Using πDDs in the Design of Reversible Circuits (Work-In-Progress)

Mathias Soeken[1,3], Robert Wille[1], Shin-ichi Minato[2], and Rolf Drechsler[1,3]

[1] Institute of Computer Science, University of Bremen
Group of Computer Architecture, D-28359 Bremen, Germany
{msoeken,rwille,drechsle}@informatik.uni-bremen.de
[2] Hokkaido University
Sapporo 060-0814, Japan
minato@ist.hokudai.ac.jp
[3] Cyber-Physical Systems, DFKI GmbH
D-28359 Bremen, Germany
rolf.drechsler@dfki.de

Abstract. With πDDs a data structure has recently been introduced that offers a compact representation for sets of permutations. Since reversible functions constitute permutations on the input assignments, they can naturally be expressed using this data structure. However, its potential has not been exploited so far. In this work-in-progress report, we present and discuss possible applications of πDDs within the design of reversible circuits including techniques for synthesis, debugging, and an efficient determination of the number of minimal circuits. We observed that πDDs inhibit the same space complexities as truth tables and, hence, do not superior existing design methods in many cases. However, they are advantageous when dealing with several functions or gates at once.

1 Introduction

Decision diagrams offer a compact representation of Boolean functions and matrices and, thus, have been widely applied in the design of reversible circuits. As examples, *Binary Decision Diagrams* (BDDs) have been applied for exact, heuristic, and hierarchical synthesis of both reversible and irreversible functions [1–3]. As an alternative to BDDs, the application of Kronecker functional decision diagrams, an extension of BDDs, has lead to further improvements [4]. *Quantum Multiple-valued Decision Diagrams* (QMDDs) [5], enabling a compact representation for complex matrices, have been used for both equivalence checking [6] and synthesis of large reversible functions ensuring a minimal number of lines [7]. Similar data-structures have efficiently been applied for the simulation and verification of quantum circuits [8, 9]. In fact, decision diagrams have been the key methodology for breakthroughs in the design of reversible circuits. For the first time, BDDs allowed synthesis of minimal circuits for a significant amount of functions [1] and they enabled the synthesis of large Boolean functions with more than 100 variables [2]. For the latter case, the main problem of the algorithm is the huge amount of extraneous lines which impedes the practical applicability of that approach. However, the problem of additional lines in the synthesis of large functions has been solved again with decision diagrams, in particular using QMDDs [7].

However, while BDDs and QMDDs offer a compact representation for functions and matrices, the recently introduced πDDs [10] allow for a compact representation for permutations. Hence, they are an interesting extension to the set of considered decision diagrams in the design of reversible circuits. Since reversible functions constitute permutations on the input assignments, they can naturally be expressed using this data structure. In fact, πDDs do not only allow a compact representation for single permutations, but for a set of permutations. Therefore, they can particularly be applied for many problems where the above mentioned data structures are not advantageous.

In this work-in-progress report, we briefly review the underlying data-structure and afterwards discuss possible applications in different areas of the design of reversible circuits. These applications include synthesis, debugging, and an efficient determination of the number of minimal circuits. Our observations show that, πDDs inhibit the same space complexities as truth tables (i.e. they are exponential in space) and, hence, in many cases do not superior existing methods. However, advantages can be gained when dealing with several functions or gates at once.

