Improving Synthesis of Reversible Circuits: Exploiting Redundancies in Paths and Nodes of QMDDs

Alwin Zulehner and Robert Wille

Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
{alwin.zulehner,robert.wille}@jku.at

Abstract. In recent years, reversible circuits have become an established emerging technology through their variety of applications. Since these circuits employ a completely different structure from conventional circuitry, dedicated functional synthesis algorithms have been proposed. Although scalability has been achieved by using approaches based on decision diagrams, the resulting circuits employ a significant complexity measured in terms of quantum cost. In this paper, we aim for a reduction of this complexity. To this end, we review QMDD-based synthesis. Based on that, we propose optimizations that allow for a substantial reduction of the quantum costs by jointly considering paths and nodes in the decision diagram that employ a certain redundancy. In fact, in our experimental evaluation, we observe substantial improvements of up to three orders of magnitudes in terms of runtime and up to six orders of magnitudes (a factor of one million) in terms of quantum cost.

1 Introduction

In the recent years, reversible circuits – circuits that additionally allow to compute the inputs from the outputs – have become an established field in research due to their characteristics regarding low power design (based on the seminal work of Landauer [5] and Bennett [3]) and their application in quantum computations [10] – a new computation paradigm that allows for solving certain tasks substantially faster. Most recently, reversible circuits also found their application in the design of on-chip interconnects [18, 20], encoders [21], and verification [1].

Reversible circuits employ a completely different structure compared to conventional circuits, because not only the outputs are determined by the inputs, but also vice versa. This leads to a structure where circuits consist of a set of circuit lines that are passed through a cascade of reversible gates and, hence, disallow direct feedback and fanout. Consequently, dedicated synthesis approaches that establish the desired unique mapping from inputs to outputs are required. Although a variety of synthesis approaches exist, functional synthesis approaches are of special interest, since they result in a circuit where the number of circuit lines is minimal. Over the years, several such synthesis approaches have been proposed, ranging from exact ones (i.e. the number of gates is minimal) [4] to heuristic ones based on truth tables [12, 6]. However, since their underlying data structure is exponential they suffer from limited scalability. Therefore, methods

based on a more compact representation of the function to be synthesized such as decision diagrams [16, 15] or Boolean satisfiability [13] have been proposed. They, in contrast, yield rather costly circuits.

In this paper, we focus on QMDD-based synthesis [16], one of the scalable synthesis approaches listed above. This synthesis approach is based on *Quantum Multiple-Valued Decision Diagrams* (QMDDs [7, 11]), a decision diagram introduced for the compact representation of reversible and quantum computations. The main premise of QMDD-based synthesis is to employ reversible gates which transform each QMDD node to the identity. By doing that for all nodes of a QMDD (representing the function to be synthesized) in a breadth first fashion, a circuit realizing the given function results. However, during this process often equal cases are considered iteratively; for each, the same cascade of gates are applied – leading to a large number of redundant steps and, hence, gates.

In this work, we introduce a new QMDD-based synthesis approach which aims to consider those redundant cases only once. To this end, a scheme is employed which considers all paths to the currently considered node together and performs logic minimization to reduce their number. Furthermore, nodes which can be transformed to the identity with the same cascade of gates are considered jointly. This allows for a substantial reduction of gates. Experimental evaluations show that the proposed scheme improves the current state of the art QMDD-based synthesis by several orders of magnitudes in terms of runtime as well as in terms of quantum cost (an abstract measure of the complexity of the resulting circuit).

This paper is structured as follows. In Section 2, we briefly review reversible functions and their representation as well as reversible circuits. Based on that, we review QMDD-based synthesis in Section 3 and discuss the proposed optimizations in Section 4. In Section 5, we discuss the improvement that can be achieved by the proposed optimization, whereas Section 6 experimentally confirms these improvements. Section 7 concludes the paper.

2 Background

In this section we briefly recapitulate reversible functions including their efficient representation as well as reversible circuits.

2.1 Reversible Functions

A Boolean function is reversible if the mapping from inputs to outputs establishes a bijection, i.e. the inputs can also be computed from the outputs.

Definition 1. *A Boolean function $f : \mathbb{B}^m \to \mathbb{B}^n$ is reversible iff $n = m$ and there exists a unique mapping from inputs to outputs and vice versa.*

Besides truth tables, reversible functions can also be represented by means of a permutation matrix.

Definition 2. *Consider a reversible Boolean function $f : \mathbb{B}^n \to \mathbb{B}^n$. Then, the permutation matrix of f is a $2^n \times 2^n$ matrix with entries $m_{i,j}$, $0 \leq i, j < 2^n$ such that*

$$m_{i,j} = \begin{cases} 1 & \text{if } f(j) = i \\ 0 & \text{otherwise.} \end{cases}$$

A 1-entry in the permutation matrix means that an input (column) is mapped to an output (row) by f. All other entries of the permutation matrix are zero. Since a permutation represents a unique mapping (i.e. a reversible function), it contains exactly one 1-entry in each column and in each row.

