0.5$ *each.*

C. Optical Circuits

1) Basics & Motivation: Optical computations are performed on optical rather than electrical signals. This is of interest particularly for ultra-high-speed networks and optical interconnects [14], [15]. In such systems, the respective signals frequently (i.e. at every interconnect interface) need to be transformed from the electrical domain to the optical domain and vice versa. This causes significant overhead which could be avoided if the respective transformations would directly be conducted at the optical signals. To this end, researchers from the field of silicon-based integrated optics (also known as *silicon photonics*) considered the realization of optical circuits.

2) Logic Model: Optical circuits allow to realize Boolean functionality e.g. by means of so called *crossbar gates*. A *crossbar gate* realizes a Boolean function $\mathbb{B}^3 \to \mathbb{B}^2$ composed of two optical inputs p and q, one electrical input x, and two optical outputs f and g. The signals p and q as well as f and g are connected by waveguides which, depending on the value of x, realize either the identity or a switch of the input values, i.e. $\overline{x} \Rightarrow (p \equiv f) \wedge (q \equiv g)$ and $x \Rightarrow (p \equiv g) \wedge (q \equiv f)$ is realized, respectively[2]. Besides that, *splitters* (dividing an optical signal into two optical signals with half the signal power each) and *combiners* (merging two optical signals into a single one) are utilized in order to realize logic functions.

[2]Note that an optical as well as an electrical signal are never assumed to interact with each other except for the crossbar gate.

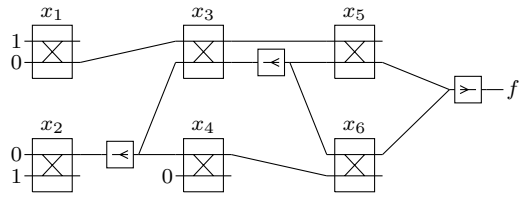
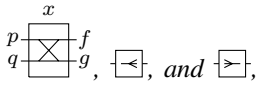


Fig. 3: An optical circuit

Example 3. *Fig. 3 shows an optical circuit composed of six crossbar gates, two splitters, and one combiner denoted by*



respectively.

III. Using Decision Diagrams for the Design of Emerging Technologies

In a first consideration, the design of circuits for emerging technologies based on decision diagrams [1] is addressed. Decision diagrams allow for a compact representation of the desired Boolean functionality to be realized what made them a suitable description means for many conventional synthesis approaches (see e.g. [16], [17]). Accordingly, these concepts have been adapted for the emerging technologies as well.

In this section, we briefly review the basics on decision diagrams first. Afterwards, how to exploit this formal method for the design of circuits for an emerging technology is exemplary shown by means of reversible circuits.

A. Decision Diagrams

A Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ can be represented by a *Decision Diagram* (DD) [1]. A DD is a directed acyclic graph $G = (V, E)$ where e.g. a Shannon decomposition

$$f = \overline{x}_i \cdot f_{x_i=0} + x_i \cdot f_{x_i=1}$$

is carried out in each node $v \in V$. The function $f_{x_i=0}$ ($f_{x_i=1}$) is the negative (positive) co-factor of f obtained by assigning x_i to 0 (1). In the following, the node representing $f_{x_i=0}$ ($f_{x_i=1}$) is denoted by $\mathrm{low}(v)$ ($\mathrm{high}(v)$), while x_i is called the select variable. A DD is called *free* if each variable is encountered at most once on each path from the root to a terminal node. A DD is called ordered if in addition all variables are encountered in the same order on all such paths. The size *k* of a DD is defined by the number of nodes.

In the past, several techniques to optimize the size of DDs have been developed. In particular *shared nodes* [1] allow significant reductions which can particularly be exploited by functions $f : \mathbb{B}^n \to \mathbb{B}^m$ (i.e. functions with more than one output). Further reduction can be achieved if *complement edges* [18] are applied. This enables the representation of a function as well as of its negation by a single node only. Furthermore, the size of a DD significantly depends on the chosen ordering of its input variables [1].

