Generalizing the Concept of Scalable Reversible Circuit Synthesis for Multiple-valued Logic

Alwin Zulehner*, P. Mercy Nesa Rani†, Kamalika Datta†, Indranil Sengupta‡, and Robert Wille*
*Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
†National Institute of Technology Meghalaya, India
‡Indian Institute of Technology, Kharagpur, India
alwin.zulehner@jku.at, mercyranip@gmail.com, kdatta@nitm.ac.in, isg@iitkgp.ac.in, robert.wille@jku.at

***Abstract*—Research on reversible circuits has gained significance due to its application in quantum computations and many further areas such as the design of encoders. At the same time, the use of multiple-valued logic gained importance since this reduces the number of required entities in physical systems (e.g. in a future quantum computer). While most research is still conducted in the Boolean domain, there exist only few approaches which realize reversible circuits for multiple-valued logic. Moreover, most of the previously proposed solutions for synthesis of multiple-valued reversible circuits are not scalable and consider ternary (i.e. 3-valued circuits) only. Instead of overcoming these issues by developing new synthesis approaches for multiple-valued reversible circuits from scratch, we propose to utilize the recent accomplishments in the design of Boolean reversible circuits and to generalize them for multiple-valued logic. To this end, we discuss how to generalize *Quantum Multiple-valued Decision Diagram* based (QMDD-based) synthesis – a synthesis approach for Boolean reversible circuits which has been proven to be scalable and which has been used in several recently developed design flows. The discussions eventually show how to bridge the development gap between Boolean and multiple-valued logic for reversible circuits.**

I. Introduction

Recently, there has been a lot of research in reversible circuits synthesis, since this kind of circuits is required to model the conventional components used in quantum computations [17]. Besides that, reversible circuits have successfully been applied in verification tasks [1] as well as in the design of on-chip interconnects [29], [32] and encoders in general [38].

At the same time, multiple-valued reversible circuits received interest since they allow for reducing the physical resources needed for realizing logic functions. For instance, ternary logic realization requires 63% fewer circuit lines compared to an equivalent Boolean realization. Moreover, multiple-valued quantum gates have been shown to be physically realizable using ion trap technology [16]. Furthermore, many physical systems work in the multiple-valued domain which implies that multiple-valued logic realizations can help in reducing the cost of data conversion.

However, despite these advantages, only a rather limited number of synthesis methods for multiple-valued reversible circuits have been proposed yet (see e.g. [5], [6], [7], [8], [9], [21], [12]). Moreover, most of them suffer from a rather poor scalability and discuss ternary circuits only instead of multiple-valued circuits in general. In contrast, there exist a huge collection of scalable synthesis approaches as well as different design flows for synthesizing reversible circuits in the Boolean domain.

More precisely, *structural* approaches became popular in which the function to be realized is represented by conventional circuit descriptions (e.g. decision diagrams or gate netlists). Then, these building blocks are mapped to their reversible counterparts [33], [28]. Since most of them are not reversible, new variables are required for almost all of these mapped blocks – leading to solutions that yield reversible circuits where the number of circuit lines is magnitudes above the theoretical minimum (as e.g. observed in [31]).

As a complementary scheme, *functional* approaches have been developed that rely on a functional description of the function to be realized. The general flow of functional synthesis approaches is composed of two distinct steps. First, the function to be realized has to be embedded into a reversible one by adding ancillary inputs and garbage outputs [10], [26], [36]. After this embedding step, the resulting function is realized by a reversible circuit. While initial approaches were limited to a rather small number of variables [23], [13], [11], scalable synthesis algorithms which are based on efficient data structures like decision diagrams or Boolean satisfiability have been proposed in the recent years [25], [24]. However, the embedding step more and more constitutes the major issue of the functional approaches since it adds a significant complexity to the function to be realized (by adding inputs and outputs).

Hence, as a kind of a compromise, new design flows came up that try to combine the advantage of both design flows while overcoming their drawbacks. As an example, one-pass design has recently been proposed, where embedding and synthesis are inherently merged [37], [39]. Overall, the development of synthesis approaches for reversible circuits in the Boolean domain has seen impressive progress in the past years.