Example 1. Consider the reversible function depicted in Fig. 1a. The function is reversible because the number of inputs is equal to the number of outputs and each output pattern occurs exactly once. Consequently, the input can be uniquely determined having the output only. The permutation matrix of this reversible function is shown in Fig. 1b. Here, reversibility can easily be seen since each column as well as each row contains exactly one 1-entry.

x_1	x_2	x_3	x'_1	x'_2	x'_3
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	1	1

(a) Truth table

		Inputs							
	$x_1 x_2 x_3$	000	001	010	011	100	101	110	111
Outputs	000	0	0	0	0	0	0	**1**	0
	001	0	0	0	**1**	0	0	0	0
	010	**1**	0	0	0	0	0	0	0
	011	0	**1**	0	0	0	0	0	0
	100	0	0	**1**	0	0	0	0	0
	101	0	0	0	0	0	**1**	0	0
	110	0	0	0	0	**1**	0	0	0
	111	0	0	0	0	0	0	0	**1**

(b) Permutation matrix

Fig. 1. Representation of reversible functions

2.2 Quantum Multiple-Valued Decision Diagrams (QMDDs)

The description means for reversible functions discussed in the previous section is rather limited. Since the size of truth tables as well as the size of permutation matrices grows exponentially with the number of variables, a more scalable representation is desirable in order to represent large functions. However, permutation matrices can be represented more compactly – and with non-exponential space in many relevant cases – using so called *Quantum Multiple-Valued Decision Diagrams* (QMDDs [7, 11]). QMDDs were initially introduced to represent the

unitary matrices of quantum computations. Since quantum computations are reversible and the matrices are also of dimension $2^n \times 2^n$, QMDDs are perfectly suited for representing permutation matrices. For simplicity, we discuss only those aspects of QMDDs that are relevant for this paper and omit all quantum related issues.

The main idea of QMDDs is a recursive decomposition of a permutation matrix M over its variables. A variable x_i represents a mapping from the i^{th} input to the i^{th} output of the function. There exist four possible mappings of a variable, since an input may be mapped from 0 to 0, from 0 to 1, from 1 to 0, or from 1 to 1. Considering the most significant variable x_1 of the permutation matrix, these four different mappings exactly describe the four quadrants of the matrix, which we denote $M_{0\to0}$, $M_{1\to0}$, $M_{0\to1}$, and $M_{1\to1}$. This decomposition can be represented by a decision diagram node labeled x_1 with four outgoing edges, describing exactly those quadrants $M_{0\to0}$, $M_{1\to0}$, $M_{0\to1}$, and $M_{1\to1}$ from left to right.

This decomposition is recursively applied until a single entry is reached. Such an entry is then represented by a terminal. The compactness of QMDDs results – as for other types of decision diagrams – from sharing equal nodes (and thus representing equal sub-matrices). For simpler graphical visualization, we represent 0-matrices (i.e. matrices containing 0-entries only) with a 0-stub, independent of their dimension.

Example 2. Consider again the permutation matrix depicted in Fig. 1b. The decomposition over its variables x_1, x_2, and x_3 yields the QMDD shown in Fig. 2. Note that the path highlighted in bold traverses the third edge of the node labeled x_1, the second edge of the node labeled x_2 and the first edge of the node labeled x_3. Consequently, this path describes a mapping of variable x_1 from 0 to 1, a mapping of variable x_2 from 1 to 0, and a mapping of variable x_3 from 0 to 0, i.e. a mapping from input $x_1x_2x_3 = 010$ to output $x_1x_2x_3 = 100$.

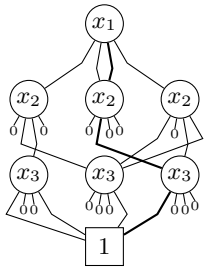


Fig. 2. QMDD of the permutation matrix shown in Fig. 1b

For a formal definition of QMDDs, as well as manipulation algorithms such as matrix multiplication we refer to [11].

2.3 Reversible Circuits

The structure of reversible circuits is completely different to classical circuitry, because fanout and feedback are not directly allowed. A reversible circuit rather consists of a set of circuit lines (one for each variable of the realized reversible function) which are passed through a cascade of reversible gates. The values of the circuit lines may be changed in a reversible fashion by these gate or passed through unaltered. In this paper, we focus on Toffoli gates – a reversible gate that has been proven to be universal (i.e. all reversible functions can be realized using Toffoli gates only).

