Synthesis of Reversible Circuits Using Decision Diagrams

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

***Abstract*—Due to its promising applications in domains like quantum computation or low-power design, synthesis of reversible circuits has become an intensely studied topic. However, many synthesis methods are limited by non-scalable function representations like truth tables. As an alternative, synthesis exploiting graph-based representations have been suggested. The underlying structure is a *decision diagram* (DD) that may vary regarding reduction methods, decomposition rules, or ordering restrictions.**

In this work, we review the progress of DD-based synthesis. It is shown that dedicated transformation rules can be applied to generate circuits for functions with a large number of inputs. We discuss the effect of different decomposition types or typical DD improvements like complement edges and re-ordering. Furthermore, we describe how DD-based synthesis can be exploited to transfer theoretical results known from decision diagrams into the domain of reversible circuits. Finally, further directions for future work are outlined.

I. Introduction

Reversible circuits realize n-input n-output functions that map each possible input vector to a unique output vector (i.e. bijections). Although reversible logic significantly differs from conventional (irreversible) logic (e.g. fan-out and feedback are not directly allowed), it has become an intensely studied research area in recent years. In particular, this is caused by the fact that reversible logic is the basis for several emerging technologies. Examples include applications in the domain of

- *Low Power Computation*, where the fact that no information is lost in reversible computation can be exploited (see e.g. [1], [2], [3]),
- *Adiabatic Circuits*, a special low power technology to where reversible circuits are particularly suited for (see e.g. [4]),
- *Encoding and Decoding Devices*, which always realize one-to-one mappings and, thus, inherently follow a reversible computing paradigm (see e.g. [5]),
- *Quantum Computation*, which enables to solve many relevant problems significantly faster than conventional circuits and inherently is reversible (see e.g. [6]), and
- *Program Inversion* (see e.g. [7]), as programs based on a reversible computation paradigm would allow an inherent and obvious program inversion.

Further applications of reversible logic can be found in the domain of optical computing [8], DNA computing [2], and nano-technologies [9].

However, for a long time, synthesis of reversible circuits was limited. Exact (see e.g. [10], [11]) as well as heuristic (see e.g. [12], [13], [14], [15], [16], [17]) methods have been proposed (see e.g. [18] for a more detailed overview). But both are applicable only for relatively small functions. Exact approaches reach their limits with functions containing more than 6 variables [11] while many heuristic methods are able to synthesize functions with at most 30 variables [16]. Moreover, often a significant amount of run-time is needed to achieve these results.

These limitations are caused by the underlying techniques. The existing synthesis approaches often rely on truth tables (or similar descriptions like permutations) of the function to be synthesized (e.g. in [12], [13]). But even if more compact data-structures like DDs [15], positive-polarity Reed-Muller expansion [16], or Reed-Muller spectra [17] are used, the same limitations can be observed since all these approaches apply similar strategies (namely selecting reversible gates so that the choosen function representation becomes the identity).

In order to address this problem, an alternative based on *decision diagrams* (DD) has been introduced in [19]. DD-based synthesis can cope with significantly larger functions. The basic idea is thereby as follows: First, for the function to be synthesized a decision diagram [20] is built. This can efficiently be done for large functions using existing well-developed techniques. Then, each node of the decision diagram is substituted by a cascade of reversible gates. Finally, all cascades are combined forming the desired circuit. By this, a much more efficient data-structure has been exploited allowing for synthesis of reversible circuits for significantly larger functions. Besides that, the concept of DD-based synthesis can be exploited to transfer theoretical results known for decision diagrams into the domain of reversible circuits.

In this work, we review the general ideas and the progress on DD-based synthesis that has been made in the last years. We particularly describe the initial idea and discuss how different decomposition types as well as typical DD improvements like complement edges or re-ordering affect the size of the resulting circuits. Furthermore, how to transfer theoretical results from DDs to reversible circuits is described. Besides that, directions for future work are outlined.