In this work, we aim to utilize this progress also for the multiple-valued logic domain. More precisely, we propose to generalize the recent accomplishments in the design of Boolean reversible circuits for the multiple-valued domain instead of developing new approaches from scratch. To this end, we propose to take a closer look at *Quantum Multiple-valued Decision Diagram* based (QMDD-based) synthesis [25] – one of the most scalable functional synthesis approaches that has recently been improved significantly [35] and that can be used for recently developed design flows that aim for merging the advantages of structural and functional approaches while overcoming their issues (as demonstrated e.g. in [37]). As the name suggests, QMDDs can be utilized to compactly represent multiple-valued quantum functionality (and, thus, reversible functions) – allowing for generalizing this scalable synthesis approach for multiple-valued logic. This way, the generalization developed in this work may serve as basis for further generalizing concepts from the purely Boolean world to the multiple-valued domain – bridging the development gap between Boolean and multiple-valued reversible circuits.

TABLE I: Reversible ternary function

x_2	x_1	x_2'	x_1'
0	0	0	0
0	1	0	1
0	2	1	2
1	0	2	1
1	1	0	2
1	2	2	0
2	0	1	0
2	1	1	1
2	2	2	2

This paper is structured as follows. In Section II, we review the background of multiple-valued reversible functions as well as multiple-valued reversible circuits. In Section III, we discuss the required aspects of QMDDs (neglecting quantum-related issues). Based on that, we show how to generalize QMDD-based synthesis for multiple-valued reversible circuits in Section IV. In Section V, we conclude the paper and give a brief outlook of future research tasks.

II. Background

In this section, we briefly recapitulate multiple-valued reversible functions as well as multiple-valued circuits.

A. Reversible Multiple-Valued Functions

We define multiple-valued functions as follows.

Definition 1. *Let $S_r = \{0, 1, \ldots, r-1\}$. Then, the mapping $f : S_r^n \rightarrow S_r^m$ describes a multiple-valued function. More precisely, f is called an* r-valued n-input m-output *function and r is called the radix of f.*

In this paper, we focus on reversible multiple-valued functions, i.e. functions that allow for computing the outputs from the inputs as well as computing the inputs from the outputs.

Definition 2. *An* r-valued *function $f : S_r^n \rightarrow S_r^m$ is reversible, iff $m = n$ and f forms a bijection.*

Example 1. *Consider the 3-valued (i.e. ternary) function f shown by means of a truth table in Table I. The function is reversible, since the number of inputs is equal to the number of outputs and each output pattern occurs only once. Therefore, the output patterns are a permutation of the input patterns.*

Since a reversible function describes a permutation of the input patterns, we also use permutation matrices to describe reversible multiple-valued functions. A permutation matrix can be defined as follows.

Definition 3. *Let $f : S_r^n \rightarrow S_r^n$ be an r-valued reversible function. Then, the permutation matrix M of f is a $r^n \times r^n$ dimensional matrix with elements $m_{i,j}$ ($0 \leq i, j < r^n$) such that*

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

The columns and rows of a permutation matrix represent the inputs and outputs of a function f, respectively. If an input is mapped to an output by f, the corresponding entry in the permutation matrix is set to 1. Consequently, each column and each row of the permutation matrix contains exactly one 1-entry.

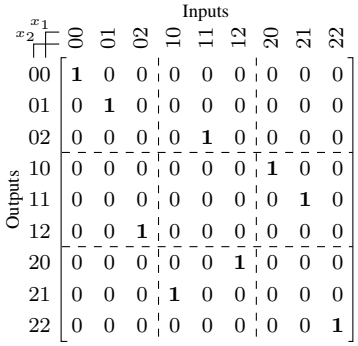

Fig. 1: Permutation matrix of the function of Table I

Example 1 (continued)**.** *Fig. 1 shows the permutation matrix P_f of f. The 1-entry in the third column of the matrix represents a mapping from input combination 02 to output pattern 12.*

B. Multiple-Valued Reversible Circuits

Reversible circuits are structurally different than conventional ones. In fact, since no feedback and fanout is allowed, these kind of circuits are composed of a set of circuit lines that are passed through a sequence of reversible gates [4], [22]. Since each gate is reversible, the overall circuit realizes a reversible function.

Definition 4. *Let $X = \{x_n, x_{n-1}, \ldots x_1\}$ be a set of n circuit lines. Then, the pair $R(X, G)$ describes a reversible circuit, where $G = g_1 g_2 \ldots g_h$ is a cascade of reversible gates g_i. Each reversible gate g_i may change the value of certain circuit lines.*

In Boolean reversible circuits, the Toffoli gate [27] got most established as building block, since it is reversible and universal (i.e. any Boolean reversible function can be realized with Toffoli gates only). The concept of a Toffoli gate can easily be extended to any r-valued logic, where r is prime [3]. Here, the respective gates are cycle gates, negations, as well as controlled versions of them. This gate set has been used e.g. for ternary reversible circuit synthesis in [5], [6], [7], [8], [9], [21], [12]. Other studies of universal multiple-valued quantum gates (and, thus, including multiple-valued reversible gates) as well as their realization with trapped ions are provided in [16]. From the findings discussed above, we generalize multiple-valued reversible unary gates as follows.