Example 4. *Fig. 4 shows a DD representing the function* $f = \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1x_2x_3\overline{x}_4 + x_1\overline{x}_2x_3\overline{x}_4 + x_1x_2\overline{x}_3x_4$ *as well as the respective co-factors resulting from the application of the Shannon decomposition.*

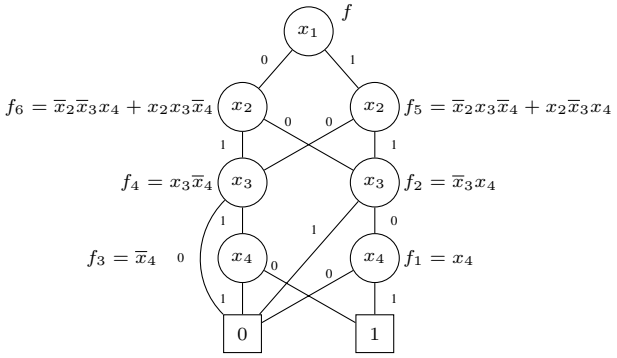


Fig. 4: Decision diagram

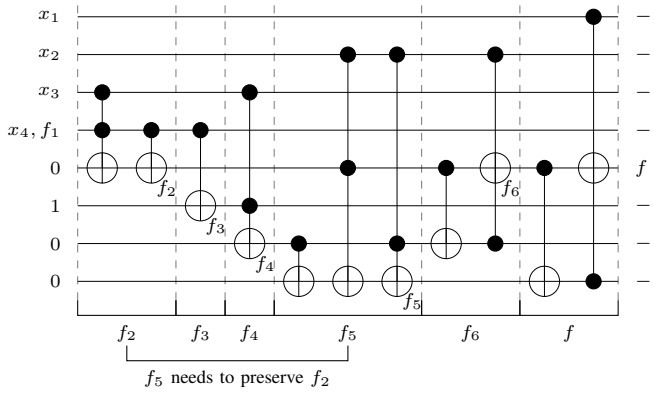


Fig. 5: Circuit derived from DD

B. DD-based Synthesis of Reversible Circuits

Having a DD $G = (V, E)$ representing the function to be realized, a corresponding reversible circuit can easily be derived from it. To this end, all nodes $v \in V$ of G are traversed in a depth-first fashion and substituted with a cascade of reversible gates. The respective cascade of gates depends on the successors of the node v. Note that this sometimes requires an additional (constant) circuit line. If the entire DD has been traversed, a circuit realizing f has been obtained.

Example 5. *Consider again the DD from Fig. 4. The co-factor* f_1 *can easily be represented by the primary input* x_4*. Having the value of* f_1 *available, the co-factor* f_2 *can be realized by the first two gates depicted in Fig. 5*[3]*. In this manner, respective sub-circuits can be added for all remaining co-factors until a circuit representing the overall function* f *results. The remaining steps are shown in Fig. 5.*

DD-based synthesis of reversible circuits constituted one of the first synthesis approaches for reversible circuits that allowed for the realization of larger Boolean functions (i.e. functions with 100 variables and more). However, they inherited the drawback that, as sketched in the example from above, circuits with a significant amount of additional lines result – usually considered a serious drawback. Hence, further developments considered alternatives which addressed this drawback [19], [20].

[3]Note that an additional circuit line is added to preserve the values of x_4 and x_3 which are still needed by the co-factors f_3 and f_4, respectively.