2 Preliminaries

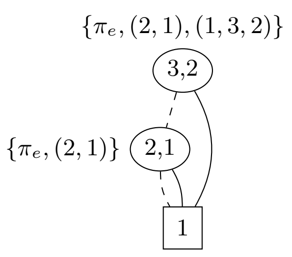
2.1 Reversible Functions and Circuits

A function $f : \mathbb{B}^n \to \mathbb{B}^n$ is called reversible if it represents a bijection, i.e. each input maps uniquely to an output pattern. As a result, reversible functions represent permutations on the set $\{0, \dots, 2^n - 1\}$. Reversible functions can be realized using reversible circuits. The process of determining a reversible circuit for a given function is called *synthesis*. The circuits are usually composed as a cascade of reversible gates where the Toffoli gate [11] constitutes their most prominent representative. Given a set of variables $X = \{x_1, \dots, x_n\}$ a Toffoli gate is a tuple (C, t) with $C \subset \bigcup_{x \in X}\{x, \overline{x}\}$ such that $\forall x \in X : \{x, \overline{x}\} \not\subset C$ being the set of *control lines* and $t \in X$ with $\{t, \overline{t}\} \cap C = \emptyset$ being the *target line* of the gate. A Toffoli gate inverts the target line if, and only if, all control lines x_i $(\overline{x}_i)$ are set to 1 (0). Positive (negative) literals in C are called positive (negative) control lines.

2.2 The πDD Data-Structure

The πDDs [10] allow for a compact representation of sets of permutations and work similar to ZDDs [12] which allow for a compact representation of sets of variables. πDDs exploit that permutations can be decomposed into elementary transpositions swapping two elements, i.e. that a permutation can be seen as a set of its transpositions. As an example the permutation $(3, 5, 2, 1, 4)$ can be represented by a sequence of transpositions $\tau_{(2,1)}\tau_{(3,2)}\tau_{(4,1)}\tau_{(5,4)}$, i.e. first the items 5 and 4 are interchanged, then 4 with 1 and so on until the identity permutation $\pi_e = (1, 2, 3, 4, 5)$ results. When always swapping the elements with the highest absolute value first, sequences of transpositions are canonical.

According to this principle, the vertices in πDDs are labeled using the respective transposition (in comparison, in ZDDs the vertices are labeled using the set element). The terminal vertices 1 and 0 represent the set containing the identity permutation $\{\pi_e\}$ and the empty set $\emptyset$, respectively. As an example, on the left-hand side, a πDD is illustrated representing the permutations $\{\pi_e, (2, 1), (1, 3, 2)\}$. Traversing this πDD from the top to the bottom leads to the transpositions to be applied so that eventually the identity permutation results.



Several operations can be carried out efficiently on πDDs, e.g. counting the number of permutations which is equivalent to counting the number of 1-paths in BDDs or ZDDs. Furthermore, calculating the Cartesian product $P*Q = \{\alpha\beta|\alpha \in P, \beta \in Q\}$ is efficient, which is the set of all possible composite permutations chosen from P and Q. Due to page limitations, the reader is referred to [10] for a comprehensive discussion on πDDs.

3 Applications in the Design of Reversible Circuits

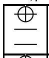
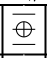
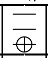
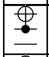
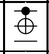
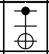
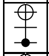
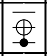
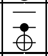
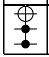
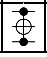
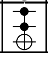
3.1 Determination of the Number of Minimal Circuits

As discussed in the previous section, πDDs allow for a compact representation of sets of permutations as well as efficient operations such as counting the permutations and the Cartesian product. If gates in a circuit are considered as permutations, the latter naturally expresses the gate composition in reversible circuits. Combining both operations allows for an efficient determination of the number of minimal circuits.

This is achieved by creating a set of all elementary gates that may occur in a circuit. Afterwards, the Cartesian product on these gates is iteratively constructed. As a result, all reversible functions are enumerated and contained in the resulting πDD. By extracting the transpositions from the paths in the πDD, it can easily be obtained how many minimal circuits composed of a certain number of gates exist. This πDD represents a set of possible functions but does not explicitly represent the structure of gates for each function. However, we can extract the actual gates by using πDD-based operations, as shown in the next section. Nevertheless, the information on the number of minimal circuits already is interesting for statistical purposes.