Definition 5. *Let $X = \{x_n, x_{n-1}, \ldots x_1\}$ be a set of n circuit lines. An r-valued reversible unary gate $g = U(C, t, Z_r)$ is then composed of a set of control lines $C \subseteq \{x_i^k | 0 \leq k < r, x_i \in X\}$, a target line $t \in X$, as well as a permutation of r elements denoted Z_r. Each circuit line can occur at most once in the set of control lines, i.e. $x_i^k \in C \wedge x_i^l \in C \implies k = l$. Furthermore, the target line must not occur in the set of control lines, i.e. $\{t^k | 0 \leq k < r\} \cap C = \emptyset$. If all control lines $x_i^k \in C$ evaluate to their polarity k, the value of the target lines is changed according to permutation Z_r.*

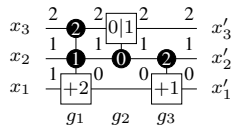

Fig. 2: 3-valued (ternary) reversible circuit

In this paper, we only use permutations describing an addition of s modulo r (denoted by $Z_r(+s)$ in the following) as well as permutations that swap two values $s, t \in S_r$ (denoted $Z_r(s|t)$ in the following). For a graphical visualization of multiple-valued circuits we adapt the representation used in [12], [15]. More precisely, we use black cycles for control lines (labeled with the polarity $k \in S_r$) and rectangles labeled with the appropriate permutation to represent the target lines.

Example 2. *Consider the ternary (3-valued) reversible circuit shown in Fig. 2 that is composed of three circuit lines and three reversible gates. Furthermore, the circuit is labeled with the circuit lines for input $x_3x_2x_1 = 211$. The first gate $g_1 = U(\{x_3^2, x_2^1\}, x_1, Z_3(+2))$ has control lines x_3 and x_2 with a polarity of 2 and 1, respectively, and target line x_1. Since the value of x_3 and x_2 matches the polarity of the respective control lines, the permutation $Z_3(+2) = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 1 \end{pmatrix}$ is applied to target line x_1. Consequently, its values changes from 1 to 0. The second gate $g_2 = U(\{x_1^0\}, x_3, Z_3(0|1))$ has control line x_1 with polarity 0 and target line x_3. Even though the control line has assigned the matching value, the value of x_3 is not altered since the permutation $Z_3(0,1) = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \end{pmatrix}$ is the identity for element 2. The third gate $g_3 = U(\{x_2^2\}, x_1, Z_3(+1))$ does not change the value of target line x_1 either since the value assigned to the circuit line x_2 (i.e. 1) does not match the polarity of the control line x_2^2 of g_3.*

III. Quantum Multiple-Valued Decision Diagrams

In this section, we review *Quantum Multiple-Valued Decision Diagrams* (QMDDs, [14], [20]). They allow for a compact representation of the unitary matrices used in quantum computations. Since permutation matrices are a special form of unitary matrices, QMDDs are also inherently suited for compactly representing *r-valued* permutation matrices. Since all these matrices typically include several redundancies, QMDDs allow for a compact representation in many cases – leading to rather scalable approaches for the simulation of quantum computations [34], quantum circuit synthesis [19], [18], or verification [30]. In the following, we omit quantum related issues of QMDDs and purely focus on the concepts required for the of synthesis of multiple-valued reversible circuits.

The general idea of QMDDs relies on a partition of the permutation matrix over its variables. This is similar to *Binary Decision Diagrams* (BDDs, [2]) that are frequently used in conventional design. In BDDs, a function is recursively decomposed (e.g. by Shannon decomposition) into sub-functions over its variables x_i. This decomposition is represented by a node labeled x_i. The two sub-functions are represented by two outgoing edges pointing to another decision diagram node or a terminal (i.e. a constant Boolean value). Even though this ends up in an exponentially large decision diagram in most cases (especially for random functions), BDDs allow for a rather compact representation of most practically relevant functions by sharing nodes that represent equal sub-functions.