C. DD-based Synthesis for Other Emerging Technologies

Approaches for the synthesis of circuits for other emerging technologies employed a similar scheme as sketched above, i.e. representing the desired functionality in terms of a decision diagram and, afterwards, using this description in order to create the desired circuits. But depending on the considered technology, additional issues have to be considered for this purpose. More precisely:

1) For Quantum Circuits: As reviewed in Section II-B, quantum circuits do not only rely on Boolean values and operations. Consequently, circuit synthesis for this technology additionally has to support the logic representation of quantum mechanical properties such as superposition, entanglement, etc. This cannot directly been represented by binary decision diagrams, but requires dedicated DDs for quantum computations. Amongst others, particular *Quantum Multiple-valued Decision Diagrams* (QMDDs; originally introduced in [21]; a comprehensive description including recent improvements can be found in [22]) have been found useful for this purpose. Although a simple mapping scheme as sketched above was, thus far, only presented in [19], QMDDs nevertheless have been successfully applied to synthesize certain quantum functionality as shown in [23]. However, a general synthesis methodology capable of realizing arbitrary quantum functionality based on QMDDs is still left for future work.

2) For Optical Circuits: Decision diagrams also enabled, for the first time, an efficient synthesis of optical circuits for large functions (initially investigated in [24]). Here, particularly the gate library composed of crossbar gates (cf. Section II-C) comes in handy. In fact, the Shannon decomposition applied in each DD node is directly realized by a crossbar gate so that each node can eventually be mapped to such as gate. However, if a shared node occurs, the respective optical signal has to be split accordingly.

However, these splittings represent a serious drawback because DDs for practical relevant functions usually are composed of a significant number of shared nodes. Since each splitter caused by a shared node decreases the strength of a signal by its half, DD-based synthesis leads to optical circuits where certain output signals are constituted by a negligible fraction of power only.

An approach addressing this issue has been proposed in [25]. Here, a DD-based synthesis is introduced which traverses the DD in a reverse fashion and, by this, allows for a splitter-free circuit. This reduces the splitting of the signal strength to none, but comes at the expense of a moderate overhead in terms of crossbar gates.

IV. Using SAT Solvers for the Emerging Technologies

The problem of Boolean satisfiability is one of the central $\mathcal{NP}$-complete problems. The main idea is to determine a satisfying assignment to a Boolean function so that this function evaluates to 1 or to prove that no such assignment exists. Although this represents a computational complex (exponential) task, very efficient algorithms (so called *SAT solvers*) have been proposed in the past (see e.g. [2]). They can handle even complex instances with large search spaces. Since many practical relevant problems from CAD of conventional circuits can easily represented by means of Boolean satisfiability, SAT solvers become another core technology in many domains (e.g. for verification [26], test pattern generation as well as compaction [27], [28], and more). In fact, they represent the method of choice when large search spaces have to completely been considered, but representations e.g. by means of decision diagrams are not suitable anymore.

In the design for emerging technologies, SAT solvers are utilized in similar fashions, i.e. for verification and testing. Besides that, their computational power and consideration of the entire search space also has been found valuable in the synthesis of minimal circuits (compared to the circuits often generated by heuristic which do not guarantee minimality). How exact synthesis of circuits for emerging technologies can be conducted using SAT solvers is reviewed in this section. Again, synthesis of reversible circuits is considered as a representative in more detail. Afterwards, related work for the synthesis of circuits for other technologies is summarized. Before that, the basics of SAT solvers are reviewed first.

A. SAT Solvers

In a more formal fashion, the *Boolean Satisfiability* (SAT) problem is defined as follows: Let h be a Boolean function. Then, the SAT problem is to determine an assignment for the variables of h such that h evaluates to 1 or to prove that no such assignment exists. The function h is usually provided in *Conjunctive Normal Form* (CNF), i.e. a product-of-sum representation. More precisely, the CNF is composed of a conjunction of clauses. A clause is a disjunction of literals and each literal is a propositional variable or its negation. Once it is proven that no solution exist, an instance is called *unsatisfiable* (UNSAT); otherwise, the instance is called *satisfiable* (SAT).