Table 1 lists all permutations for positively controlled Toffoli gates acting on 3 lines. For the sake of an improved readability transpositions $\tau_{(x,y)}$ are written as $\binom{x}{y}$. Each permutation is denoted $T_{t,\mu}$ where t is the target line of the gate and μ is an index denoting the set of control lines. Given that, the

Table 1. Permutations for all positively controlled Toffoli gates on 3 lines

μ	$T_{0,\mu}$		$T_{1,\mu}$		$T_{2,\mu}$	
0		$\binom{000}{001}$ $\binom{010}{011}$ $\binom{100}{101}$ $\binom{110}{111}$		$\binom{000}{010}$ $\binom{001}{011}$ $\binom{100}{110}$ $\binom{101}{111}$		$\binom{000}{100}$ $\binom{001}{101}$ $\binom{010}{110}$ $\binom{011}{111}$
1		$\binom{010}{011}$ $\binom{110}{111}$		$\binom{001}{011}$ $\binom{101}{111}$		$\binom{001}{101}$ $\binom{011}{111}$
2		$\binom{100}{101}$ $\binom{110}{111}$		$\binom{100}{110}$ $\binom{101}{111}$		$\binom{010}{110}$ $\binom{011}{111}$
3		$\binom{110}{111}$		$\binom{101}{111}$		$\binom{011}{111}$

set of all positively controlled Toffoli gates acting on n lines can be written as $T_n = \bigcup_{t=0}^{n-1} \bigcup_{\mu=0}^{2^{n-1}-1} T_{t,\mu}$. Since reversible functions can also be represented by permutations, we can count all functions realized by minimal circuits using F_k where k denotes the number of gates with

$$F_0 = \{\pi_e\}, \quad F_1 = F_0 \cup T_n, \quad \text{and} \quad F_k = F_{k-1} * T_n \text{ for } k > 1 \ . \tag{1}$$

The approach can easily be adapted to support other gate libraries such as Toffoli gates containing also negative control lines (denoted by $T_n^{\pm}$) as well as libraries that only consist of Toffoli gates that are fully controlled, i.e. $|C| = n-1$ for each gate (denoted by $\hat{T}_n$ and $\hat{T}_n^{\pm}$). Table 2 shows the number of circuits determined for k ranging from 0 to 12 for all these four gate libraries. In fact, only the number of newly found functions is listed, i.e. $|F_k| - |F_{k-1}|$. As can be seen, all gate libraries except for $\hat{T}_3$ are universal, since the number of all reversible functions over 3 variables is $2^3! = 40320$. In fact, only 24 functions can be represented when using gates exclusively from library $\hat{T}_3$. Furthermore, the numbers for the gate library $\hat{T}_3^{\pm}$ are very interesting as they are more balanced than the other ones. Also, this gate library should be more efficient when used with πDDs since all elementary gates can be represented by permutations that are composed of only one transposition. The largest minimal function consists of 12 gates when using this library and is described by the permutation $(7, 6, 4, 5, 1, 0, 2, 3)$.

Table 2. Size of $|F_k| - |F_{k-1}|$ for four gate libraries

k	T_3	$T_3^{\pm}$	$\hat{T}_3$	$\hat{T}_3^{\pm}$
0	1	1	1	1
1	12	27	3	12
2	102	369	6	90
3	625	2925	9	476
4	2780	13282	5	1903
5	8921	20480	0	5472
6	17049	3236	0	10388
7	10253	0	0	11756
8	577	0	0	7347
9	0	0	0	2408
10	0	0	0	430
11	0	0	0	36
12	0	0	0	1
$\sum$	40320	40320	24	40320

All this information can easily be extracted using the operations supported on πDDs within less than a second for all gate libraries. However, when performing the same experiments for $n = 4$, the approach is not scalable anymore. Note that increasing n by 1 doubles the number of elements in the respective permutations. Further, the number of vertices in the πDD is the number of elements squared, i.e. 2^{2n}.

3.2 Synthesis with Minimal Number of Gates

Based on the results and procedures of the previous section, a synthesis algorithm realizing a function f with a minimal number of gates can be formulated. For this purpose, the function f to be synthesized is represented as permutation π_f. Then, all functions are enumerated as in Eq. (1). After each step k, it is checked whether π_f is contained in all functions F_k. In that case, the minimal number of gates k is already determined. However, the actual circuit has not been obtained yet. For this purpose, the algorithm moves backwards going to F_0 by applying gates from T_n. More precisely, an algorithm for the synthesis of reversible functions ensuring minimal number of gates can be formulated as follows.

