On the Guidance of Reversible Logic Synthesis by Dynamic Variable Reordering

David Y. Feinstein
Innoventions, Inc.
10425 Bissonnet Street
Houston, TX, USA
david@innoventions.com

Mitchell A. Thornton
Dept. of Computer Science and Engineering
Southern Methodist University
Dallas, TX, USA
mitch@engr.smu.edu

Abstract

This paper proposes a framework that improves reversible logic synthesis by employing a dynamically determined variable order for quantum multiple-valued decision diagrams (QMDD). We demonstrate our approach through augmentation of the Miller-Maslov-Dueck (MMD) algorithm that processes the complete function specification in lexicographical order with our technique. We represent and minimize the complete specification with the QMDD and then synthesize the function specification based on the minimized variable order. The framework produces significantly smaller reversible circuits in many cases. Experimental results also show the effectiveness of using the QMDD size as a measure of the complexity of MVL and binary reversible circuits.

1. INTRODUCTION

The extensive research interest in reversible logic synthesis is motivated by its potential for quantum computing applications and low power circuit design. Quantum computing has been actively pursued for the past few decades due to its potential to achieve exponential speed up for some intractable problems such as prime factoring [1] and searching [2]. Quantum devices are necessarily logically and physically reversible logic where there is no fan-out, fan-in, or any feedback. Pioneering work by Landauer, Bennett, Fredkin, and Toffoli investigated the potential of reversible logic to create circuits that theoretically eliminate power dissipation [3,4,5,6]. The process of synthesizing reversible circuits with some desired functionality results in a serial cascade of a variety of basic reversible gates. The synthesis objective is to minimize the number of required gates while also keeping garbage outputs and ancillary inputs to a minimum [7].

Synthesis of reversible circuits is significantly more difficult than conventional logic synthesis as evidenced by the fact that current synthesizers are limited to circuits with less than 20 inputs. Furthermore, optimal synthesis has been achieved only for much smaller circuits with very few inputs [8]. Synthesis is even more difficult for *multiple-valued logic* (MVL) reversible circuits [9]. Numerous synthesis paradigms have been proposed, including search-based [10,11], template matching [12,13], formal methods using SAT [14], group theory and symmetry based techniques [15], spectral methods [16], and non-search-based lexicographical approaches [17,18]. Reversible logic synthesizers often use a combination of these techniques.

The well known MMD synthesizer combines a fast non-search-based lexicographic algorithm with a search-based template matching post processing phase [17]. While the lexicographic synthesis algorithm is relatively fast and guarantees that a circuit will be synthesized, it often produces an excessively non-optimal circuit implementation. This paper proposes the framework we refer to as "QMDDsyn" that improves the lexicographic synthesis algorithm by adjusting the lexicographic order based on the best variable order obtained by the recently introduced *quantum multiple-valued decision diagrams* (QMDD) [19].

Several researchers have used other decision diagram techniques to determine the cost function during a search-based synthesis approach [9,10,20]. To the best of our knowledge, this research is the first to investigate the relation between the function specification variable order in a decision diagram and the circuit synthesis process.

This paper is organized as follows. In Section 2 we briefly discuss reversible logic, lexicographical reversible logic synthesis, and the QMDD structure. We describe our approach in Section 3 and present the experimental results in Section 4. Conclusions and suggestions for further research appear in Section 5.

2. PRELIMINARIES

2.1. Reversible Logic

Definition 1: A binary or MVL gate/circuit is logically *reversible* if it maps each input pattern to a unique output pattern. This mapping is defined by the *transformation matrix* of the circuit. □

For binary reversible logic, the transformation matrix is of the form of a permutation matrix. For quantum circuits, the transformation matrix is a unitary matrix with complex-valued elements. An $n \times n$ reversible circuit with n inputs and n outputs requires a $r^n \times r^n$ transformation matrix, where r is the radix. The method we present in this paper is applicable to both binary reversible logic circuits (r=2) and MVL circuits (r>2) since both the QMDD data structure and the MMD synthesis approach support MVL reversible logic circuits [9,21].