Example 6. *Let $h = (x_1 + x_2 + \overline{x}_3)(\overline{x}_1 + x_3)(\overline{x}_2 + x_3)$. Then, $x_1 = 1, x_2 = 1$ and $x_3 = 1$ is a satisfying assignment for h. The values of x_1 and x_2 ensure that the first clause becomes satisfied while x_3 ensures this for the remaining two clauses. That is, this instance is satisfiable.*

In the past, SAT solvers have been proposed which, instead of simply traversing the complete space of assignments, employ intelligent decision heuristics, *conflict based learning*, and sophisticated engineering of the implication algorithm by *Boolean Constraint Propagation* (BCP) [2]. This led to an effective search procedure which can handle problem instances consisting of hundreds of thousands of variables, millions of clauses, and tens of millions of literals. Recently, SAT solvers have also been enriched with theories and further levels of descriptions. This led to the domain of *SAT Modulo Theories* (SMT) and corresponding SMT solvers as e.g. introduced in [3], [4].

B. SAT-based Synthesis of Reversible Circuits

In [29], the exact synthesis problem is formulated as a sequence of decision problems. Given the reversible function $f : \mathbb{B}^n \to \mathbb{B}^n$, it is checked whether f can be synthesized using $d = 1$ Toffoli gates. If this fails, then d is increased until a realization has been determined. Since d is iteratively increased starting with $d = 1$, minimality is ensured. The respective checks are performed by

- formulating the synthesis problem as an instance of Boolean satisfiability and
- using a SAT solver to solve this instance.

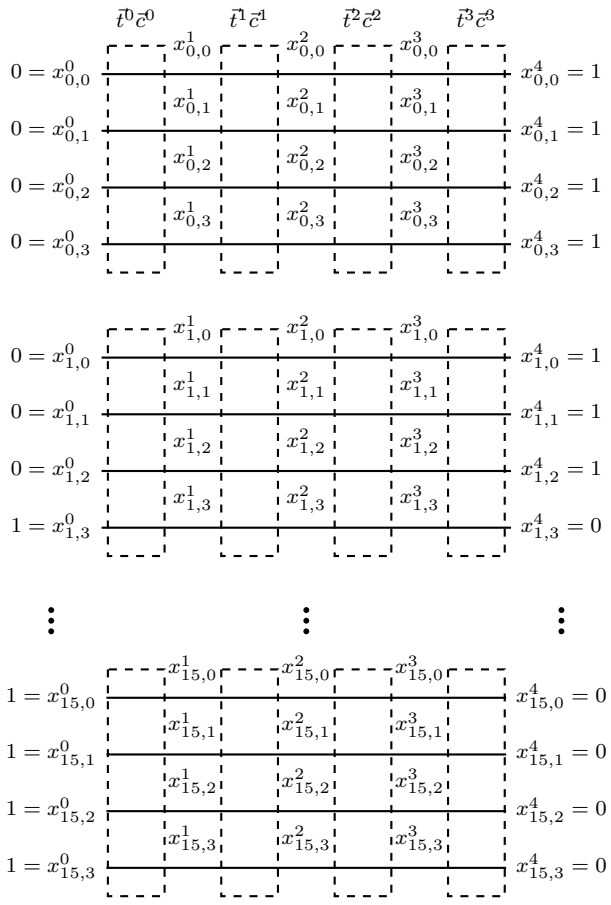


Fig. 6: Exact SAT formulation for $n = 4$ and $d = 4$.

For this purpose, SAT instances are created that become satisfiable if and only if a circuit with d gates representing the function exists. To this end, Boolean variables and constraints as introduced in the following are applied.

Definition 1. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a reversible function to be synthesized. Then, variables $x^k_{i,0}, x^k_{i,1}, \ldots, x^k_{i,n-1}$ are applied to symbolically represent the input- (for $k = 0$), the output- (for $k = d$), and the auxiliary values (for $1 \leq k \leq d-1$) of the circuit to be synthesized for each truth table line i of f. Therefore, the left-hand side of the truth table corresponds to the variables $x^0_{i,0}, x^0_{i,1}, \ldots, x^0_{i,n-1}$, while the right-hand side corresponds to the variables $x^d_{i,0}, x^d_{i,1}, \ldots, x^d_{i,n-1}$.*

Fig. 6 shows the respective variables for a function f to be synthesized with $n = 4$ variables. The first row of Fig. 6 represents the variables for the first truth table line of f, the second row the ones for the second truth table line of f, and so on.

