社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Decision Diagram Data Structure to Represent Quantum Circuit

山下　茂†　D.Michael Miller††

† 奈良先端科学技術大学院大学 情報科学研究科　〒 630-0192　奈良県生駒市高山町 8916–5
†† Faculty of Engineering, University of Victoria　PO Box 3055, Victoria, BC, CANADA V8W 3P6
E-mail: †ger@is.naist.jp, ††mmiller@cs.uvic.ca

あらまし　本稿では，ブール関数を計算する量子回路の組織的な設計手法を提案する．提案手法は，文献 [1] の設計手法のアイデアに基づくが，よりシンプルなアルゴリズムであり，また，彼らの手法が入力数に関して指数的な精度のゲートを必要とするのに比べ我々の手法は 3 種類のゲートしか用いないという点でより実用的であると言える．提案手法の導入のために，量子回路の動作を記述する *matrix function* という概念を新たに導入した．また，*matrix function* を二分決定グラフで表現する *(DDMF) Decision Diagram for a Matrix Function* というデータ構造も導入した．これらの概念の導入自体も理論的に興味深いと考えられる．

キーワード　量子回路設計, 二分決定グラフ, 制御-V ゲート

Decision Diagram Data Structure to Represent Quantum Circuit

Shigeru YAMASHITA† and D. MICHAEL MILLER††

† Nara Institute of Science and Technology　8916-5 Takayama, Ikoma, Nara, 630–0192 Japan
†† Faculty of Engineering, University of Victoria　PO Box 3055, Victoria, BC, CANADA V8W 3P6
E-mail: †ger@is.naist.jp, ††mmiller@cs.uvic.ca

Abstract　In this paper, we propose a systematic design method for quantum circuits that compute Boolean functions. Our synthesis strategy is based on the idea proposed in [1], but it is much simpler, and also may be more practical since we only use three different types of gates whereas the work in [1] may use many types of gates including rotation gates with *exponential* (with respect to the number of inputs of the desired function) accuracy. To formulate our synthesis strategy, we introduce and formulate a new concept, *matrix function*, which can describe a quantum circuit's behaviors very well. Then, in order to represent matrix functions efficiently, we naturally introduce decision diagram structures, which we call *(DDMF) Decision Diagram for a Matrix Function*. These formulation may be interesting on their own.

Key words　quantum circuit design, binary decision diagrams, controlled-V gates

1. Introduction

Recently *quantum computing* has attracted great attention by many researchers in both scientific and technological fields. Although the implementation of quantum computers is still a big challenge and seems to be very difficult, there are good reasons for pursuing the challenge of quantum computing. One of them is that quantum computing has significant potential over the current (or often referred as *classical* because the technology is based on *classical* physics in contrast to quantum physics) computing. There have been many superior quantum algorithms [2], [3], [4], and we know that the current technologies will soon reach to the microscopic level where the laws of the quantum world will dominate.

To perform quantum computation, it is indispensable to construct a circuit to compute a desired Boolean function as for classical computing. However, the circuit design seems to be very difficult for quantum circuits. Indeed there have been no good systematic ways proposed for constructing a quantum circuit for practical size (say, more than 30 variables) Boolean functions.

Recently, Abdollahi and Pedram suggested in [1] that a binary decision diagram structure (what they call *a quantum decision diagram*) can be used for the systematic synthesis of quantum circuits. Their work may be considered as early stage research since they do not give a clear formulation

of their approach with respect to the differences from conventional Boolean functions. Also their synthesis algorithm needs rotation gates with *exponential* (with respect to the number of inputs of the desired function) accuracy, which is very likely not practical.

In this paper, we generalize their idea to give clearer definitions and concepts of decision diagram structures for quantum circuit design: in their work, *quantum decision diagrams* are used to represent Boolean function like binary decision diagram, and thus there is a confusion between Boolean functions and quantum specific functions using such a data structure. Here, we introduce a rigorous definition of a *quantum function* and a new concept, *matrix function*, which can formalize the concept of designing quantum circuits quite well. We then introduce a decision diagram to represent matrix function called *a (DDMF) Decision Diagram for a Matrix Function*. This data structure is appears similar to Abdollahi and Pedram's quantum decision diagrams, however, as mentioned above, the concept is very different between DDMFs and quantum decision diagrams in [1].

Our contributions may be seen as improvements to the work in [1], and can be listed as follows:

- We generalized quantum decision diagrams in [1] to a decision diagram structure for general unitary matrices.
- We generalized the target from Boolean functions to *quantum functions* whose notion is also introduced in this paper.
- To express the functionality of quantum functions, we introduce *matrix functions*, and appropriate operators between them.
- Our decomposition algorithm is simpler, and is complete. (The algorithm presented in [1] is not complete.)
- Our method uses only three types of rotation gates, which should be more practical. (Their presented algorithm is not practical since they may use rotation gates with *exponential* (with respect to the number inputs of the desired function) accuracy.