Definition 3. *Consider a set* $X = \{x_1, x_2, \ldots, x_n\}$ *of* n *circuit lines and a sequence* $G = g_1, g_2, \ldots, g_k$ *of* k *reversible gates. Then, the pair* $C = (X, G)$ *describes a reversible circuit. A Toffoli gate* $g_i = TOF(C_i, t_i)$ *consists of a set* $C_i \subseteq \{x_i^- | x_i \in X\} \cup \{x_i^+ | x_i \in X\}$ *of* negative *and* positive control lines*, and a* target line t_i*. The Boolean value of the target line is inverted iff the Boolean value of all positive and negative control lines is 1 and 0, respectively. All circuit lines except the target line are passed unaltered through the gate.*

Toffoli gates as defined above are self-inverse, i.e. applying a Toffoli gate twice results in the identity. For graphical visualization we use the symbols ●, ○, and ⊕ to represent a positive control line, a negative control line, and a target line, respectively.

Example 3. Consider the reversible circuit shown in Fig. 3. The circuit is composed of three Toffoli gates and is labeled with the intermediate values resulting when the input lines are assigned $x_1x_2x_3 = 010$. The first gate $g_1 = TOF(\{x_1^-, x_2^+\}, x_3)$ inverts the value of target line x_3, because the negative control line x_1 is assigned 0 and the positive control line x_2 is assigned 1. Similarly, the second gate $g_2 = TOF(\{x_3^+\}, x_1)$ is activated, because the value of circuit line x_3 is now 1. Consequently, the value of target line x_1 is changed to 1. The last gate $g_3 = TOF(\{x_2^+, x_3^-\}, x_1)$ does not change the value of target line x_1, because the negative control line x_3 is assigned 1. Therefore, all three circuit lines are passed through the gate unaltered in this case. Eventually, the circuit maps input $x_1x_2x_3 = 010$ to output $x_1'x_2'x_3' = 111$.

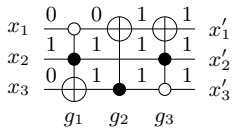


Fig. 3. Reversible circuit

The complexity of reversible circuits is usually measured in terms of quantum cost. These cost result from mapping the circuit to specific libraries of quantum gates. Two commonly used libraries for determining the quantum cost of a reversible circuit are the *NCV* library [8] and the *Clifford+T* library [2]. Here, the quantum cost are determined by the overall number of gates (*NCV-cost*) or the length of the sequence of *T-gates* (and, therefore, denoted *T-depth*), respectively. As shown in Table 1, the *NCV-cost* as well as the *T-depth* of a Toffoli gate depends on the number of control lines. The quantum costs listed in Table 1 were determined using RevKit [14].

Table 1. Quantum cost of Toffoli gates

Control lines	*NCV-cost*	*T-depth*
1	1	0
2	5	3
3	20	12
4	40	24
5	60	36

Example 3 (continued). The *NCV-cost* and the *T-depth* of the circuit shown in Fig. 3 are 11 and 6, respectively.

3 QMDD-based Synthesis

In this section, we review QMDD-based synthesis (originally proposed in [16]). As discussed in Section 1, QMDD-based synthesis is a functional synthesis approach which yields a circuit with the minimal number of circuit lines. The main idea behind the algorithm is described as follows.

Assume the function to be synthesized is described by a permutation matrix M. Then, due to reversibility its inverse M^{-1} exists, and their product $M \circ M^{-1} = I$ is the identity matrix. Consequently, if we find a cascade of reversible gates G that transforms M to the identity, we implicitly found a reversible circuit for M^{-1}. Reversing G yields a reversible circuit that realizes M (because the Toffoli gates are self-inverse). Therefore, an algorithm is required that transforms the QMDD representing M to the identity. Since the identity matrix only maps input 0 to output 0 and input 1 to output 1, the identity QMDD imposes the structure depicted in Fig. 4.

To obtain the desired identity QMDD, we traverse the QMDD in breadth-first manner and transform each node we encounter to the identity structure shown in Fig. 5 by applying Toffoli gates. To this end, the paths to the 1-terminal (called 1-paths in the following) through the second and third edge of the currently considered node have to be moved to the first and fourth edge, respectively. These 1-paths refer to the input of the encountered variables and therefore contain a

Fig. 4. Identity QMDD with n variables

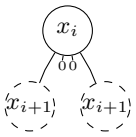


Fig. 5. Identity structure

negative literal $\overline{x}_j$ (positive literal x_j) whenever the first or third (second or fourth) edge of a node labeled x_j is traversed.

In the following, we denote the sets of 1-paths through the first, second, third and fourth edge of the currently considered node by P_1, P_2, P_3, and P_4, respectively. Similarly, we refer to the sets of 0-paths (i.e. paths that terminate in a 0-stub) with $\overline{P}_1$, $\overline{P}_2$, $\overline{P}_3$, and $\overline{P}_4$. Since the QMDD represents a permutation matrix, each column and each row of the matrix contains exactly one 1-entry. Therefore, the number of 0-paths in $\overline{P}_1$ ($\overline{P}_4$) is equal to the number of 1-paths in P_2 (P_3), i.e. $\left|\overline{P}_1\right| = |P_2|$. Moreover, $\overline{P}_1 = P_3$ and $\overline{P}_4 = P_2$.