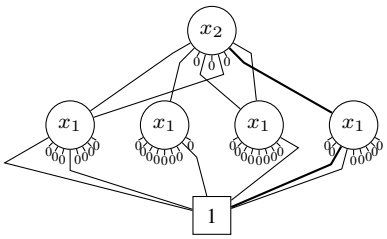

Fig. 3: QMDD representing the matrix of Fig. 1

QMDDs use a similar decomposition scheme for matrices. Recall that a permutation matrix represents a mapping from inputs (columns) to outputs (rows). In an *r-valued* logic there are r^2 possibilities how one input x_i can be mapped to an output x'_i. In the following, we denote such a mapping from an input to an output as a mapping of a *variable* x_i. We further assume that the significance of a variable x_i is defined by its index, i.e. x_{i+1} is more significant that x_i. Consider an r-valued permutation matrix for n variables, i.e. an $r^n \times r^n$ dimensional matrix. Decomposing this matrix over its most significant variable x_n yields r^2 sub-matrices of dimension $r^{n-1} \times r^{n-1}$, i.e.

$$M = \begin{bmatrix} M_{0,0} & M_{0,1} & \cdots & M_{0,r-1} \\ M_{1,0} & M_{1,1} & \cdots & M_{1,r-1} \\ \vdots & \vdots & \ddots & \vdots \\ M_{r-1,0} & M_{r-1,1} & \cdots & M_{r-1,r-1} \end{bmatrix}.$$

We represent this decomposition with a decision diagram node labeled x_n (i.e. the root node of the matrix) that has r^2 outgoing edges. In the following, we use the convention that mapping from $k \to l$ (i.e. sub-matrix $M_{l,k}$) for a variable is represented by the $(l \cdot r + k + 1)^{th}$ outgoing edge of a node (from the left; also denoted as edge $e_{k \to l}$). We recursively proceed with this decomposition of the sub-matrices until a 1×1 dimensional matrix results, which is represented by a terminal that holds the value of the matrix entry. To gain a compact representation, we again use shared nodes whenever equal sub-functions occur. For a simpler graphical visualization, we represent zero matrices (i.e. matrices composed of zeros only) with a zero stub independent of their size.[1] This way, a decision diagram with a single terminal node labeled 1 results.

Example 3. *Consider again the permutation matrix (Fig. 1) of the 3-valued reversible function shown in Table I. The corresponding QMDD is shown in Fig. 3. The path to the 1-terminal highlighted in bold traverses the eighth edge of the node labeled x_0 (i.e. $e_{1 \to 2}$) as well as the third edge of the node labeled x_1 (i.e. $e_{2 \to 0}$). Consequently, the path represents a mapping for variables x_0x_1 from 12 to 20 and, thus, the 1-entry in the sixth column of the permutation matrix shown in Fig. 1.*

[1] Note that, in QMDDs, these zero matrices are realized by an edge to the 1-terminal with an edge weight of 0.

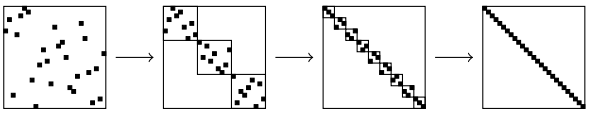

Fig. 4: General flow of the synthesis approach

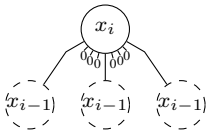

Fig. 5: Identity structure for a ternary QMDD

IV. Multiple-valued QMDD-based synthesis

In this section, we show how to generalize QMDD-based synthesis (originally proposed in [25]) to the multiple-valued domain. While the scalability of this functional approach was already demonstrated in [25], further accomplishments regarding scalability and the cost of the resulting circuits have been proposed recently in [35]. Due to these reasons, QMDD-based synthesis is one of the most promising methods for reversible synthesis and has also been utilized recently to develop new design flows in this research area [37], [39].

A. General Idea of Synthesis

The general task of a functional synthesis approach for reversible circuits is to determine a cascade of reversible gates $G = g_1 g_2 g_3 \dots g_h$ that realizes the desired function M (e.g. represented by means of a permutation matrix). To this end, we apply gates to M until the identity (i.e. the identity matrix) results. Since $M \circ M^{-1} = I$, the cascade G realizes M^{-1}. Since G is a sequence of reversible gates, its inverse G^{-1} (which realizes M due to reversibility) can easily be formed by reversing the gate sequence and exchanging the gates with their inverse. The inverse of each gate exists since they are reversible (cf. Section II).