2. Quantum Functions and Matrix Functions

In this section, we introduce the definitions of a *quantum function* and a *matrix function*. These concepts are useful for designing quantum circuits with quantum specific gates (i.e., gates whose functionality cannot be specified by Boolean functions). Such a concept may be used implicitly in the literature, but this paper is the first to discuss rigorous definitions to the best of our knowledge. Note that the term "quantum function" has been used in the literature in different contexts without clear definition. Note also that our definition may be essentially different from most of the previous usages, but our definition is very reasonable for our purpose.

2.1 Quantum States and Quantum Gates

Before introducing our definitions, let us briefly explain the basics of quantum computation.

In quantum computation, it is assumed that we can use a *qubit* which is an abstract model of a *quantum state*. A qubit can be described as $\alpha|0\rangle + \beta|1\rangle$, where $|0\rangle$ and $|1\rangle$ are two basic states, and α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. It is convenient to use the following vectors to denote $|0\rangle$ and $|1\rangle$, respectively: $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and, $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Thus, $\alpha|0\rangle + \beta|1\rangle$ can be described as a vector:

$$\alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

Accordingly, any quantum operation on a qubit can be described as a 2x2 matrix. By the laws of quantum mechanics, the matrix must be *unitary*. We call one quantum operation a *quantum gate*. For example, the operation which transforms $|0\rangle$ and $|1\rangle$ to $|1\rangle$ and $|0\rangle$, respectively, is called a NOT gate whose matrix representation is $NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

Rotation gates are often used in quantum computation. Their matrix representation is parameterized by an angle θ as follows: $R(\theta) = \begin{pmatrix} \cos(\theta/2) & i\sin(\theta/2) \\ i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$. This has an interesting feature: $R(\theta_1)R(\theta_2) = R(\theta_1 + \theta_2)$ which means that the application of $R(\theta_1)$ after applying $R(\theta_2)$ is equal to the application of $R(\theta_1 + \theta_2)$. We can consider that $R(\pi)$ and NOT work in the same way as $-iR(\pi) = NOT$ since the *global phase shift* $-i$ does not affect the result of quantum computation.

In this paper, we also consider a special gate called a V gate: $V = \frac{1+i}{2}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$. This gate has the interesting property that $V^2 = NOT$. This property will be utilized in our synthesis algorithm.

To compute a function, we need gates which operate on more than one qubit. A *controlled-U gate* is a gate that operates on two qubits in such a way that it applies U to the second qubit (called the *target bit*) when the first qubit (called the *control bit*) is in the state $|1\rangle$. In this paper we consider gates with multiple control bits as well as *negative controls* as explained in the following example. (However, note that such gates can always be decomposed into gates with one control bit [5].) Consider the example in Fig. 1. In this circuit, there are two gates, the left of which is called a *Toffoli gate*. The target bit is w and the symbol $\oplus$ means

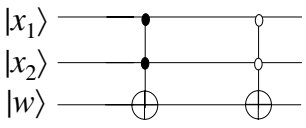

Fig. 1 A Quantum Circuit (1)

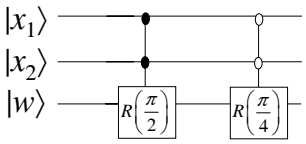

Fig. 2 A Quantum Circuit (2)

the NOT operation. The control bits are x_1 and x_2 denoted by black circles. This gate performs NOT on w only when both x_1 and x_2 are in the state $|1\rangle$. Consider the gate on the right side in the same figure. The white circles denote negative controls, which means the gate performs NOT only when both of the states of x_1 and x_2 are in the state $|0\rangle$.

We can consider any unitary operation for a controlled gate. For example, the gate's functionalities of the left and the right gates in Fig. 2 are $R(\frac{1}{2}\pi)$ and $R(\frac{1}{4}\pi)$, respectively.

2.2 Quantum Functions and Matrix Functions

Consider Fig. 1. This circuit transform the state of the third bit $|w\rangle$ into $|w \oplus f(x_1, x_2)\rangle$, where $f(x_1, x_2) = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$. Thus, this circuit realizes the Boolean function $f(x_1, x_2) = x_1 \cdot x_2 + \overline{x_1} \cdot \overline{x_2}$, and indeed this type of circuit can be used as an oracle for the famous quantum algorithm called Grover Search [4].

Consider Fig. 2. In this circuit, we cannot write the resultant state of the third bit as $|w \oplus f(x_1, x_2)\rangle$ such that $f(x_1, x_2)$ is a Boolean function.

Although our goal is to construct a quantum circuit to calculate a Boolean function (e.g., the circuit shown in Fig. 1), we would like to consider quantum circuits which can use rotation gates such as the circuit in Fig. 2. The reason is that such circuits are very useful as a part of the desired circuit as we will see in our design strategy. Since the functionality of such circuits is not Boolean, we need to define how to represent the functionality of such circuits. Before introducing our representation, we need the following definition.

[Definition 1] A quantum function with respect to n Boolean variables $x_1, x_2, \cdots, x_n$ is a mapping from $\{0,1\}^n$ to a qubit state.

See the third bit in the circuit in Fig. 2 again. If the initial state of $|w\rangle$ is $|0\rangle$, the resultant state of the third bit can be seen as a quantum function. This quantum function is described as $qf_2(x_1, x_2)$ in Table 1. For example, the resultant quantum state becomes $R(\frac{1}{4}\pi)\,|0\rangle$ when $x_1 = 0, x_2 = 0$. Thus, $qf_2(0,0)$ is defined as $R(\frac{1}{4}\pi)\,|0\rangle$ as shown in the table.