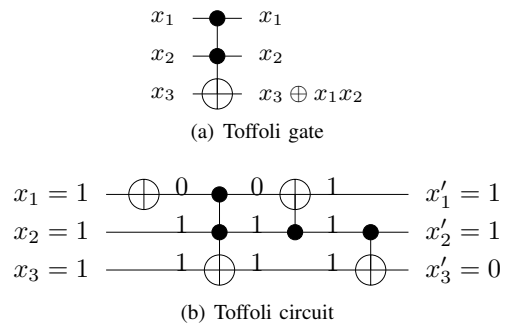
II. Background

A. Reversible Circuits

Reversible circuits are digital circuits with the same number of input signals and output signals. Furthermore, reversible circuits realize bijections only, i.e. each input assignment maps to a unique output assignment. Accordingly, computations can be performed in both directions (from the inputs to the outputs and vice versa).

Reversible circuits are composed as cascades of reversible gates. Each reversible gate over the inputs $X = \{x_1, \ldots, x_n\}$ consists of a (possibly empty) set $C = \{x_{i_1}, \ldots, x_{i_k}\} \subset X$ of *control lines* and a set $T \subset X \setminus C$ of *target lines*. The most commonly used reversible gate is the *Toffoli gate* $TOF(C, x_t)$ [21], which consists of a single target line $x_t \in X \setminus C$ whose value is inverted if all values on the control lines are set to 1 or if $C = \emptyset$, respectively. All remaining values are passed through the gate unaltered.

Example 1: Fig. 1(a) shows a Toffoli gate drawn in standard notation, i.e. control lines are denoted by ●, while the target line is denoted by $\oplus$. A circuit composed of several Toffoli gates is depicted in Fig. 1(b). This circuit maps e.g. the input 111 to the output 110 and vice versa.



(a) Toffoli gate

(b) Toffoli circuit

Fig. 1. Toffoli gate and Toffoli circuit

To measure the cost of a reversible circuit, different metrics are applied (sometimes depending on the addressed technology). In general, the number of circuit lines is an important criterion. In particular in the domain of quantum computation, the number of lines is equal to the number of qubits – a restricted resource.

Beyond that, the costs of the respective gates themselves are important, too. Since simply counting the number of gates does not adequately reflect the effort to realize them, so called *quantum costs* are applied. They measure how many elementary quantum operations are needed in order to realize a reversible gate [6]. In the past, different methods have been introduced that convert a reversible gate into its equivalent quantum operations (see e.g. [22], [23], [24]). Accordingly, different metrics for quantum costs are applied.

B. Binary Decision Diagrams

A Boolean function $f : \mathbb{B}^n \rightarrow \mathbb{B}$ can be represented by a *Decision Diagram* (DD) [20]. A DD is a directed acyclic graph $G = (V, E)$ where e.g. a Shannon decomposition

$$f = \overline{x}_i f_{x_i=0} + x_i f_{x_i=1} \quad (1 \leq i \leq n)$$

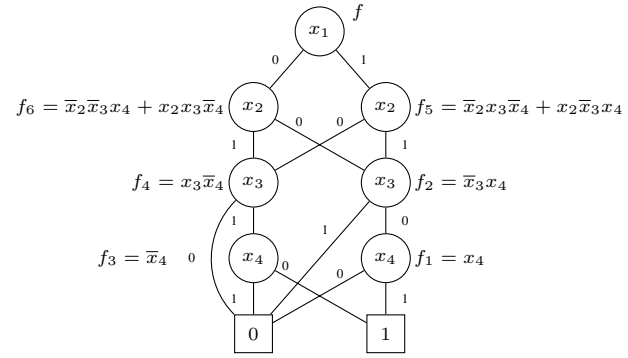
is carried out in each node $v \in V$. The functions $f_{x_i=0}$ and $f_{x_i=1}$ are the *cofactors* of f. In the following, the node representing $f_{x_i=0}$ ($f_{x_i=1}$) is denoted by $low(v)$ ($high(v)$) while x_i is called the *select variable*. A DD is called free if each variable is encountered 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* [20], i.e. nodes v which have more than one predecessor, allow significant reductions. In particular, functions $f : \mathbb{B}^n \rightarrow \mathbb{B}^m$ (i.e. functions with more than one output) can be represented more compactly using shared nodes. Further reductions can be achieved if *complement edges* [25] 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 [20].