Example 4. Consider the QMDD shown in Fig. 2 and assume that the root node is currently considered. The sets of 1-paths are $P_1 = P_4 = \{\overline{x}_2\overline{x}_3, \overline{x}_2x_3, x_2x_3\}$ and $P_2 = P_3 = \{x_2\overline{x}_3\}$.

To obtain the identity structure for the currently considered node labeled x_i, we swap the 1-paths in P_2 with the 0-paths in $\overline{P}_1$. This inherently swaps the 1-paths in P_3 with the 0-paths in $\overline{P}_4$. Swapping paths can be accomplished by applying Toffoli gates, since applying a Toffoli gate $TOF(C, x_i)$ inverts the input of variable x_i for all paths that are represented by C.

Example 4 (continued). The path $p = x_2\overline{x}_3$ is contained in the set P_2 of 1-paths through the second edge as well as in the set $\overline{P}_1$ of 0-paths through the first edge. These two paths can be swapped by applying the Toffoli gate $TOF(\{x_2^+, x_3^-\}, x_1)$. This automatically swaps the 1-path in P_3 with the 0-path in $\overline{P}_4$. The resulting QMDD is depicted in Fig. 6.

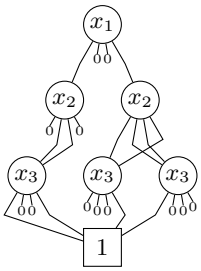


Fig. 6. QMDD resulting from transformation of the root node of the QMDD in Fig. 2

For a more formal description of how the currently considered node can be transformed to the desired identity structure we refer to [16], because understanding the main idea of QMDD-based synthesis (the breadth-first traversal of the nodes) is sufficient for the purpose of this paper.

If the currently considered node is not the root node of the QMDD, we have to ensure that no other node is affected by the applied gates. To this end, we add further control lines that describe the path to the currently considered node to each Toffoli gate that is applied. More precisely, if the path to the currently considered node traverses the first edge of another node (representing a mapping from 0 to 0), we add a negative control line for the corresponding variable. Analogously, we add a positive control line for the corresponding variable if the path traverses the fourth edge of that node[1]. If there exist k such paths to the currently considered node, we have to replicate each Toffoli gate for each of those k paths in order to eventually transform the currently considered node to the identity structure.

Example 5. Consider the QMDD depicted in Fig. 7 and assume that the node highlighted in blue is currently considered. This node can be transformed to the desired identity structure by applying a Toffoli gate with target line x_3. Since there exist two paths to this node, namely $\overline{x}_1 x_2$ and $x_1 \overline{x}_2$, we have to apply two gates $TOF(\{x_1^- x_2^+\}, x_3)$ and $TOF(\{x_1^+ x_2^-\}, x_3)$ to eventually transform this node to the desired identity structure. The resulting circuit is shown in Fig. 8a.

4 Improving QMDD-based Synthesis

In the synthesis scheme originally proposed in [16] and reviewed above, the overall number of paths to the nodes of a certain variable grows exponentially with the number of variables above. This leads to a substantial number of gates with (partially) redundant sets of control lines. This poses a significant drawback to QMDD-based synthesis since

[1] A path to the currently considered node can only traverse the first or the fourth edge of other nodes, because they already establish the identity structure.

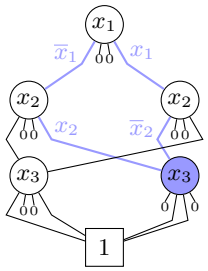


Fig. 7. Paths to the currently considered node

(a) Without optimization (b) ESoP-minimized

Fig. 8. Gates required to transform the currently considered node from Fig. 7

- a significant number of gates is applied to transform each single node of the considered QMDD to the identity structure and
- the applied gates usually include rather large sets of control lines.

More precisely, the number of gates is heavily influenced by the number of paths to the currently considered node, because each gate that is required to transform the currently considered node to the identity has to be replicated for each path. Furthermore, the number of control lines of these Toffoli gates depend on the number of literals of the path from the root node to the currently considered node (because these literals are added to the Toffoli gates in form of control lines to ensure that no other node is affected). Since the overall costs of a reversible circuit depend on both, the total number of gates as well as the respective number of control lines, this makes circuits generated using QMDD-based synthesis rather expensive.

In order to address the problem, we propose two optimization techniques to reduce the cost of the circuits generated by QMDD-based synthesis, namely

1. a straightforward solution which performs logic minimization on the paths to the currently considered node to reduce the number of paths as well as the number of their literals and
2. a more elaborate approach which considers nodes that require the same sequence of Toffoli gates in order to get transformed to the identity structure jointly.

The straight forward solution utilizes logic minimization techniques to reduce the overall number of paths to the currently considered node as well as to reduce the overall number of literals in the paths. To this end, each path to the currently considered node is described as a product (conjunction) of its literals. Then, the exclusive sum (exclusive disjunction) of all these products is formed. The sum has to be exclusive, because applying a Toffoli gate an even times does not have any effect (since a Toffoli gate is self-inverse). The resulting *Exclusive Sum of Products* (ESoP) can be minimized using techniques such as proposed in [9]. Such a minimization reduces the overall number of products (and, hence, the number of paths and gate replications) as well as the number of literals in these products (and, hence, the number of control lines that have to be added to each gate).