Furthermore, variables symbolically representing the type of a Toffoli gate are introduced:

Definition 2. *Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be a reversible function to be synthesized. Then, $t^k_{\lceil \log_2 n \rceil}, t^k_{\lceil \log_2 n \rceil - 1}, \ldots, t^k_1$ and $c^k_1, c^k_2, \ldots, c^k_{n-1}$ with $0 \leq k < d$ are introduced, whose assignments symbolically represent the type of the Toffoli gate at depth k (for brevity denoted by $\vec{t}^k$ and $\vec{c}^k$ in the following).*

The variable $\vec{t}^k$ is used as a binary encoding of a natural number $t^k \in \{0, \ldots, n-1\}$ which defines the chosen target line. In contrast, $\vec{c}^k$ denotes the control lines. More precisely, assigning $c^k_l = 1$ $(1 \leq l \leq n-1)$ means that line $(t^k + l) \mod n$ becomes a control line of the Toffoli gate at depth k.

Fig. 7 gives some examples for assignments to $\vec{t}^k$ and $\vec{c}^k$ with their respective Toffoli gate representation.

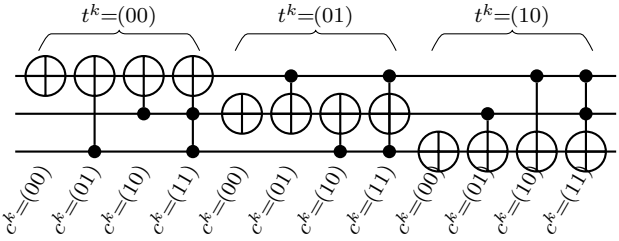


Fig. 7: Representation of Toffoli gates by assignments to $\vec{t}^k$ and $\vec{c}^k$

Using these variables, constraints are introduced assigning the input and output of the truth table to their respective $x^0_{i,j}$ and $x^d_{i,j}$ variables. Furthermore, for each gate to be synthesized at depth k, functional constraints are added so that (depending on the assignment to $\vec{t}^k$ and $\vec{c}^k$ as well as to the input $x^k_{i,j}$) the respective gate output $x^{k+1}_{i,j}$ is computed. As an example, consider $\vec{t}^k = (01)$ and $\vec{c}^k = (001)$, i.e. with $c^k_3 = 1$. This assignment states that the Toffoli gate at depth k has line $t^k = [01]_2 = 1$ as target line and line $(t^k + l) \mod n = (1+3) \mod 4 = 0$ as single control line. To cover this case, constraints

$$\vec{t}^k = (01) \wedge \vec{c}^k = (001) \Rightarrow \left[\begin{array}{lrcl} & x^{k+1}_{i,0} & = & x^k_{i,0} \\ \wedge & x^{k+1}_{i,1} & = & x^k_{i,1} \oplus x^k_{i,0} \\ \wedge & x^{k+1}_{i,2} & = & x^k_{i,2} \\ \wedge & x^{k+1}_{i,3} & = & x^k_{i,3} \end{array} \right. \quad (1)$$

are added for each truth table line i. In other words, the values of lines 0, 2, and 3 are passed through, while the output value of line 1 becomes inverted, if line 0 is assigned 1. Similar constraints are added for all remaining cases.

As a result, a functional description has been constructed which is satisfiable, if there is a valid assignment to $\vec{t}^k$ and $\vec{c}^k$ such that for all truth table lines the desired input-output mapping is achieved. Then, the precise Toffoli gates are obtained by the assignments to $\vec{t}^k$ and $\vec{c}^k$ as depicted in Fig. 7. If there is no such assignment (i.e. the instance is unsatisfiable), then it has been proven, that no circuit representing the function with d gates exists.