Consequently, when using a permutation matrix to describe the function to be synthesized, the problem reduces to diagonalizing the matrix. In QMDD-based synthesis, the matrix is diagonalized variable-wise – converging towards the identity in each step. Fig. 4 visualizes the main synthesis flow for a ternary reversible circuit with three variables. A 1-entry (0-entry) in the matrix is visualized with a black (white) square.

On a QMDD-level, such a matrix-diagonalization can be realized by a breadth-first traversal of the QMDD. Each visited node is thereby transformed to the identity structure. In the identity structure, only mappings of form $j \to j$ $(0 \leq j < r)$ are allowed. Since we consider fully specified reversible functions, the corresponding edges must not point to a zero-matrix. In contrast, all other edges (representing a mapping $j \to k$ with $j \neq k$) must point to a zero-matrix.

Example 4. *Fig. 5 shows the identity structure for 3-valued (ternary) QMDDs. The first, the fifth, and the ninth edge (i.e. edges $e_{0\to0}$, $e_{1\to1}$, and $e_{2\to2}$) point to a successor node, while the remaining edges point to a zero-matrix (represented by a zero-stub).*

How such an identity structure can be established for a QMDD node is discussed in the following section.

B. Transforming a QMDD Node to the Identity

In this section, we discuss how to transform a QMDD node to the identity structure shown in Fig. 5. To this end, we assume without loss of generality that we aim for transforming a node of the QMDD labeled x_i $(n \geq i \geq 1)$[2] to the identity and that all nodes labeled with a more significant variable already establish the identity structure.

Since we aim for realizing the identity structure for a QMDD node by applying the reversible gates reviewed in Section II, we first analyze how they affect the QMDD. Recall that a QMDD represents a permutation matrix. Applying a reversible gate (i.e. another permutation matrix) means to change the permutation matrix. Since we apply gates from left to right (thus, affecting the inputs), this is indeed equivalent to swapping columns in the matrix. To establish the identity structure, one has to move all 1-entries to the sub-matrices representing a mapping $j \to j$. Since these 1-entries are represented by paths to the 1-terminal in a QMDD, we define a 1-path as follows.

Definition 6. *Let x_i be the label of an r-valued QMDD node. Then, a path from this node to the 1-terminal is called* 1-path *and is represented by a set that contains one literal for each variable x_j with $i \geq j \geq 1$. Each literal is composed of a variable x_j as well as a polarity t $(0 \leq t < r)$ and is denoted x_j^t. The polarity of a variable x_j in the path is determined by the input of the mapping represented by the outgoing edge $e_{k\to l}$ of the node labeled x_j, i.e. k – leading to a literal x_j^k. All other paths of a node labeled x_i (i.e. those that do not end in the 1-terminal) are called 0-paths since they they terminate in a zero stub (the zero terminal).*

From the matrix perspective, a 1-path in the QMDD as defined in Definition 6 describes an input combination (column) that contains a 1-entry. A 0-path of a QMDD node describes a column of the (sub) matrix that is solely composed of 0-entries (it does not contain any 1-entry).

Example 5. *Consider again the QMDD shown in Fig. 3. The 1-path highlighted in bold represents the set $p = \{x_2^1, x_1^2\}$ since it traverses the edges $e_{1\to2}$ and $e_{2\to0}$ of the nodes labeled x_2 and x_1, respectively. For simplicity in terms of writing we also write paths as concatenation of their elements (sorted descendant to the variable index) in the following, i.e. $p = x_2^1 x_1^2$.*

Based on the definition of 1-paths, we further define so-called sets of 1-paths through the edges of a QMDD node.

Definition 7. *Let x_i be the label of an r-valued QMDD node. Then, we define the set of 1-paths $P_{k\to l}$ $(0 \leq k, l < r)$ through edge $e_{k\to l}$ of this node as set of all 1-paths of the node that traverse edge $e_{k\to l}$, i.e. all 1-paths that contain a literal x_i^k. Analogously, we define the set of 0-paths $\overline{P}_{k\to l}$ of the node as set of all 0-paths that contain a literal x_i^k.*

Informally spoken, the sets of 1-paths of a node labeled x_i represent the columns of the sub-matrices resulting from a decomposition over variable x_i that contain a 1-entry. Since we

[2]Note that a node labeled x_i represents a $r^i \times r^i$ dimensional matrix.

consider only fully specified reversible functions, the number of 1-paths in the union of all sets of 1-paths through all edges is r^i, i.e. $\left|\bigcup_{0\leq k,l<r} P_{k\to l}\right| = r^i$. Furthermore, the permutation matrix contains exactly one 1-entry in each row and in each column. Consequently, for any fixed output value $l \in S_l$, there exist exactly r^{i-1} 1-entries, i.e. $\left|\bigcup_{0\leq k<r} P_{k\to l}\right| = r^{i-1}$.