Definition 2: An *n*-variable generalized *Toffoli gate* is an *n*×*n* reversible circuit that passes *n*-1 control lines without change and complements the remaining target line if, and only if, the control lines are asserted. We denote a *n*-variable Toffoli gate as TOF(x_0,x_1,…,x_{i-1},x_{i+1},…,x_{n-1};x_i), where the target line x_i is separated by a semicolon from the control lines. □

Fig. 1 illustrates an *n*×*n* Toffoli gate. The *n*-1 control lines are denoted by filled circles while the target line X_i is marked by an open circle.

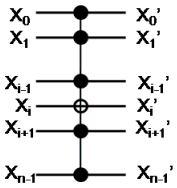


Figure 1. *n*-variable Toffoli reversible gate

Many binary and MVL reversible gates have been proposed [22]. We use the cycle by one, the cycle by two and the self sift ternary gates for the MVL example circuits presented in this work. The transformation matrix of a circuit cascade is computed by multiplying the transformation matrices of the gates, starting with the rightmost gate.

Definition 3: The *complete specification* of a reversible function is defined as a listing of all output permutations of the function in a lexicographic order. □

The synthesis of a reversible circuit is a process that produces a gate cascade implementing the complete specification while minimizing a cost function. From this point of view, it is crucial to use an appropriate metric for measuring the cost of the circuit.

Definition 4: The *quantum cost* of a circuit *C* is the number of elementary quantum operations that are used to implement the complete specification [23]. □

Definition 5: The *trivial identity reversible circuit* is a reversible circuit that implements the identity transformation matrix **I**. □

Any subsection (or sub-cascade) of a reversible circuit that (inadvertently) implements the identity transformation matrix **I** is redundant. For a cost function based on minimizing the total number of gates in the cascade, it is desirable to detect and remove such subsections.

2.2. Lexicographic Reversible Logic Synthesis

Lexicographic reversible logic synthesizers inspect a complete specification and for each input, implement a series of one or more gates that translate the input pattern into the specified output pattern. The key idea is that the synthesizer proceeds in a lexicographic order along the specification and once an input-output transformation is made, it must not be changed again. The best known lexicographic reversible synthesizer for reversible logic is the "MMD" synthesizer proposed in [17]. An MVL version of the algorithm is described in [9]. The "MMD" basic algorithm is bidirectional since the transformation proceeds simultaneously from the inputs to the outputs and from the outputs to the inputs. Such approaches may proceed in any order, and in the case of MMD, can proceed in orders from the top of the lexicographic list to the bottom, or vice versa. Lexicographic synthesizers are non-search based because they select the required translation gates for each output pattern without a search. This results in a relatively fast (although exponentially complex) algorithm which completes the synthesis of the r^n patterns of the complete specification of an *n*×*n* function with $O(nr^n)$ complexity and is guaranteed to converge. These features of the MMD lexicographic synthesis method motivated us to use it as the basis for this research.

The MMD algorithm can produce a non-optimal circuit implementation. To mitigate this limitation, the generated circuit undergoes a further minimization step using transformations based on template matching [13]. Template matching can be time consuming as new potential matches may emerge after the completion of a prior match replacement.

2.3. Quantum Multiple-valued Decision Diagrams

The transformation matrix *M* of dimension $r^n \times r^n$ representing an MVL reversible/quantum logic circuit *C* with radix r can be partitioned as

$$M = \begin{bmatrix} M_0 & M_1 & \cdots & M_{r-1} \\ M_r & M_{r+1} & \cdots & M_{2r-1} \\ \vdots & \vdots & \ddots & \\ M_{r^2-r} & M_{r^2-r+1} & \cdots & M_{r^2-1} \end{bmatrix}$$

where each M_i element is a submatrix of dimension $r^{n-1} \times r^{n-1}$. This partitioning is exploited by the QMDD structure and is used to specify the circuit *C* in a compact form [19]. In a similar manner to a reduced ordered binary decision diagram (*ROBDD*) [24], a QMDD adheres to a fixed variable ordering and common substructures (representing submatrices) are shared. A QMDD has a single terminal vertex with value 1, and each edge in the QMDD, including the edge pointing to the start vertex, has an associated complex-valued weight.