Example 6. Consider again the QMDD depicted in Fig. 7 and assume that the node highlighted in blue is currently considered. There exist two paths to the currently considered node, namely $x_1\overline{x}_2$ and $\overline{x}_1x_2$. The ESoP of the two paths is then $x_1\overline{x}_2 \oplus \overline{x}_1x_2$. Minimizing this ESoP yields $x_1\overline{x}_2 \oplus \overline{x}_1x_2 = x_1 \oplus x_2$. Consequently, also the two paths x_1 and x_2 can be used to describe all paths to the currently considered node. The resulting gates are shown in Fig. 8b. Although the number of paths (and therefore the number of gates) did not change, the *NCV-cost* and *T-depth* are reduced from 10 to 2 and from 6 to 0, respectively.

The more elaborated optimization approach aims to further reduce the cost of the circuits considering more than one node simultaneously. The general idea is based on the key observation that sometimes different QMDD nodes require the same sequence of Toffoli gates in order to get transformed to the identity. As described in Section 3, this sequence of Toffoli gates depends on the set P_1 of 1-paths through the first edge and the set P_2 of 1-paths through the second edge only. From these two sets, the other sets of 1-paths ($P_3 = \overline{P}_1$, $P_4 = \overline{P}_2$) as well as the set of 0-paths can uniquely be determined. Cases frequently occur where nodes in the QMDD have equal sets of 1-paths P_1 and P_2, even though they are structurally different.

Example 7. Consider the QMDD depicted in Fig. 9. The root node already establishes the desired identity structure and the two nodes labeled x_2 (highlighted in blue) are structurally different. However, their sets of 1-paths are equal, i.e. $P_1 = \{\overline{x}_3\}$ and $P_2 = \{\overline{x}_3\}$ for both nodes. Consequently, both nodes can be transformed to the identity structure with the same sequence of Toffoli gates. One possible sequence is $TOF(\{x_2^+\}, x_3), TOF(\{x_3^+\}, x_2)$.

Without applying this scheme, the sequence of Toffoli gates has to be replicated twice – once for each node (including control line x_1^- for the left node and control line x_1^+ for the right node). This resulting in the circuit shown in Fig. 10a. The gates g_1 and g_2 thereby transform the left node to the identity, whereas the gates g_3 and g_4 transform the right node to the identity.

Since QMDD nodes that employ an equal characteristic regarding their sets of 1-paths P_1 and P_2 can be transformed to the identity structure with the same

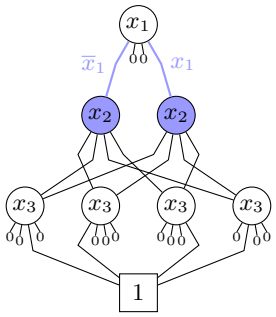


Fig. 9. QMDD nodes with equal sets P_1 and P_2

(a) Without joint consideration (b) With joint consideration

Fig. 10. Gates required to transform the nodes labeled x_2

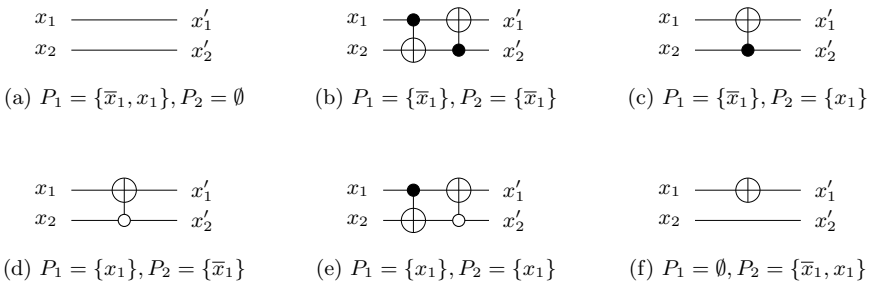
sequence of Toffoli gates, they can considered jointly for synthesis purposes and thus processed together. To this end, we form the ESoP of the paths to all nodes with equal sets P_1 and P_2 and apply the logic minimization as described in the straight forward approach.

Example 7 (continued). Since both nodes labeled x_2 have equal sets of 1-paths P_1 and P_2, they can be considered jointly and processed together. The minimized ESoP of all paths to these nodes is $x_1 \oplus \overline{x}_1 = 1$, a sum consisting of a single product without any literals. Therefore no additional control lines are required (all nodes labeled x_2 can be considered jointly). The resulting circuit that transforms all nodes labeled x_2 to the identity structure is shown in Fig. 10b. Compared to the gate sequence depicted in Fig. 10a we observe a reduction of the *NCV-cost* from 20 to 2 and a reduction of the *T-depth* from 12 to 0.