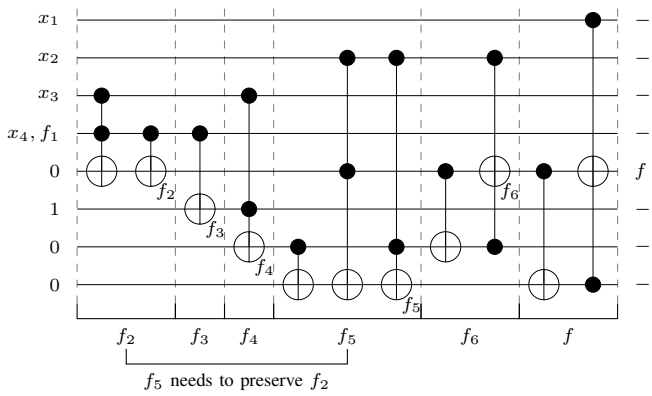
Example 2: Fig. 2(a) shows a DD representing the function $f = \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1x_2x_3\overline{x}_4 + x_1\overline{x}_2x_3\overline{x}_4 + x_1x_2\overline{x}_3x_4$ as well as the respective co-factors resulting from the application of the Shannon decomposition.

III. DD-based Synthesis

In this section, we briefly review DD-based synthesis of reversible circuits as introduced in [19] for *Binary Decision Diagrams* [20]. This provides the basis for the remaining paper, where the application of optimization techniques for decision diagrams to the synthesis approach is discussed.



(a) BDD



(b) Resulting circuit

Fig. 2. Example for BDD-based synthesis

The aim of each synthesis approach is to determine a circuit realization for a given Boolean function. It is well known, that Boolean functions can efficiently be represented by DDs [20]. Having a DD $G = (V, E)$, a reversible circuit can be derived by traversing the DD and substituting each node $v \in V$ with a cascade of reversible gates. The respective cascade of gates depends on the successors of the node v. Table I provides the cascades of Toffoli gates for all possible scenarios of a DD node. Note that this sometimes requires an additional (constant) line.

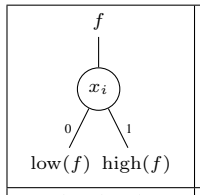
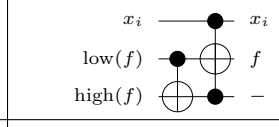
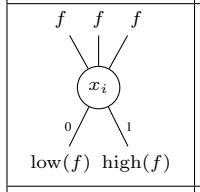
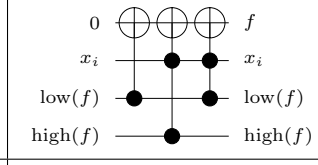
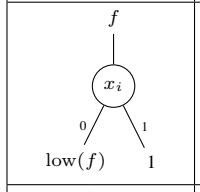
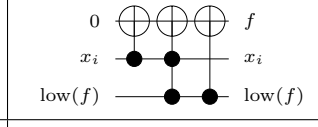
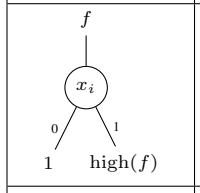
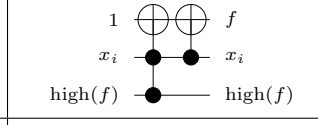
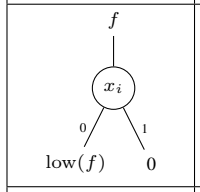
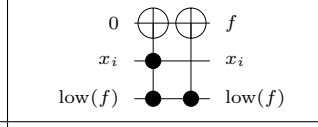
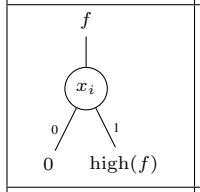
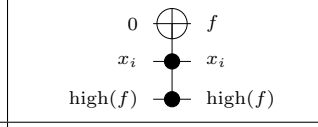
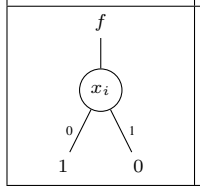
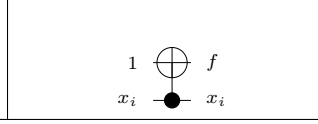
Based on these substitutions, a method for synthesizing a Boolean function as a reversible circuit can be formulated: First, a DD for function f to be synthesized is created. This can efficiently be done using state-of-the-art DD packages (e.g. CUDD [26]). Next, the resulting DD $G = (V, E)$ is traversed by a depth-first search. For each node $v \in V$, cascades as depicted in Table I are added to the circuit. If the entire DD has been traversed, a circuit realizing f has been obtained.