Algorithm E (*Exact Synthesis*). The reversible function f to be synthesized is given as permutation π_f.

E1. [Initialize.] Set $F_0 \leftarrow \{\pi_e\}$, $F_1 \leftarrow F_0 \cup T_n$, and $k \leftarrow 1$.

E2. [Found minimum?] If $F_k \cap \{\pi_f\} \neq \emptyset$, i.e. $\pi_f \in F_k$, go to step E4.

E3. [Increase k.] Set $k \leftarrow k+1$, $F_k \leftarrow F_{k-1} * F_1$ and return to step E2.

E4. [Extract gate.] Select $\pi \in T_n$ such that $\pi_f \pi \in F_{k-1}$.

E5. [Next gate?] Set $k \leftarrow k-1$ and $\pi_f \leftarrow \pi_f \pi$. If $k = 1$, terminate, otherwise return to step E4. ▮

This algorithm faces similar problems as discussed in the end of Sect. 3.1, i.e. the complexity raises significantly with increasing size of the function. One possibility to address that is to use a different gate library such as $\hat{T}_n^{\pm}$ which contains smaller permutations. However, as for any other existing exact synthesis approach, the size of all elements contained is still exponential, i.e. $|T_n| = |\hat{T}_n^{\pm}| = n \cdot 2^{n-1}$. This will always cause scalability problems in step E4 in which all gates need to be traversed in the worst case.

3.3 Heuristic Synthesis

Unlike the exact synthesis approach, where all functions are enumerated first in order to check whether the function f to be synthesized is contained, the starting point of the heuristic synthesis approach is the function f itself. Similar to the QMDD-based synthesis procedure [7] gates (or permutations) should be applied according to the structure of the πDD for the function in its current state. However, the πDD-based approach allows for applying several gates at once instead of only one at a time. The result can then be checked for the best current solution and then proceed from there. The goal is to transform the πDD representing π_f by means of gate operations such that eventually the identity function, i.e. π_e is reached. Since the corresponding πDD consists only of one terminal vertex, the aim during synthesis is to constantly reduce the number of non-terminal vertices in the πDD.

3.4 Debugging

As discovered in the previous section, the πDDs for reversible functions grow exponentially with respect to the number of lines. As a result, it is likely that the above mentioned techniques are not applicable to circuits of a larger scale. However, πDDs are advantageous when considering multiple functions at once which should be illustrated in this section. Consider a simple debugging problem to be solved where a faulty circuit should be checked for a *missing-gate* defect. Given a circuit $C = g_1 \dots g_d$ consisting of n lines where each gate g_i can be described by its permutation π_{g_i} and a function f represented by π_f, this debugging problem can be solved using πDDs by checking if $\pi_f \in F$, where

$$F = \bigcup_{i=0}^{d} \left(\{\pi_{g_1} \dots \pi_{g_i}\} * T_n * \{\pi_{g_i+1} \dots \pi_{g_d}\} \right) \quad .$$

All operations can be carried out efficiently on the πDD data structure.

4 Conclusions and Future Work

The πDD for one function can be exponential in size, in fact the permutation for a NOT gate (Toffoli gate without control lines) in a circuit with n lines consists of 2^{n-1} transpositions which is equal to the number of non-terminal vertices in the πDD. As a result, πDDs are not suitable for the synthesis of large functions and thus probably not suitable for synthesis in general. Since determining the minimal number of circuits for 4 circuit lines is already inefficient, it is not to expect that the exact synthesis algorithm based on πDDs can keep up with the results achieved using the exact synthesis approaches based on Boolean satisfiability [13]. However, conceptual algorithms that have been discovered allow an efficient counting and enumerating of reversible functions of small sizes. Interesting statistics comparing different gate libraries have been observed.