5 Discussion

In this section, we briefly analyze the potential of the optimization scheme described above, i.e. we discuss how many nodes might be considered together in the best case. To this end, we assume that the nodes of n variables are left to be processed, i.e. the currently considered nodes are sub-QMDDs with height n, and that the sequence of Toffoli gates that transforms these nodes to the identity structure is uniquely determined by the set of 1-paths P_1 and P_2.

First, we determine how many different sequences of Toffoli gates exist. Since we assume that the sequence of Toffoli gates is uniquely determined by P_1 and P_2, we analyze how many combinations of sets P_1 and P_2 exist: The QMDD of the currently considered node represents a $2^n \times 2^n$ permutation matrix (since all nodes above already employ the identity structure). Consequently, there must be exactly one 1-entry in each of the 2^n rows and in each of the 2^n columns. This means, that there must be exactly 2^{n-1} 1-entries in the upper half of the matrix, i.e. $|P_1| + |P_2| = 2^{n-1}$. These 2^{n-1} 1-entries (1-paths) are arbitrarily distributed in the 2^n columns (in the sets P_1 and P_2). Consequently, there exist $\binom{2^n}{2^{n-1}}$ possibilities in which rows the 1-entries are located, i.e. possible different pairs of sets (P_1, P_2). If we assume $n = 2$, there exist $\binom{2^2}{2^{2-1}} = 6$ different sequences that transform a currently considered node to the identity. These sequences as well as their corresponding sets P_1 and P_2 are depicted in Fig. 11.



(a) $P_1 = \{\overline{x}_1, x_1\}, P_2 = \emptyset$ (b) $P_1 = \{\overline{x}_1\}, P_2 = \{\overline{x}_1\}$ (c) $P_1 = \{\overline{x}_1\}, P_2 = \{x_1\}$

(d) $P_1 = \{x_1\}, P_2 = \{\overline{x}_1\}$ (e) $P_1 = \{x_1\}, P_2 = \{x_1\}$ (f) $P_1 = \emptyset, P_2 = \{\overline{x}_1, x_1\}$

Fig. 11. Sequences of Toffoli gates for $n = 2$

As a second step, we analyze how many different sub-QMDDs with n variables exist. Recall, that a QMDD composed of n variables represents a permutation matrix of dimension $2^n \times 2^n$, i.e. a matrix that represents a permutation of 2^n elements. Since 2^n elements can be permuted in $2^n!$ ways, there exist $2^n!$ structurally different QMDDs with n variables. Considering again that $n = 2$, there exist $2^2! = 4! = 24$ structurally different QMDDs.

Having an arithmetic expression for the number of sequences as well as for the number of QMDDs allows one to analyze the potential of the proposed optimization. The resulting numbers for several values of n are provided in Table 2. As one can easily see, there are many more different QMDDs than sequences, because $\binom{2^n}{2^{n-1}} \ll 2^n!$. Consider the case that the $n = 3$ variables of the QMDD are not yet processed. In the worst case, the QMDD has 40 320 nodes labeled with the currently considered variable. For each of those nodes, a sequence of Toffoli gates has to be determined. If we apply the proposed optimization (i.e. if we jointly consider nodes with equal sets of 1-paths), the number of nodes that have to be processed drops to 70 – reducing the computational effort by a factor of 576. Furthermore, the logic minimization used to reduce the paths to the cur-

rently considered nodes is applied to a larger set of paths, which makes it more likely to obtain a more compact ESoP.

Table 2. Potential of the proposed optimization

n	No. sequences $\binom{2^n}{2^{n-1}}$	No. QMDDs $2^n!$
2	6	24
3	70	40 320
4	12 870	$2 \cdot 10^{13}$
5	$6 \cdot 10^8$	$2.6 \cdot 10^{35}$

Obviously, it is more likely that many nodes can be processed together if their currently considered variable is the label for a large number of nodes. Therefore, this optimization has a higher impact on large QMDDs than on small ones. Since we observed that large QMDDs tend to yield circuits with rather high quantum cost, we expect higher improvements for these cases.

6 Experimental Results

In this section, we evaluate the reduction of the quantum cost we achieve by applying the proposed optimizations to the QMDD-based synthesis algorithm. To this end, we have reimplemented the QMDD-based synthesis as originally proposed in [16] (including some minor optimizations regarding performance) in C++ using the QMDD package provided in [11] and the BDD package CUDD [17]. This implementation represents the current state-of-the-art and serves as baseline. Based on that, we have implemented the optimizations discussed in Section 4. In the following, Scheme A denotes the optimization where the paths to the currently node are reduced using logic minimization[2] and Scheme B denotes the case if we additionally process nodes with equal sets of 1-paths jointly. As benchmarks served the reversible circuits provided at RevLib [19]. All experiments were conducted on a 4 GHz processor with 32 GB of memory running Linux 4.4.