Example 3: A circuit realizing the function represented by the DD as shown in Fig. 2(a) is realized. First, 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. 2(b)[1]. 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. 2(b).

As a result, circuits are synthesized which realize the given function f. Since, each node of the DD is only substituted by a cascade of gates, the proposed method has a linear worst case run-time and memory complexity with respect to the number of nodes in the DD.

[1]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.

TABLE I
MAPPING SHANNON NODES TO REVERSIBLE CASCADES

NODE	CIRCUIT
	
	
	
	
	
	
	

IV. IMPROVEMENTS

For the first time, DD-based synthesis allowed the automatic realization of large reversible functions, i.e. functions with more than 100 inputs. At the same time, optimization of decision diagrams has heavily been considered by researchers in the past (using e.g. new decomposition types, complement edges, or reordering strategies). This builds the basis for further improvements of reversible circuit synthesis through decision diagrams. In this section, some of the improvements, which have already been investigated, are briefly discussed. Afterwards, further research directions are outlined.

A. Consideration of Alternative Decompositions

Besides Shannon, further decompositions of Boolean functions exist. In particular, *positive Davio* and *negative Davio* decomposition defined by

$$f = f_{x_i=0} \oplus x_i \cdot f_{x_i=2} \qquad \text{(pos. Davio)}$$

$$f = f_{x_i=1} \oplus \overline{x}_i \cdot f_{x_i=2} \qquad \text{(neg. Davio)}$$

with $f_{x_i=2} = f_{x_i=0} \oplus f_{x_i=1}$ have been established in the past[2]. Considering positive and negative Davio decomposition has several benefits for reversible circuit design since

- for certain functions they enable decomposing a given function into a smaller number of different sub-functions, i.e. the size of the decision diagram is reduced,
- in many cases they enable more compact realizations as reversible circuits, and
- they enable the preservation of the values of some co-factors without additional circuit lines so that the overall line count can be kept small.

Consequently, smaller circuits may result if these alternative decompositions are applied. This has been investigated in detail in [28] for *Kronecker Functional Decision Diagrams* [29]. To this end, further substitutions for the respective decompositions have been developed. Table II lists the cases which additionally have to be considered.

Example 4: Fig. 3(a) shows a decision diagram representing the function $f = \overline{x}_1\overline{x}_2\overline{x}_3x_4 + \overline{x}_1x_2x_3\overline{x}_4 + x_1\overline{x}_2x_3\overline{x}_4 + x_1x_2\overline{x}_3x_4$, where positive Davio decomposition is applied to each node. Traversing this decision diagram, first the co-factor f_1 is considered. This can be represented by the primary input x_4. Then, the co-factor f_2 can be realized. Continuing this process until the depth first-traversal is completed, a circuit as depicted in Fig. 3(b) results.