From this, we can further conclude that for any fixed output value $l \in S_r$ the number of 0-paths in $\overline{P}_{l\to l}$ is equal to the number of 1-paths in all other sub-matrices with input value l, i.e. $\left|\overline{P}_{l\to l}\right| = \left|\bigcup_{k\in S_r\setminus\{l\}} P_{k\to l}\right|$.

Example 6. *Consider the root node of the QMDD shown in Fig. 3. The set of 0-paths through edge $e_{0\to 0}$, i.e. $\overline{P}_{0\to 0} = \{x_2^0 x_1^2\}$, has the same cardinality as the set of 1-paths through the edges $e_{k\neq 0\to 0}$, i.e $\bigcup_{k\in\{1,2\}} = P_{1\to 0} \cup P_{2\to 0} = \{x_2^1 x_1^1\}$.*

Consequently, the task of establishing the identity structure for a QMDD node can be reformulated as follows. For any $l \in S_r$, we have to swap the 0-paths in $\overline{P}_{l\to l}$ with the 1-paths in $\bigcup_{k\in S_r\setminus\{l\}} P_{k\to l}$. To achieve this, we choose any $p_j \in \overline{P}_{l\to l}$ and determine the most similar 1-path $p'_j \in \bigcup_{k\in S_r\setminus\{l\}} P_{k\to l}$. In this regard, the most similar path is the path in which the number of literals that occur in p_j and p'_j with a different polarity is the minimum (similar to the Hamming distance in the Boolean case). Note that the paths p_j and p'_j differ by at least one literal, since they are in different sub-matrices of the currently considered QMDD node. Hence, they have a different polarity for the literal x_i.

To swap the two paths that were chosen, one has to adjust them. More precisely, all literals except the one for x_i have to be adjusted. This can easily be established by the reversible gates reviewed in Section II. In fact, for each mismatching literal for a variable x_h (neglecting the literal for variable x_i, i.e. $i > h > 1$) one gate is required. Assume that a the literal p_j contains literal x_h^s and that p'_j contains literal x_h^t (with $s \neq t$). Then, a gate $U(\{x_i^k\}, x_h, Z_r(+(s-t)))$ is required to adjust the polarity of variable x_h. The control variable x_i^k with polarity k ensures that p'_j is changed without modifying p_j. The polarities are adjusted by adding $s-t$ (modulo r) to the value of variable x_h.

Example 6 (continued)**.** *There exists only one 0-path $p = x_2^0 x_1^2$ in $\overline{P}_{0\to 0}$ as well as a single 1-path $p' = x_2^1 x_1^1$ in the set of 1-paths $P_{1\to 0} \cup P_{2\to 0}$. To adjust these paths (neglecting the variable x_2), the gate $U(\{x_2^1\}, x_1, Z_3(+1)$ is required. The control line x_2^1 with polarity 1 ensures that path p' is modified while p does not change.*

After adjusting each variable of a path (neglecting the most significant variable x_i), another unary gate is required to swap the two paths $p_j \in \overline{P}_{l\to l}$ and $p'_j \in P_{k\to l}$. To this end, a gate $U(p_i \setminus \{x_i^l\}, x_i, Z_3(l|k))$ is applied. The set of control lines is the path that shall be swapped (excluding the literal for x_i). This way, a 1-path is moved to the edge $e_{l\to l}$ while a 0-path is moved to another edge $e_{k\to l}$ with $k \neq l$.