C. SAT-based Synthesis for Other Emerging Technologies

The computational power of SAT solvers has also been exploited for the synthesis of minimal circuits for other emerging technologies. However, depending on the respectively considered logic model as well as cost metrics, the corresponding SAT formulation required adjustments or needed to be redeveloped. More precisely:

1) For Quantum Circuits: Also the development of synthesis methods for minimal quantum circuits suffered from the fact that not only Boolean signal values and operations have to be considered. In the approach reviewed in the previous

section, logic computations based on a purely Boolean model were sufficient. If quantum values are additionally considered, a multiple-valued formulation is required. For a restricted quantum gate library (supporting four different quantum values), this was successfully accomplished in [30]. If, however, arbitrary quantum functionality is supposed to be represented, much bigger obstacles occur – in general, an infinite number of values need to be represented. However, evaluations in [31] confirmed that often a subset of all possible quantum values is sufficient. Nevertheless, how to employ SAT-based synthesis as sketched above for the minimal synthesis of arbitrary quantum functionality remains an open issue thus far.

2) For Optical Circuits: As optical circuits work on Boolean values, the SAT-based synthesis scheme can easily be adapted. In fact, only the gate library as well as the distinction between electrical signals (e.g. provided by the inputs) and optical signals (e.g. provided by the outputs) has to be considered. A corresponding synthesis scheme has been proposed and evaluated in [24].

V. Conclusions

In this tutorial paper, we provided an overview on how formal methods can be exploited for the design of circuits for emerging technologies. As representatives, synthesis of reversible circuits, quantum circuits, and optical circuits has been considered. By this, we addressed the current momentum caused by the recent accomplishments in these areas. This tutorial, however, did not cover the application of formal methods for other important design tasks such as verification, debugging, testing, etc. Also here, decision diagrams and/or SAT solvers have successfully been applied as shown e.g. in [32], [33], and [34], respectively. Besides that, a broad variety of open problems remains. The corresponding discussions from above and particularly in the cited references give an impression of issues left to be addressed in future work.

Acknowledgments

The authors would like to thank all researchers and collaborators which, in the past years, worked with us on the development of the approaches which have been reviewed in this paper.

References

[1] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[2] N. Eén and N. Sörensson. An extensible SAT solver. In *SAT 2003*, volume 2919 of *LNCS*, pages 502–518, 2004.

[3] B. Dutertre and L. Moura. The YICES SMT solver. 2006. Available at http://yices.csl.sri.com/.

[4] R. Brummayer and A. Biere. Boolector: An efficient SMT solver for bit-vectors and arrays. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 174–177, 2009.

[5] A. Barenco, C. H. Bennett, R. Cleve, D.P. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J.A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *The American Physical Society*, 52:3457–3467, 1995.

[6] D. M. Miller, R. Wille, and Z. Sasanian. Elementary quantum gate realizations for multiple-control Toffolli gates. In *Int'l Symp. on Multi-Valued Logic*, pages 288–293, 2011.

[7] R. Landauer. Irreversibility and heat generation in the computing process. *IBM J. Res. Dev.*, 5:183, 1961.

[8] C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Dev*, 17(6):525–532, 1973.

[9] A. Berut, A. Arakelyan, A. Petrosyan, S. Ciliberto, R. Dillenschneider, and E. Lutz. Experimental verification of Landauer's principle linking information and thermodynamics. *Nature*, 483:187–189, 2012.

[10] R. Wille, R. Drechsler, C. Osewold, and A. García Ortiz. Automatic design of low-power encoders using reversible circuit synthesis. In *Design, Automation and Test in Europe*, pages 1036–1041, 2012.

[11] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[12] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Theory of computing*, pages 212–219, 1996.

[13] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Foundations of Computer Science*, pages 124–134, 1994.

[14] A. Shacham, K. Bergman, and L. P. Carloni. Photonic Network-on-Chip for Future Generations of Chip Multi-Processors. *IEEE Transactions on Computers*, 57(9):1246–1260, 2008.