As shown by the example, using positive and negative Davio decomposition, a more compact reversible circuit can be realized for the considered function. More precisely, in comparison to the resulting circuit from Fig. 2 the number of lines is reduced by 2, the number of gates by 5, and the quantum cost by 17 for this simple example.

B. Complement Edges

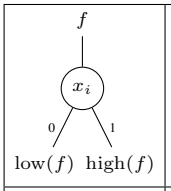
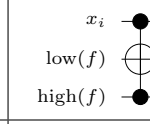
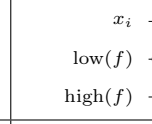
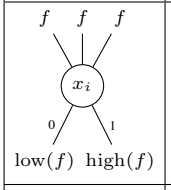
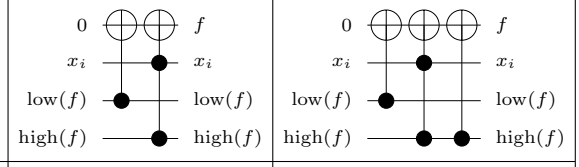
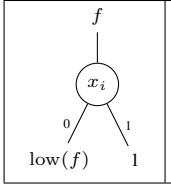
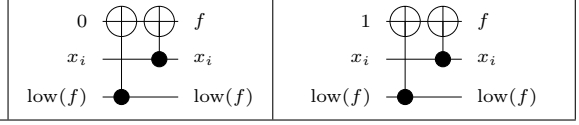
Further reductions in DD sizes can be achieved if *Complement Edges* (CEs) [25] are applied. If a CE is applied, the output value of its connected node becomes inverted. This allows to represent a (sub-)function as well as its negation by a single node only. The effect of CEs to the resulting circuit size has been investigated in [30]. Also here, proper substitutions have been determined which take the inversion by CEs into account. Based on this, it was shown that the consideration of CEs, in fact, may lead to *larger* cascades in comparison to the substitution without CEs. However, the resulting reduction in the size of the decision diagram compensated this drawback in the majority of the cases. This demonstrates that exploiting optimization of decision diagrams does not necessarily improve the resulting reversible circuits.

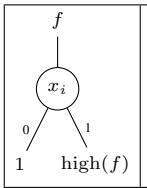
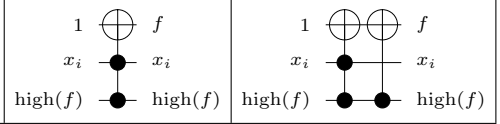
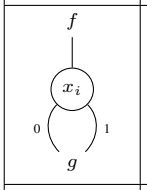
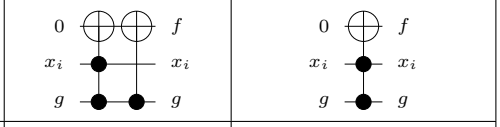
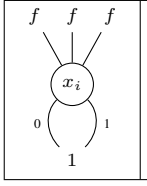
C. Ordering of DDs

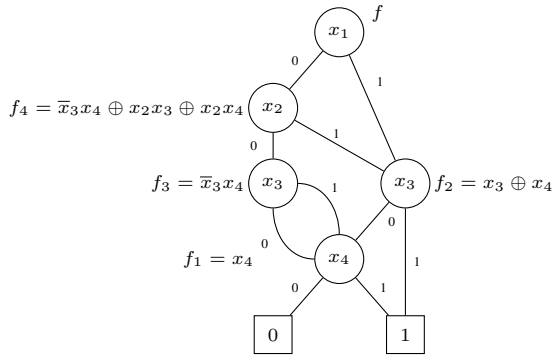
Finally, the exploitation of different ordering strategies of decision diagrams has been considered [30]. It is well known that the order of the variables has a high impact on the size of the resulting DD [20]. Hence, several approaches

[2]In fact, it has been proven that Shannon, positive Davio, and negative Davio decomposition are sufficient to efficiently decompose Boolean functions [27].