Theorem 1: *An $r^n \times r^n$ complex valued matrix M representing a reversible or quantum circuit has a unique (up to variable reordering or relabeling) QMDD representation.*

Proof: A proof by induction based on the normalization of edge weights that is performed during the QMDD construction is detailed in [19]. □

3. THE SYNTHESIS METHOD

3.1. The QMDDsyn Framework

The "QMDDsyn" framework attempts to reduce the size measured by the quantum cost of the circuit produced by the reversible logic lexicographic synthesizer. We were inspired by the approach that Fujita et al. took in minimizing binary decision diagram for classical irreversible logic [25]. Following the structural shape of the circuit, they minimized the number of wire crossings and gave priority to fan-outs encountered in the depth-first transversal from the outputs to the inputs of the circuit. In this work we use a similar approach, but in reverse - we use the best variable order corresponding to a minimized QMDD representation of the circuit to reorder the specification given to the lexicographic synthesizer. The process of template matching is delayed until it can be provided with a smaller circuit to optimize. Fig. 2 illustrates this process.

We first synthesize the function specification lexicographically to obtain the *baseline circuit C1*. In the experiments described here, we used MMD for the first step, however any initial mapping or existing circuit cascade could be used and optimized with our approach. This circuit is converted into a QMDD representation and a sifting minimization using *dynamic variable ordering* (DVO) is performed [21,26,27]. While the QMDD can be built directly from the complete specification, we use the baseline circuit *C1* to gauge the efficiency of the proposed framework's results.

The complete specification is re-ordered based on the variable order that resulted from the minimized QMDD. The re-ordered specification is used to lexicographically synthesize a new *candidate circuit C2*. Since we use the bidirectional version of the MMD algorithm, the framework also re-orders the complete specification in the *reversed* variable order of the minimized QMDD. This reversed re-ordered specification is used to synthesize another *candidate circuit C3*.

The framework then selects the smaller of circuits *C1*, *C2*, and *C3* for the final step of template matching minimization. This minimization process is time consuming and therefore it is desirable to perform it only once with the smallest circuit. Since the QMDD minimization may provide several variable orders that produce the same minimized QMDD, the process of Fig. 2 may be repeated to obtain an even smaller circuit.

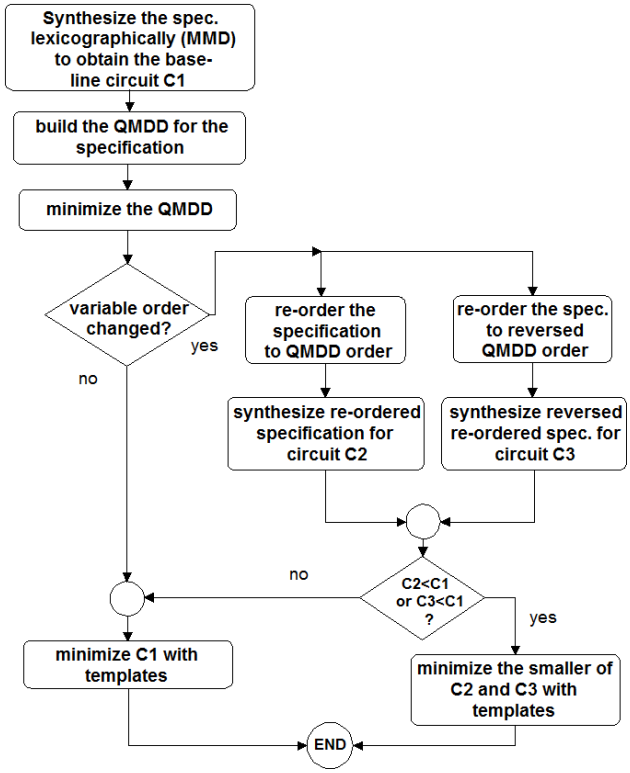


Figure 2. The "QMDDsyn" Framework

Example 1: In the following example the QMDDsyn framework demonstrates the synthesis of the 3-variable function specification representing the mapping (or permutation) of the ordered 8-tuple (0,1,2,3,4,5,6,7) to (5,3,4,7,2,6,1,0). We first use the MMD unidirectional algorithm to produce circuit *C1* of 12 gates with a quantum cost of 30 as shown in Fig. 3a. For simplicity, we first illustrate the approach using the simplified case of binary circuit benchmarks with a unidirectional version of the MMD algorithm. Results for ternary-valued circuits are given later in the paper.