Example 6 (continued)**.** *To swap the paths p and p' we apply the unary gate $U = (\{x_1^2\}, x_2, Z_3(0|1))$. This establishes the identity of the root node in the QMDD for output value 0 as shown in Fig. 6. Consequently, the first edge (i.e. edge $e_{0\to 0}$) points to a successor node, while all other edges that represent a mapping from or to value 0 (i.e. edges $e_{1\to 0}$, $e_{2\to 0}$, $e_{0\to 1}$, and $e_{0\to 2}$) point to a 0-stub.*

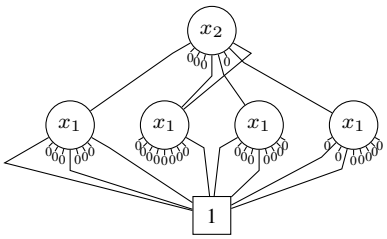

Fig. 6: Establishing the identity for value 0 of the root node

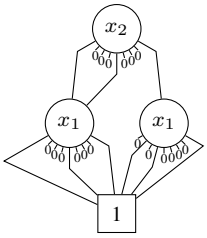

Fig. 7: QMDD with identity mapping for the root node

To obtain the identity for the currently considered node, we have to repeat the procedure discussed above for all possible output values $l \in S_r$.

Example 6 (continued)**.** *After establishing the identity for value 0, we have to establish the identity for value 1. This inherently establishes the identity mapping for value 2 since we consider ternary circuits in this example. Furthermore, since we have already established the identity for value 0, we do not have to consider any sets of 1-paths that realize a mapping from or to 0 (they must be empty).*

The set of 0-paths $\overline{P}_{1\to 1} = \{x_2^1 x_1^0, x_2^1 x_1^1\}$ contains two paths as well as the set of 1-paths $P_{2\to 1} = \{x_2^2 x_1^0, x_2^2 x_1^1\}$. Since the paths $p_1 = x_2^1 x_1^0$ and $p'_1 = x_2^2 x_1^0$ as well as the paths $p_2 = x_2^1 x_1^1$ and $p'_2 = x_2^2 x_1^1$ are equal (neglecting the literal for variable x_2), we do not have to adjust them. Consequently, we can immediately swap them with the unary gates $U(\{x_1^0\}, x_2, Z_3(1|2))$ and $U(\{x_1^1\}, x_2, Z_3(1|2))$, respectively. Fig. 7 shows the resulting QMDD, that established the identity structure for the root node.

Up to this point, we have demonstrated how to transform a QMDD node to the identity structure shown in Fig. 5. However, when doing this, we have to ensure that no other nodes that have already been transformed to the identity are modified. To this end, we have to add further control lines to each gate that is required to transform the currently considered node. These additional control lines represent the path from the top of the QMDD to the currently considered node and ensure that no other node is modified.

Example 7. *Consider again the QMDD shown in Fig. 7. The node labeled x_2 already establishes the identity structure as well as the left node labeled x_1. Since we do not want to destroy the identity structure of one of these nodes when transforming the right node labeled x_1, we have to add the control x_2^2 to each gate that is required. The polarity of the*

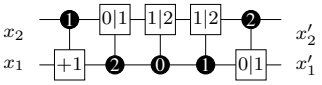

Fig. 8: Ternary reversible circuit resulting from QMDD-based synthesis

control is 2 *since the edge to the right node labeled* x_1 *is reached by the edge* $e_{2\to 2}$*. Consequently, one more gate is required to establish the identity for the whole QMDD, i.e.* $U(\{x_2^2\}, x_1, Z_3(0|1))$*. The resulting circuit is shown in Fig. 8.*

V. Conclusions

In this paper, we have shown how to generalize QMDD-based synthesis for multiple-valued reversible circuits – serving as basis for bringing recent accomplishments of the area of reversible circuit synthesis from the Boolean to the multiple-valued domain. This way, a concept is provided which bridges the design gap between Boolean and multiple-valued reversible circuits by generalizing the synthesis methods rather than developing them from scratch. In this regard, QMDD-based synthesis serves as promising approach since its internal data-structure inherently supports a compact representation of multiple-valued reversible functions, it is scalable, it has been recently improved, and it can be utilized in recently developed new design flows for reversible circuit synthesis.

Acknowledgements

This work has partially been supported by the EU COST Action IC1405 and the Austrian Agency for International Cooperation in Education and Research (OeAD) within a project under grant no. IN 08/2017.

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

[3] A. De Vos, B. Raa, and L. Storme. Generating the group of reversible logic gates. *Journal of Physics A: Mathematical and General*, 35(33):7063, 2002.

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

[5] M. H. A. Khan, M. A. Perkowski, M. R. Khan, and P. Kerntopf. Terary GFSOP minimization using Kronecker decision diagrams and their synthesis with quantum cascades. *Multiple-Valued Logic and Soft Computing*, 11(5-6):567–602, 2005.