[15] P.K. Kaliraj, P. Sieber, A. Ganguly, I. Datta, and D. Datta. Performance Evaluation of Reliability Aware Photonic Network-on-Chip Architectures. In *Intl. Green Computing Conference*, pages 1–6, 2012.

[16] B. Becker and R. Drechsler. Decision diagrams in synthesis - algorithms, applications and extensions -. In *VLSI Design Conf.*, pages 46–50, 1997.

[17] R. Drechsler, J. Shi, and G. Fey. MuTaTe: An efficient design for testability technique for multiplexor based circuits. In *Great lakes symposium on VLSI*, pages 80–83, 2003.

[18] K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient implementation of a BDD package. In *Design Automation Conf.*, pages 40–45, 1990.

[19] M. Soeken, R. Wille, C. Hilken, N. Przigoda, and R. Drechsler. Synthesis of reversible circuits with minimal lines for large functions. In *ASP Design Automation Conf.*, pages 85–92, 2012.

[20] M. Soeken, L. Tague, G. W. Dueck, and R. Drechsler. Ancilla-free synthesis of large reversible functions using binary decision diagrams. *Journal of Symbolic Computation*, 2015.

[21] D. M. Miller and M. A. Thornton. QMDD: A decision diagram structure for reversible and quantum circuits. In *Int'l Symp. on Multi-Valued Logic*, page 6, 2006.

[22] P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler. Qmdds: Efficient quantum function representation and manipulation. *IEEE Trans. on CAD*, 2015.

[23] P. Niemann, R. Wille, and R. Drechsler. Efficient synthesis of quantum circuits implementing Clifford group operations. In *ASP Design Automation Conf.*, pages 483–488, 2014.

[24] C. Condrat, P. Kalla, and S. Blair. Logic Synthesis for Integrated Optics. In *Great lakes symposium on VLSI*, pages 13–18. ACM, 2011.

[25] R. Wille, O. Keszocze, C. Hopfmuller, and R. Drechsler. Reverse BDD-based synthesis for splitter-free optical circuits. In *ASP Design Automation Conf.*, pages 172–177, 2015.

[26] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1579 of *LNCS*, pages 193–207. Springer Verlag, 1999.

[27] R. Drechsler, S. Eggersglüß, G. Fey, A. Glowatz, F. Hapke, J. Schloeffel, and D. Tille. On acceleration of SAT-based ATPG for industrial designs. *IEEE Trans. on CAD*, 27:1329–1333, 2008.

[28] S. Eggersglüß, R. Wille, and R. Drechsler. Improved SAT-based ATPG: more constraints, better compaction. In *Int'l Conf. on CAD*, pages 85–90, 2013.

[29] D. Große, R. Wille, G. W. Dueck, and R. Drechsler. Exact multiple control Toffoli network synthesis with SAT techniques. *IEEE Trans. on CAD*, 28(5):703–715, 2009.

[30] D. Große, R. Wille, G. W. Dueck, and R. Drechsler. Exact synthesis of elementary quantum gate circuits for reversible functions with don't cares. In *Int'l Symp. on Multi-Valued Logic*, pages 214–219, 2008.

[31] R. Wille, N. Przigoda, and Rolf Drechsler. A compact and efficient SAT encoding for quantum circuits. In *AFRICON*, 2013.

[32] P. Niemann, R. Wille, and R. Drechsler. Equivalence checking in multi-level quantum systems. pages 201–215, 2014.

[33] R. Wille, D. Große, S. Frehse, G. W. Dueck, and R. Drechsler. Debugging of Toffoli networks. In *Design, Automation and Test in Europe*, pages 1284–1289, 2009.

[34] R. Wille, H. Zhang, and R. Drechsler. ATPG for reversible circuits using simulation, Boolean satisfiability, and pseudo Boolean optimization. In *IEEE Annual Symposium on VLSI*, 2011.