While this initial circuit may not be optimal, it is an exact implementation of the specification. Using the initial variable order of $a \prec b \prec c$, *C1* is represented by a QMDD with 10 non-terminal nodes. The QMDD is minimized by the QMDDsyn framework to 9 non-terminal nodes with variable order of $b \prec a \prec c$.

The specification is reordered by first applying the new variable order as shown on the left truth table of Table 1. The results need to be ordered lexicographically (in ascending binary order) according to the input as shown in the right truth table of Table 1.

The MMD unidirectional algorithm is applied on this reordered specification represented by the 8-tuple

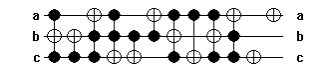
(3,4,6,5,1,2,7,0) to produce the *candidate circuit C2* of 7 gates with quantum cost of 11 as shown in Fig. 3b.

The variable lines of *C2* are re-arranged according to the variable order by moving the top line to the bottom while still keeping the same gates' connections. The final circuit *C2'* is shown in Fig. 3c.

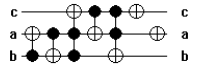
Table 1. Re-ordering the specification

b a c	b'a'c'		b a c	b'a'c'
000	011		000	011
010	110	**Ordering**	001	100
100	001	**lexicographically**	010	110
110	111	→	011	101
001	100		100	001
011	101		101	010
101	010		110	111
111	000		111	000

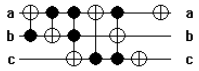
The QMDDsyn framework verifies that circuit *C2'* is equivalent to *C1* as they are both represented by the same QMDD in view of Theorem 1. This example illustrates an overall size reduction in terms of quantum cost of 11/30 (53%). In this illustrative example *C3* was not computed since only the unidirectional MMD algorithm is used to demonstrate the concept. In the following experimental results we use the bidirectional MMD algorithm for the lexicographic synthesis step and report the reversed order *candidate circuit C3* as well as *C2* size for each benchmark.



(a) Baseline circuit C1



(b) candidate circuit C2



(c) re-arranged circuit C2'

Figure 3. Minimization of a 3-variable circuit for the specification (5,3,4,7,2,6,1,0)

3.2. Circuit Complexity versus QMDD Size

In this section we analyze the relation between the reversible logic circuit size and the number of nodes in the QMDD that represents the circuit.

Lemma 1: Two equivalent reversible circuits composed of a different number of gates result in the same QMDD for a given variable ordering.

Proof: The proof results directly from the canonic property of QMDD of Theorem 1, and following the discussion in [27,28]. □

Let us assume that circuit *C* of *s* gates and circuit *C'* of *t* gates are two implementations of the same function specification, where *t*>*s*. Since Lemma 1 indicates that *C* and *C'* are represented by the same QMDD, it is clear that the QMDD size cannot be used as a measure of circuit size complexity. A reversible circuit may even include partially redundant logic that artificially increases its size as shown in [28]. Furthermore, lemma 2 illustrates that an arbitrarily large section of a synthesized circuit may be a trivial identity that cannot be reflected in the size of the QMDD.