[6] M. M. M. Khan, A. K. Biswas, S. Chowdhury, M. Hasan, and A. I. Khan. Synthesis of GF(3) based reversible/quantum logic circuits without garbage output. In *Int'l Symp. on Multi-Valued Logic*, pages 98–102, 2009.

[7] A. Kole, P. M. N. Rani, K. Datta, I. Sengupta, and R. Drechsler. Exact synthesis of ternary reversible functions using ternary Toffoli gates. In *Int'l Symp. on Multi-Valued Logic*, pages 179–184, 2017.

[8] X. Li, G. Yang, and D. Zheng. Logic synthesis of ternary quantum circuits with minimal qutrits. *JCP*, 8(8):1941–1946, 2013.

[9] S. B. Mandal, A. Chakrabarti, and S. Sur-Kolay. Quantum ternary circuit synthesis using projection operations. *Multiple-Valued Logic and Soft Computing*, 24(1-4):73–92, 2015.

[10] D. Maslov and G. W. Dueck. Reversible cascades with minimal garbage. *IEEE Trans. on CAD*, 23(11):1497–1509, 2004.

[11] D. Maslov, G. W. Dueck, and D. M. Miller. Techniques for the synthesis of reversible Toffoli networks. *ACM Trans. on Design Automation of Electronic Systems*, 12(4), 2007.

[12] D. M. Miller, G. W. Dueck, and D. Maslov. A synthesis method for MVL reversible logic. In *Int'l Symp. on Multi-Valued Logic*, pages 74–80, 2004.

[13] 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.

[14] 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.

[15] D. M. Miller and M. A. Thornton. *Multiple Valued Logic - Concepts and Representations*, volume 12 of *Synthesis lectures on digital circuits and systems*. Morgan & Claypool Publishers, 2008.

[16] A. Muthukrishnan and C. R. Stroud Jr. Multivalued logic gates for quantum computation. *Physical Review A*, 62(5):052309, 2000.

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

[18] P. Niemann, R. Datta, and R. Wille. Logic synthesis for quantum state generation. In *Int'l Symp. on Multi-Valued Logic*, pages 247–252. IEEE, 2016.

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

[20] 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.

[21] P. M. N. Rani, K. Datta, and I. Sengupta. Improved decomposition of multiple-control ternary Toffoli gates using Muthukrishnan-Stroud quantum gates. In *Int'l Conf. of Reversible Computation*, pages 202–213, 2017.

[22] M. Saeedi and I. L. Markov. Synthesis and optimization of reversible circuits - a survey. *ACM Comput. Surv.*, 45(2):21, 2013.

[23] 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.

[24] 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.

[25] 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 Conf.*, pages 85–92, 2012.

[26] M. Soeken, R. Wille, O. Keszocze, D. M. Miller, and R. Drechsler. Embedding of large boolean functions for reversible logic. *CoRR*, 2014.

[27] T. Toffoli. Reversible computing. In W. de Bakker and J. van Leeuwen, editors, *Automata, Languages and Programming*, page 632. Springer, 1980. Technical Memo MIT/LCS/TM-151, MIT Lab. for Comput. Sci.

[28] R. Wille and R. Drechsler. BDD-based synthesis of reversible logic for large functions. In *Design Automation Conf.*, pages 270–275, 2009.

[29] 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.

[30] R. Wille, D. Große, D. M. Miller, and R. Drechsler. Equivalence checking of reversible circuits. In *Int'l Symp. on Multi-Valued Logic*, pages 324–330, 2009.

[31] 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.

[32] 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.

[33] Z. Zilic, K. Radecka, and A. Kazamiphur. Reversible circuit technology mapping from non-reversible specifications. In *Design, Automation and Test in Europe*, pages 558–563, 2007.

[34] A. Zulehner and R. Wille. Advanced simulation of quantum computations. *CoRR*, abs/1707.00865, 2017.

[35] A. Zulehner and R. Wille. Improving synthesis of reversible circuits: Exploiting redundancies in paths and nodes of QMDDs. In *Conf. on Reversible Computation*, 2017.

[36] A. Zulehner and R. Wille. Make it reversible: Efficient embedding of non-reversible functions. In *Design, Automation and Test in Europe*, 2017.

[37] A. Zulehner and R. Wille. One-pass design of reversible circuits: Combining embedding and synthesis for reversible logic. *IEEE Trans. on CAD*, 2017.

[38] 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.

[39] A. Zulehner and R. Wille. Exploiting coding techniques for logic synthesis of reversible circuits. In *Asia and South Pacific Design Automation Conf.*, 2018.