Further, the πDDs are advantageous when several functions at once should be considered at the same time, whereas no other graphical data-structure that has been used in the design for reversible circuits so far possesses this property. As a result, the efficient storing of multiple permutations can be exploited. Hence, πDDs should be used in the context of algorithms for reversible functions and circuits that inhibit these properties, e.g. within debugging where πDDs allow to consider several solutions at once.

Algorithms of such kind should be considered in future work. Furthermore, the performance of the πDD implementation should be enhanced such that the determination of the minimal number of circuits can be performed for a larger number of circuit lines. The size of the permutations that are represented through reversible functions grows exponentially as the number of circuit lines grows linearly. This affects the performance of the πDDs in a bad manner, since they allow permutations of all sizes and not only the special cases represented by reversible functions. As a result, also the decomposition technique that serves as the base for current πDDs should be inspected for improvement in the special case of reversible functions, e.g. by explicitly targeting reversible gates as atomic unit instead of transpositions.

Nevertheless, we are convinced that πDDs fit well in the current zoo of graphical data-structures. Although they do not allow an improvement of current techniques they will serve as an efficient data-structure for problems that explicitly require the use of several gate operations or the consideration of several functions in general.

Acknowledgments. This work was supported by the German Research Foundation (DFG) (DR 287/20-1).

References

1. Wille, R., Le, H.M., Dueck, G.W., Große, D.: Quantified Synthesis of Reversible Logic. In: Design, Automation and Test in Europe, pp. 1015–1020. IEEE (March 2008)
2. Wille, R., Drechsler, R.: BDD-based synthesis of reversible logic for large functions. In: Design Automation Conference, pp. 270–275. ACM (July 2009)
3. Kerntopf, P.: A New Heuristic Algorithm for Reversible Logic Synthesis. In: Design Automation Conference, pp. 834–837 (June 2004)
4. Soeken, M., Wille, R., Drechsler, R.: Hierarchical synthesis of reversible circuits using positive and negative Davio decomposition. In: Int'l Design and Test Workshop, pp. 143–148 (December 2010)
5. Miller, D.M., Thornton, M.A.: QMDD: A Decision Diagram Structure for Reversible and Quantum Circuits. In: Int'l Symp. on Multiple-Valued Logic, p. 30. IEEE Computer Society (May 2006)
6. Wille, R., Große, D., Miller, D.M., Drechsler, R.: Equivalence Checking of Reversible Circuits. In: Int'l Symp. on Multiple-Valued Logic, pp. 324–330. IEEE Computer Society (May 2009)
7. Soeken, M., Wille, R., Hilken, C., Przigoda, N., Drechsler, R.: Synthesis of Reversible Circuits with Minimal Lines for Large Functions. In: Asia and South Pacific Design Automation Conference (January 2012)
8. Viamontes, G.F., Markov, I.L., Hayes, J.P.: Quantum Circuit Simulation. Springer, Heidelberg (2009)
9. Wang, S.A., Lu, C.Y., Tsai, I.M., Kuo, S.Y.: An XQDD-based verification method for quantum circuits. IEICE Transactions 91-A(2), 584–594 (2008)
10. Minato, S.-I.: πDD: A New Decision Diagram for Efficient Problem Solving in Permutation Space. In: Sakallah, K.A., Simon, L. (eds.) SAT 2011. LNCS, vol. 6695, pp. 90–104. Springer, Heidelberg (2011)
11. Toffoli, T.: Reversible Computing. In: de Bakker, J.W., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 632–644. Springer, Heidelberg (1980)
12. Minato, S.: Zero-Supressed BDDs for Set Manipulation in Combinational Problems. In: Design Automation Conference, pp. 272–277 (June 1993)
13. Große, D., Wille, R., Dueck, G.W., Drechsler, R.: Exact Multiple-Control Toffoli Network Synthesis With SAT Techniques. IEEE Trans. on CAD 28(5), 703–715 (2009)