Table 3 summarizes the experimental results. The first two columns list the name of the benchmark and the number of circuit lines n. Then, for each synthesis scheme, the runtime, the *NCV-cost* as well as the *T-depth* is listed. Finally, we list the reduction of the quantum cost for Scheme A with respect to the baseline and for Scheme B with respect to Scheme A. Since the improvement rates observed for *NCV-cost* and for *T-depth* are almost identical for each of the benchmarks (they deviate in a fraction of a percent only), we only list the improvement regarding *T-depth* in the last two columns of Table 3.

The obtained results clearly show a significant improvement in terms of runtime. While the original approach requires a significant amount of runtime for

[2] We utilized the methods available at RevKit [14] for logic minimization.

some benchmarks (e.g. more than 1000 seconds for benchmarks *sym9*, *rd84*, and *cycle10*), the optimizations proposed in this paper allowed to synthesize all benchmarks within a few seconds (Schemes A and B). Only one benchmark (*cordic*) required slightly more than a minute.

Furthermore, a substantial improvement in terms of quantum cost can be observed for the benchmarks. For all benchmarks that result in a circuit with a *T-depth* of more than half a million using the original approach, substantial improvements of several orders of magnitudes were determined. Consider for example the benchmarks *plus127mod8192* and *plus63mod8192*. Performing logic optimizations on the paths to the currently considered node (i.e. Scheme A) already result in a reduction of the quantum cost by a factor of 145.21. If we additionally transform nodes together that have equal sets of 1-paths (i.e. Scheme B), we get another improvement by a factor of 8029.35 and, hence, an overall improvement of six orders of magnitudes. On average, we observe an improvement by a factor of 4.22 for Scheme A with respect to the original approach and an improvement by a factor of 5.35 of Scheme B with respect to Scheme A. This results in an overall improvement by a factor of 22.57.

7 Conclusion

In this paper, we reviewed the QMDD-based synthesis algorithm (proposed in [16]) for reversible circuits. Based on that review, we discovered cases that result in the same sequence of Toffoli gates, but are considered iteratively – leading to a substantial overhead in the number of gates. To reduce the costs of the resulting circuits, we proposed optimizations that consider such redundant cases jointly during synthesis. Experimental evaluations show that substantial improvements to the current state-of-the-art can be achieved. More precisely, improvements of up to three orders of magnitudes were observed for the runtime and improvements up to six orders of magnitudes were observed regarding the quantum cost of the resulting circuits.

Acknowledgements

This work has partially been supported by the European Union through the COST Action IC1405.

References

1. L. G. Amarù, P. Gaillardon, R. Wille, and G. D. Micheli. Exploiting inherent characteristics of reversible circuits for faster combinational equivalence checking. In *Design, Automation and Test in Europe*, pages 175–180, 2016.
2. M. Amy, D. Maslov, M. Mosca, and M. Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 32(6):818–830, 2013.