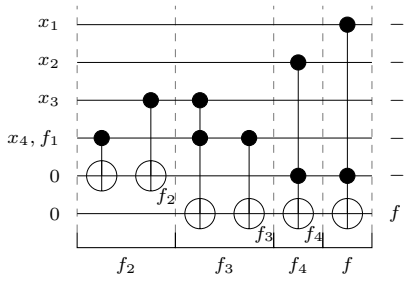
TABLE II
MAPPING DAVIO NODES TO REVERSIBLE CIRCUITS

NODE	CIRCUIT (POS. DAVIO)	CIRCUIT (NEG. DAVIO)
		
		
		

NODE	CIRCUIT (POS. DAVIO)	CIRCUIT (NEG. DAVIO)
		
		
		



(a) KFDD



(b) Resulting circuit

Fig. 3. Example for synthesis exploiting Davio decomposition

have been proposed to achieve good orderings (e.g. through sifting [31] or evolutionary algorithms [32]) or to determine exact results [33] in the past. As observed in [30], the selected ordering, in fact, also has a strong effect on the size of the resulting circuits. Usually, an ordering which improves the DD size also improves the circuit size. However, also here examples have been found showing that optimization for DDs not always leads to smaller circuits.

D. Future Directions

The improvements discussed above only represent "the tip of the iceberg" of possible methods one can apply to improve DD-based synthesis. In fact, DD-optimization is a very well investigated topic. Most of the presented methods have not yet been investigated for exploitation during reversible circuit synthesis. Promising directions to be considered include:

- *Consideration of Entire Sub-trees:* So far, only single nodes of a decision diagram are solely mapped to reversible circuit cascades. However, it is very likely that much better results can be achieved if also frequently occurring sub-trees in the DD are substituted with corresponding cascades.
- *Adjusting the Cost Function of the DD:* So far, all optimizations have been applied using the number of nodes in the decision diagram as cost function. However, it would be much more appropriate if instead the expected quantum costs or the expected number of lines (according to the mappings of Table I and Table II) would be applied. Then, e.g. re-ordering would not try to minimize the number of nodes, but the actually desired result, i.e. the size of the resulting circuit.
- *Extending the Graph Structure:* Due to the restrictions e.g. with respect to ordering and decomposition, many Boolean functions cannot efficiently be represented. Consequently, extended graph structures have been proposed in the past. Examples include e.g. *Read-k-times DDs*, i.e. a generalization of DDs in which variables may occur up to k times on each path [34], or *Mod2OBDDs* [35], where certain XOR nodes are exploited. Both techniques allow for a smaller representation of functions which also can be exploited during reversible circuit synthesis.
- *Reducing the Number of Resulting Circuit Lines:* DD-based synthesis particularly suffers from a large number of additional lines in the resulting circuits (as observed in detail in [36]). An initial approach to reduce this number has been presented in [37]. However, more detailed investigations are still left for future work.
- *Consideration of Alternative DDs:* Alternative DDs, e.g. *Quantum Multiple Valued Decision Diagrams* (QMDDs) [38], provide a more specific data-structure for reversible functions. Hence, developing synthesis based on these DDs seems plausible. First promising results following this direction have been presented in [39] where QMDDs have been exploited to realize large functions with the minimal number of circuit lines.

In order to realize these ideas, the implementation of the existing DD-based synthesis approaches available through *RevKit* [40] can be exploited.

V. Theoretical Consideration

Besides practical issues, also the theoretical background of decision diagrams has intensely been studied in the past (see e.g. [41], [42], [43]). Since the results of DD-based synthesis as applied so far are asymptotically bounded by the number of nodes in the decision diagram, these theoretical results can easily be transferred to the domain of reversible circuits. More precisely, if a function f with n primary inputs and represented by a DD of size k invoking Shannon decomposition only is synthesized, the resulting circuits are composed of *at most*

- $k+n$ circuit lines (since besides the input lines, for each node at most one additional line is added) and
- $3 \cdot k$ gates (since for each node cascades of at most 3 gates are added according to the substitutions of Table I).