Lemma 2: *There exists an n×n trivial identity reversible circuit cascade with any arbitrary even number of gates.*

Proof: Let *m=2k* be the number of gates in a circuit, where *k* is an integer so that *k*≥0. For *k*=0, the circuit merely consists of all the lines and therefore is a *trivial identity* circuit. For *k*=1, we make a cascade of two gates consisting of an arbitrary gate followed by its inverse. It is easy to see that such a circuit is a trivial identity reversible circuit. For *k*>1, we first build a cascade *C* of *k* arbitrary gates. We then build another cascade *C'* using exactly the same gates but in a *reversed order*. Let **A** and **A'** be the transformation matrices of *C* and *C'* respectively. Since *C* is equivalent to reversing a cascade so that the outputs exchange with the inputs, $\mathbf{A'}=\mathbf{A}^{\mathbf{T}}$. It then follows that the constructed circuit with an even number of gates *m=2k* implements the transformation matrix $\mathbf{A'}\times\mathbf{A}=\mathbf{A}^{\mathbf{T}}\times\mathbf{A}=I$, which completes the proof. □

We try to estimate the size of the binary QMDD (in non-terminal nodes) in view of the circuit specification. A trivial identity *n×n* reversible circuit requires *n* non-terminal nodes. Since a binary QMDD has four edges exiting from each non-terminal node, the upper limit for QMDD size is reached if each edge points to a different node in a unique way. This condition is a geometric series with size equal to

$$S_n = \sum_{i=1}^{n} 4^{i-1} \qquad (1)$$

for the binary case. For the ternary MVL case, the expression becomes

$$S_n = \sum_{i=1}^{n} 9^{i-1} \qquad (2)$$

However, the transformation matrix represented by the QMDD for reversible logic is a permutation matrix that has only a single '1' element in each row and each column [21]. This immense sparsity leads to significantly smaller QMDD sizes than predicted by (1) which we examine empirically in the next section.

4. EXPERIMENTAL RESULTS

4.1 QMDD Size versus Circuit Size

We explored binary and ternary circuits of 5, 6, and 7 variables, with a number of gates ranging up to 100 to gauge the relation between the circuit size and its QMDD representation. Our experimental results for binary

circuits are collected in Table 2 while our experiments with ternary circuits appear in Table 3.

For each binary circuit with a given number of gates we show the quantum cost and the number of nodes in the QMDD representing the circuit. The results in Table 2 that use published benchmarks are marked with a superscripted file identifier as follows: 1 – "mod5d1" (8 gates), 2 – "hwb5tc" (55 gates), 3 – "2of5d1" (18 gates), 4 – "rd53d1" (12 gates), and 5-ham7tc (23 gates). The other circuits were generated by selecting Toffoli gates with a random number of control lines and random connections.

The binary circuits demonstrate that the QMDD size does not follow the circuit size once the circuit size passes 20-40 gates. In contrast, the quantum cost does follow the circuit size, as expected. For example, for 5-variables, a circuit with 90 gates is represented by a QMDD with 42 nodes, less than the 46-node QMDD that represents the circuit with 40 gates. These experimental results illustrate the outcome of Lemma 2, as well as the work on partially redundant reversible logic reported in [27,28].

Table 2. QMDD Size versus Binary Circuit Size

Gates	5 -vars		6-vars		7 –vars	
	Quant. cost	QMDD size	Quant. cost	QMDD size	Quant. cost	QMDD Size
0 (Ident)	0	5	0	6	0	7
1	5	13	1	11	5	17
10	24 [1]	19 [1]	158	42	120 [4]	26 [4]
20	208	36	158 [3]	56 [3]	81 [5]	130 [5]
30	310	41	494	82	836	137
40	424	46	696	77	968	141
50	510	44	830	84	1340	134
60	313 [2]	47 [2]	1016	83	1712	153
70	682	42	1186	72	1968	149
80	808	46	1388	82	2344	140
90	918	42	1388	82	2716	164
100	1036	45	1824	83	3047	176

Since there are very few published ternary benchmark circuits, we have re-used the 5-variable circuits discussed in our previous work [21]. We randomly created additional ternary circuits with 6, and 7 variables as shown in Table 3. Following the pattern we saw with the binary circuits of Table 2, the QMDD size does not follow the circuit size, and the number of nodes is substantially lower than the maximum bound of equation (2).