Table 3. T-depth improvements compared to the state-of-the-art

		Baseline (original QMDD-based synth.)			Scheme *A*			Scheme *B*			Improvement	
Benchmark	n	t	NCV	T-depth	t	NCV	T-depth	t	NCV	T-depth	Base/A	A/B
alu1	20	0.02	319	180	0.03	319	180	0.04	319	180	1.00	1.00
cmb	20	0.00	1 462	864	0.01	1 462	864	0.00	1 462	864	1.00	1.00
cycle17_3	20	0.00	50 270	30 105	0.01	50 270	30 105	0.01	46 513	27 852	1.00	1.08
ex1010	20	1.91	139 558	83 724	1.76	140 282	84 156	1.64	136 098	81 648	0.99	1.03
C7552	21	0.01	963	576	0.00	963	576	0.01	963	576	1.00	1.00
decod	21	0.00	963	576	0.00	963	576	0.01	963	576	1.00	1.00
dk17	21	0.01	2 308	1 371	0.00	2 308	1 371	0.01	2 308	1 371	1.00	1.00
ham7	21	0.06	2 528 008	1 516 800	0.03	165 254	99 147	0.03	1 169	696	15.30	142.45
pcler8	21	0.00	509	300	0.00	509	300	0.00	509	300	1.00	1.00
alu4	22	6.66	193 546	116 109	1.89	204 945	122 946	1.95	196 765	118 038	0.94	1.04
apla	22	0.02	4 442	2 652	0.02	4 442	2 652	0.03	4 442	2 652	1.00	1.00
cm150a	22	0.19	1 302	768	0.19	1 302	768	0.19	1 302	768	1.00	1.00
f51m	22	1.48	61 066	36 630	1.46	61 066	36 630	1.49	61 066	36 630	1.00	1.00
mux	22	0.19	1 284	768	0.19	1 284	768	0.18	1 284	768	1.00	1.00
tial	22	1.26	107 806	64 665	1.21	107 806	64 665	1.30	98 646	59 169	1.00	1.09
plus63mod4096	23	5.71	174 926 090	104 955 648	0.59	1 822 510	1 093 500	0.32	465	276	95.98	3961.96
add8	25	7.92	753 638	452 175	3.90	252 538	151 512	4.00	252 538	151 512	2.98	1.00
cordic	25	76.91	747 026	448 080	72.43	747 026	448 080	73.42	747 026	448 080	1.00	1.00
cu	25	0.01	1 535	915	0.02	1 535	915	0.01	1 535	915	1.00	1.00
plus127mod8192	25	21.83	588 787 850	353 272 704	1.78	4 054 830	2 432 892	1.24	510	303	145.21	8029.35
plus63mod8192	25	21.91	588 787 850	353 272 704	1.85	4 054 830	2 432 892	1.23	510	303	145.21	8029.35
rd73	25	39.61	1 047 169 928	628 301 952	0.65	1 184 798	710 868	0.12	13 899	8 334	883.85	85.30
in0	26	0.44	25 117	15 057	0.46	25 117	15 057	0.44	25 117	15 057	1.00	1.00
sym9	27	1572.09	42 859 993 608	25 715 996 160	2.75	6 353 079	3 811 839	1.13	531 033	318 615	6 746.35	11.96
apex4	28	2.25	97 777	58 641	2.12	97 777	58 641	2.22	96 717	58 005	1.00	1.01
cm151a	28	0.03	1 246	744	0.04	1 246	744	0.06	1 246	744	1.00	1.00
hwb5	28	88.62	1 761 636 344	1 056 981 792	2.41	2 895 078	1 737 033	1.91	19 298	11 556	608.50	150.31
misex3	28	2.22	155 005	92 991	2.24	155 005	92 991	2.29	154 905	92 931	1.00	1.00
misex3c	28	2.26	155 006	92 991	2.27	155 006	92 991	2.32	154 906	92 931	1.00	1.00
table3	28	2.22	98 535	59 112	2.16	98 535	59 112	2.15	98 755	59 244	1.00	1.00
cm163a	29	0.02	1 144	678	0.01	1 144	678	0.02	1 144	678	1.00	1.00
in2	29	0.15	32 177	19 296	0.16	32 257	19 344	0.15	32 177	19 296	1.00	1.00
frg1	31	2.50	25 384	15 219	2.62	25 384	15 219	3.06	25 384	15 219	1.00	1.00
mod5adder	32	446.53	10 236 860 020	6 142 116 000	1.72	1 746 181	1 047 696	1.30	20 789	12 453	5 862.50	84.13
rd84	34	MO	–	–	1.25	1 699 111	1 019 451	0.92	28 039	16 794	–	60.70
cycle10	39	MO	–	–	6.24	11 338 986	6 803 379	5.71	2 520	1 500	–	4535.59

3. C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Dev*, 17(6):525–532, 1973.
4. 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.
5. R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.
6. D. M. Miller, D. Maslov, and G. W. Dueck. A transformation based algorithm for reversible logic synthesis. In *Design Automation Conf.*, pages 318–323, 2003.
7. 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.
8. 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.
9. A. Mishchenko and M. Perkowski. Fast heuristic minimization of exclusive-sums-of-products. In *Int'l Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pages 242–250, 2001.
10. M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.
11. P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler. QMDDs: Efficient quantum function representation and manipulation. *IEEE Trans. on CAD*, 35(1):86–99, 2016.
12. V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes. Reversible logic circuit synthesis. In *Int'l Conf. on CAD*, pages 353–360, 2002.
13. M. Soeken, G. W. Dueck, and D. M. Miller. A fast symbolic transformation based algorithm for reversible logic synthesis. In *Reversible Computation - 8th Int'l Conf.*, pages 307–321, 2016.
14. M. Soeken, S. Frehse, R. Wille, and R. Drechsler. RevKit: A toolkit for reversible circuit design. In *Workshop on Reversible Computation*, pages 69–72, 2010. RevKit is available at http://www.revkit.org.
15. M. Soeken, L. Tague, G. W. Dueck, and R. Drechsler. Ancilla-free synthesis of large reversible functions using binary decision diagrams. *J. Symb. Comput.*, 73:1–26, 2016.
16. 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.
17. F. Somenzi. CUDD: CU decision diagram package release 3.0. 0. 2015.
18. R. Wille, R. Drechsler, C. Osewold, and A. G. Ortiz. Automatic design of low-power encoders using reversible circuit synthesis. In *Design, Automation and Test in Europe*, pages 1036–1041, 2012.
19. R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler. RevLib: an online resource for reversible functions and reversible circuits. In *Int'l Symp. on Multi-Valued Logic*, pages 220–225, 2008. RevLib is available at http://www.revlib.org.
20. R. Wille, O. Keszocze, S. Hillmich, M. Walter, and A. G. Ortiz. Synthesis of approximate coders for on-chip interconnects using reversible logic. In *Design, Automation and Test in Europe*, 2016.
21. A. Zulehner and R. Wille. Taking one-to-one mappings for granted: Advanced logic design of encoder circuits. In *Design, Automation and Test in Europe*, 2017.