Although this already allows some interesting conclusions (e.g. each function can be realized as reversible circuits with at most $3 \cdot 2^n$ gates, symmetric functions can always be realized with a quadratic number of gates, etc.), the underlying upper bound is quite weak. It is very likely that many further results (e.g. tighter upper bounds for general functions as well as for respective function classes) can be derived. Hence, a detailed analysis of the theoretical results that can be obtained by the DD-based synthesis is a promising research direction.

VI. Conclusions

In this work, we reviewed the general idea and the progress on DD-based synthesis that has been made in the last years. Besides that, we outlined further directions that can be addressed in future work. To this end, the implementation of the existing DD-based synthesis approaches available through *RevKit* [40] may provide a good starting point.

References

[1] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, no. 3, pp. 183–191, 1961.

[2] C. H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, pp. 525–532, 1973.

[3] B. Desoete and A. De Vos, "A reversible carry-look-ahead adder using control gates," *Integration*, vol. 33, no. 1–2, pp. 89–104, 2002.

[4] P. Patra and D. Fussell, "On efficient adiabatic design of MOS circuits," in *Workshop on Physics and Computation*, Boston, 1996, pp. 260–269.

[5] R. Wille, R. Drechsler, C. Oswald, and A. Garcia-Ortiz, "Automatic design of low-power encoders using reversible circuit synthesis," in *Design, Automation and Test in Europe*, 2012, pp. 1036–1041.

[6] M. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[7] R. Glück and M. Kawabe, "A method for automatic program inversion based on LR(0) parsing," *Fundamenta Informaticae*, vol. 66, no. 4, pp. 367–395, 2005.

[8] R. Cuykendall and D. R. Andersen, "Reversible optical computing circuits," *Optics Letters*, vol. 12, no. 7, pp. 542–544, 1987.

[9] R. C. Merkle, "Reversible electronic logic using switches," *Nanotechnology*, vol. 4, no. 1, pp. 21–40, 1993.

[10] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis." *IEEE Trans. on CAD*, vol. 25, no. 9, pp. 1652–1663, 2006.

[11] D. Große, R. Wille, G. W. Dueck, and R. Drechsler, "Exact multiple control Toffoli network synthesis with SAT techniques," *IEEE Trans. on CAD*, vol. 28, no. 5, pp. 703–715, 2009.

[12] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Trans. on CAD*, vol. 22, no. 6, pp. 710–722, 2003.

[13] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Design Automation Conference*, 2003, pp. 318–323.

[14] D. Maslov, G. W. Dueck, and D. M. Miller, "Toffoli network synthesis with templates," *IEEE Trans. on CAD*, vol. 24, no. 6, pp. 807–817, 2005.

[15] P. Kerntopf, "A new heuristic algorithm for reversible logic synthesis," in *Design Automation Conference*, 2004, pp. 834–837.

[16] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Trans. on CAD*, vol. 25, no. 11, pp. 2317–2330, 2006.

[17] D. Maslov, G. W. Dueck, and D. M. Miller, "Techniques for the synthesis of reversible Toffoli networks," *ACM Trans. Design Automation Electronic Systems*, vol. 12, no. 4, 2007.

[18] R. Drechsler and R. Wille, "From truth tables to programming languages: Progress in the design of reversible circuits," in *Int'l Symp. on Multiple-Valued Logic*, 2011, pp. 78–85.

[19] R. Wille and R. Drechsler, "BDD-based synthesis of reversible logic for large functions," in *Design Automation Conference*, 2009, pp. 270–275.

[20] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. on Comp.*, vol. 35, no. 8, pp. 677–691, 1986.

[21] T. Toffoli, "Reversible Computing," in *Automata, Languages and Programming*, W. de Bakker and J. van Leeuwen, Eds. Springer, 1980, p. 632.

[22] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, vol. 52, no. 5, pp. 3457–3467, 1995.