Table 3. QMDD Size versus Ternary Circuit Size

Gates	5 -vars	6-vars	7-vars
10	60	23	25
25	132	171	271
50	204	369	827
75	196	619	1661
100	203	641	1275

4.2 Lexicographic Synthesis Guided by Variable Order

The synthesis results obtained with the QMDDsyn framework for binary circuits are shown in Table 4. We use the complete specification from the published benchmark circuits [29] as well as the specification obtained from the random circuits introduced in Table 2. For each specification, we list the number of variables in column 2 and the QMDD size in non-terminal nodes in column 3. The minimization results are shown in columns 4 to 5. Column 6 lists the variable order obtained during the QMDD minimization. Column 7 shows the quantum cost of the baseline circuit *C1*. Column 8 displays the quantum cost of candidate circuit *C2* obtained with the variable reorder of column 6. In both cases, the quantum cost is computed by the Maslov's RCviewer utility [29] which follows definition 4. Column 9 lists the quantum cost of the *reversed order* candidate circuit *C3*. In column 10 there is a comparison quantum cost for circuit *C4* that is synthesized with a random variable order.

The quantum cost improvement obtained when using the QMDD variable reorder is depicted in column 11. Negative percentage indicates that the QMDDsyn obtained a smaller circuit with the variable reorder. We should note that the smaller of circuits *C2* and *C3* is compared to the baseline circuit *C1*, as discussed in Section 3.1.

The results show significant improvement in the quantum cost using the approach of QMDDsyn for many benchmark circuits. There is generally a good correlation between the level of QMDD size reduction and the improvement in the quantum cost.

As is observed in Table 2, the QMDD size grows rapidly from the number of variables (for trivial circuit) to a certain fluctuating range. Smaller QMDD size indicates less permutation by the specification. It seems from our results that specifications with smaller QMDD sizes (i.e. less permutation) are likely to produce better quantum cost reduction on the QMDDsyn framework.

Since this approach is based on a heuristic, it does not work in all cases, and for several specifications ("hwb4tc", "rand5v1", "rand7v6"), the selection of a random variable order obtained a better quantum cost reduction. There is an 8% quantum cost deterioration for "rand7v6". The file "5mod5tc" resists any size change apparently due to its unique construction with 5 garbage outputs and only one controlled output.

The run times for the QMDDsyn framework are not shown in Table 4 since they are not substantially different than the "MMD" run times, except that the synthesis process is repeated two times to evaluate candidate circuits *C2* and *C3*.

It is interesting to note that the efforts of Fujita et al. to reduce the size of BDD for classical irreversible circuits based on variable order selection produced spectacular results [25]. Our results indicate that similar heuristics for reversible logic are likely to have much less dramatic improvements due to the inherent maximal connectivity associated with each circuit variable [27].

5. CONCLUSIONS AND FUTURE WORK

This paper has considered the affect of variable re-ordering of the complete specification on the lexicographic synthesis of reversible circuits. We have demonstrated a framework that changes the variable order of the function specification based on the minimization of the QMDD representation of the function. Like any heuristic approach, the experimental results show the method to be quite effective for some circuit benchmarks while sometimes having little or no benefit for others. Since the execution time of the lexicographic synthesizer is predictable, our approach does not significantly increase the overall execution time. We demonstrated that the size of ternary and binary reversible circuits has no direct correlation to the size of their QMDD representation. We are investigating the suitability of the proposed method on other search-based reversible logic synthesizers. We are also considering methods to obtain synthesized reversible circuits directly from the minimized QMDD data structure representing the function's complete specification.

Because the MVL version of MMD that we used works efficiently only up to about 4 input circuits, we are currently automating the synthesis method based on MVL circuit functions with heredity [30]. While this approach will only generate MVL example circuits that have heredity, it is possible to synthesize significantly larger cascades that will be useful for future experimentation.

ACKNOWLEDGEMENT

The authors would like to acknowledge Professor D. M. Miller for providing versions of the binary and MV-MMD synthesis programs for use in this work.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: Discrete log and factoring", In *Proc. of the 35th Annual Symposium on Foundations of Computer Science*, pp. 124-134, 1994.

[2] L. Grover, "A Fast Quantum Mechanical Algorithm for Database Search", In *Proc. ACM Symp. on Theory of Computing*, pp. 212-219, 1996.

[3] R. Landauer, "Irreversibility and Heat Generation in the Computing Process," In *IBM J. Res. and Dev.*, vol. 3, pp. 183-191.

[4] C.H. Bennett, "Logical reversibility of computation, "*IBM J. Res. Dev.*, Vol. 17, No. 6, pp. 525-532, 1973.

[5] E. Fredkin and T. Toffoli, "Conservative Logic," Int. *J. Theoretical Physics*, vol. 21, nos. 3-4, pp. 219-253, 1982.

[6] T. Toffoloi, "Reversible Computing," In *Automata, Languages, and Programming*, Springer Verlag, pp. 632-644, 1980.

[7] D.C. Marinescu and G.M. Marinescu. *Approaching Quantum Computing*. Pearson Prentice Hall, 2005.

[8] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits", In *ICCAD'02*, pp. 353-360, 2002.

[9] D. M. Miller, G. Dueck, and D. Maslov, "A Synthesis Method for MVL Reversible Logic," In *ISMVL'04*, pp. 74-80, 2004.

[10] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits", In IEEE Trans. CAD, Vol. 25, No. 11, pp. 2317-2330, 2006.

[11] P. Kerntopf, "A new heuristic algorithm for reversible logic synthesis", In *DAC'04*, pp. 835-837, 2004.

[12] K. Iwama, Y. Kambayashi and S. Yamashita, "Transformation rules for designing CNOT-based quantum circuits", In *DAC'02*, pp. 419-425, 2002.

[13] D. Maslov, G.W. Dueck, and D.M. Miller, "Simplification of Toffoli networks via templates", In *16th Symposium on Integrated Circuits and System Design*, Sao Paulo, Brazil, On CD ROM, 2003.

[14] R. Wille and D. Groβe, "Fast exact Toffoli network synthesis of reversible logic", *ICCAD'07*, pp. 60-63, 2007.

[15] Y. Guowu, X. Fei, S. Xiaoyu, W.N.N. Hong, M.A. Perkowski, "A Constructive Algorithm for Reversible Logic Synthesis", In *CEC'06,* pp. 2416-2421, 2006.

[16] D. M. Miller, "Spectral and two-place decomposition techniques in Reversible Logic", In *Proc. IEEE Midwest Symp. Circuits and Systems*, August 2002.

[17] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis", In *DAC'03*, pp. 318-323, 2003.

[18] M. Saeedi, M. Seighi, and M.S. Zamani, "A novel synthesis algorithm for reversible circuits", In *ICCAD'07*, pp. 65-68, 2007.

[19] D.M. Miller and M.A. Thornton, "QMDD: A Decision Diagram Structure for Reversible and Quantum Circuits," In *ISMVL'06*, on CD, 2006.

[20] A. Abdollahi and M. Pedram, "Efficient synthesis of quantum logic circuits by rotation-based quantum operators and unitary functional bi-decomposition", In *IWSL'05,* on CD ROM, 2005.

[21] D. M. Miller, D. Y. Feinstein, and M. A. Thornton, "QMDD Minimization using Sifting for Variable Reordering", In *Journal of Multiple-valued Logic and Soft Computing,* Vol. 13, no. 4-6, pp. 537-552, 2007.

[22] A. De Vos, B. Raa and L. Storme, "Generating the group of reversible logic gates", In *J. Phys. A:Math. Gen.* 35(2002) pp. 7063-7078, 2002.

[23] A.Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Marolus, P.Shor, T. Sleator, J.A. Smolin, and H. Weinfurter, "Elementary gates for quantum computations", In *Phys. Rev. A*., Vol. 52, no. 5, pp. 3457-3467, 1995.

[24] R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", In *IEEE Trans. Computers*, C-35(8):677-691, 1986.

[25] M Fujita, H. Fujisawa, and T. Kawato, "Evaluations and Improvements of a Boolean Comparison Method Based on Binary Decision Diagrams", In *ICCAD'88*, pp. 50-54, 1988.

[26] D. Y. Feinstein, M. A. Thornton, and D. M. Miller, "On the Data Structure Metrics of Quantum Multiple-valued Decision Diagrams", In *ISMVL'08,* pp. 138-143, 2008.