[23] D. M. Miller, R. Wille, and Z. Sasanian, "Elementary quantum gate realizations for multiple-control Toffolli gates," in *Int'l Symp. on Multiple-Valued Logic*, 2011, pp. 288–293.

[24] Z. Sasanian, R. Wille, and D. M. Miller, "Realizing reversible circuits using a new class of quantum gates," in *Design Automation Conference*, 2012, pp. 36–41.

[25] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package," in *Design Automation Conference*, 1990, pp. 40–45.

[26] F. Somenzi, *CUDD: CU Decision Diagram Package Release 2.3.1*. University of Colorado at Boulder, 2001.

[27] B. Becker and R. Drechsler, "How many decomposition types do we need?" in *European Design & Test Conf.*, 1995, pp. 438–443.

[28] M. Soeken, R. Wille, and R. Drechsler, "Hierarchical synthesis of reversible circuits using positive and negative davio decomposition," in *Int'l Design and Test Workshop*, 2010, pp. 143–148.

[29] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M. Perkowski, "Efficient representation and manipulation of switching functions based on ordered Kronecker functional decision diagrams," in *Design Automation Conference*, 1994, pp. 415–419.

[30] R. Wille and R. Drechsler, "Effect of BDD optimization on synthesis of reversible and quantum logic," *Electr. Notes Theor. Comput. Sci.*, vol. 253, no. 6, pp. 57–70, 2010.

[31] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," in *Int'l Conf. on Computer-Aided Design*, 1993, pp. 42–47.

[32] N. Göckel, R. Drechsler, and B. Becker, "Learning heuristics for OKFDD minimization by evolutionary algorithms," in *Asia and South Pacific Design Automation Conference*, 1997, pp. 469–472.

[33] R. Drechsler, N. Drechsler, and W. Günther, "Fast exact minimization of BDDs," *IEEE Trans. on CAD*, vol. 19, no. 3, pp. 384–389, 2000.

[34] W. Günther and R. Drechsler, "Implementation of read-k-times BDDs on top of standard BDD packages," in *VLSI Design Conf.*, 2001, pp. 173–178.

[35] C. Meinel and H. Sack, "Mod2OBDDs - a BDD structure for probabilistic verification," *Electronic Notes in Theoretical Computer Science*, vol. 22, 2000.

[36] R. Wille, O. Keszöcze, and R. Drechsler, "Determining the minimal number of lines for large reversible circuits," in *Design, Automation and Test in Europe*, 2011, pp. 1204–1207.

[37] R. Wille, M. Soeken, and R. Drechsler, "Reducing the number of lines in reversible circuits," in *Design Automation Conference*, 2010, pp. 647–652.

[38] D. M. Miller and M. A. Thornton, "QMDD: A decision diagram structure for reversible and quantum circuits," in *Int'l Symp. on Multiple-Valued Logic*, 2006, p. 30.

[39] M. Soeken, R. Wille, C. Hilken, N. Przigoda, and R. Drechsler, "Synthesis of reversible circuits with minimal lines for large functions," in *Asia and South Pacific Design Automation Conference*, 2012, pp. 85–92.

[40] M. Soeken, S. Frehse, R. Wille, and R. Drechsler, "RevKit: An Open Source Toolkit for the Design of Reversible Circuits," in *Reversible Computation 2011*, ser. Lecture Notes in Computer Science, vol. 7165, 2012, pp. 64–76, RevKit is available at www.revkit.org.

[41] H. Liaw and C. Lin, "On the OBDD-representation of general Boolean functions," in *IEEE Trans. on Comp.*, vol. 41, 1992, pp. 661–664.

[42] I. Wegener, *Branching programs and binary decision diagrams: theory and applications*. Society for Industrial and Applied Mathematics, 2000.

[43] R. Drechsler and D. Sieling, "Binary decision diagrams in theory and practice," *Software Tools for Technology Transfer*, vol. 3, pp. 112–136, 2001.