[27] D. Y. Feinstein, *Computer-Aided-Design Methods for Emerging Quantum Computing Technologies*, Ph.D. Thesis, Southern Methodist University, 2008.

[28] D. Y. Feinstein, M. A. Thornton, and D. M. Miller, "Partially Redundant Logic Detection Using Symbolic Equivalence Checking in Reversible and Irreversible Logic Circuits", In *DATE'08* pp 1378-1381, 2008.

[29] D. Maslov. Reversible logic synthesis benchmarks page. Online: http://www.cs.uvic.ca/~dmaslov/.

[30] M.A. Thornton, D.W. Matula, L. Spenner, and D.M. Miller, "Quantum Logic Implementation of Unary Arithmetic Operations", In *ISMVL'08*, pp. 202-207, 2008.

Table 4. QMDDsyn Synthesis Results

Specification Source	Vars	QMDD nodes	Minim. nodes	QMDD reduction (%)	Variable reorder	Circuit *C1* (w/o reorder) Quantum cost	Circuit *C2* (with reorder) Quantum cost	Circuit *C3* (reversed reorder) Quantum cost	Circuit *C4* (random reorder) Quantum cost	Reordered Quantum cost change (%)
4_49tc1	4	21	20	4.76%	0 2 1 3	135	126	169	199	-6.66%
hwb4tc	4	22	20	9.09%	0 2 1 3	96	87	87	88	-9.35%
hwb5tc	5	47	38	19.15%	0 2 3 1 4	488	487	518	536	0%
rand5v1	5	36	32	11.11%	2 0 3 1 4	441	312	287	200	-35.60%
rand5v2	5	44	34	22.73%	2 1 0 3 4	567	370	409	464	-34.74%
rand5v3	5	41	37	9.76%	2 3 0 1 4	421	475	360	466	-14.48%
5mod5tc	6	28	15	46.43%	5 3 1 4 0 2	185	185	185	185	0%
2of5d1	6	56	49	12.50%	3 1 0 4 5 2	525	481	507	766	-8.38%
rand6v1	6	64	42	34.38%	3 1 0 4 2 5	1114	619	661	845	-44.43%
rand6v2	6	78	67	14.10%	3 1 0 2 4 5	1518	1643	1460	1941	-3.82%
rand6v3	6	82	72	12.00%	2 1 0 3 4 5	2218	1753	2121	1951	-20.96%
rand6v4	6	77	69	10.39%	2 1 0 3 4 5	1668	1584	1767	1861	-5.03%
rd53d1	7	26	21	19.23%	2 1 0 3 4 5 6	176	188	120	139	-31.81%
rand7v1	7	95	56	41.00%	2 4 3 0 6 1 5	2997	3109	1744	2413	-41.81%
rand7v2	7	123	75	39.02%	3 4 2 0 5 16	6902	6159	4170	4987	-39.58%
rand7v4	7	141	100	29.08%	4 2 0 3 5 1 6	5028	4640	7099	6999	-7.71%
rand7v5	7	134	111	17.16%	3 2 0 5 4 1 6	6390	5770	7162	7558	-9.70%
rand7v6	7	176	147	16.48%	2 3 0 4 1 5 6	7075	7792	7609	6981	+7.54%
rand8v1	8	167	60	64.07%	2 0 5 4 6 1 3 7	2933	1136	8081	8510	-61.26%
rand8v2	8	242	150	38.02%	2 5 0 3 4 1 6 7	22968	19227	23270	22388	-16.29%
rand8v3	8	313	210	32.91%	3 5 2 0 1 6 4 7	20951	17094	13585	23105	-35.18%
rand9v1	9	114	59	48.25%	0 4 2 3 7 5 1 8 6	14242	11680	4930	22970	-65.38%
rand9v2	9	242	186	23.14%	2 0 3 6 4 7 5 1 8	55200	56569	29422	55512	-46.69%
rand9v3	9	368	337	8.42%	5 2 0 3 4 7 6 1 8	63009	55924	62431	60341	-11.24%