Note that a Boolean function can be seen as a special case of quantum functions. For example, Table 2 shows the quantum function of the resultant third qubit in the circuit in Fig. 1 when the initial state of $|w\rangle$ is $|0\rangle$. This can be considered as the output of a Boolean function when $|0\rangle$ and $|1\rangle$ are considered as Boolean values 0 and 1, respectively.

Table 1 A truth table for a quantum function

x_1, x_2	qf_2
$0,0$	$R(\frac{1}{4}\pi)\,\|0\rangle$
$0,1$	$\|0\rangle$
$1,0$	$\|0\rangle$
$1,1$	$R(\frac{1}{2}\pi)\,\|0\rangle$

Table 2 A truth table for a classical function

x_1, x_2	qf_1
$0,0$	$\|1\rangle$
$0,1$	$\|0\rangle$
$1,0$	$\|0\rangle$
$1,1$	$\|1\rangle$

Table 3 A truth table for a matrix function

x_1, x_2	mf_2
$0,0$	$R(\frac{1}{4}\pi)$
$0,1$	I
$1,0$	I
$1,1$	$R(\frac{1}{2}\pi)$

Table 4 A truth table for constant matrix functions

x_1, x_2	$CM(I)$	$CM(R(\frac{1}{2}\pi))$
$0,0$	I	$R(\frac{1}{2}\pi)$
$0,1$	I	$R(\frac{1}{2}\pi)$
$1,0$	I	$R(\frac{1}{2}\pi)$
$1,1$	I	$R(\frac{1}{2}\pi)$

The value of a quantum function $q(x_1, x_2, \cdots, x_n)$ can always be expressed as $mf(x_1, x_2, \cdots, x_n)\,|0\rangle$, where $mf(x_1, x_2, \cdots, x_n)$ is a mapping from $\{0,1\}^n$ to 2x2 unitary matrices. It is convenient to consider $mf(x_1, x_2, \cdots, x_n)$ instead of $q(x_1, x_2, \cdots, x_n)$ itself, thus we introduce the following definition.

[Definition 2] A matrix function mf with respect to n Boolean variables $x_1, x_2, \cdots, x_n$ is a mapping from $\{0,1\}^n$ to 2x2 matrices.

Table 3 shows the matrix function $mf(x_1, x_2)$ for the quantum function qf_2 in Table 1.

We define a special type of matrix function called *constant matrix function* as follows.

[Definition 3] A matrix function $mf(x_1, x_2, \cdots, x_n)$ is called a constant matrix function if $mf(x_1, x_2, \cdots, x_n)$ are the same for all the assignments to $x_1, x_2, \cdots, x_n$. $CM(M)$ denotes a constant matrix function that always equals to the matrix M.

Table 4 shows the truth tables for constant matrix functions, $CM(I)$ and $CM(R(\frac{1}{2}\pi))$.

Table 5 An example of an operator $\oplus$.

x_1, x_2	mf_1	mf_2	$mf_1 \oplus mf_2$
$0, 0$	$R(\frac{1}{2}\pi)$	$R(\frac{1}{2}\pi)$	$R(\pi)$
$0, 1$	I	I	I
$1, 0$	I	$R(\frac{1}{4}\pi)$	$R(\frac{1}{4}\pi)$
$1, 1$	$R(\frac{1}{2}\pi)$	$R(\frac{1}{4}\pi)$	$R(\frac{3}{4}\pi)$

Table 6 An example of an operator $*$.

x_1, x_2	mf_1	f_1	$f_1 * mf_1$
$0, 0$	$R(\frac{1}{2}\pi)$	1	$R(\frac{1}{2}\pi)$
$0, 1$	I	0	I
$1, 0$	$R(\pi)$	1	$R(\pi)$
$1, 1$	$R(\pi)$	0	I

By using the matrix function mf_2 in Table 3, we can easily see how the circuit in Fig. 2 transforms the third qubit $|w\rangle$: $|w\rangle$ is transformed to $mf_2(x_1, x_2)\,|w\rangle$. For example, when $x_1 = 0, x_2 = 0$, $|w\rangle$ is transformed to $R(\frac{1}{4}\pi)\,|w\rangle$.

For matrix functions, an operator "$\oplus$" is defined as follows.

[Definition 4] Let $mf_1(x_1, x_2, \cdots, x_n)$ and $mf_2(x_1, x_2, \cdots, x_n)$ be matrix functions with respect to x_1 to x_n. Then $mf_1(x_1, x_2, \cdots, x_n) \oplus mf_2(x_1, x_2, \cdots, x_n)$ is a matrix function $mf(x_1, x_2, \cdots, x_n)$ such that $mf(x_1, x_2, \cdots, x_n) = mf_1(x_1, x_2, \cdots, x_n) \cdot mf_2(x_1, x_2, \cdots, x_n)$ where $\cdot$ means normal matrix multiplication.

Consider the example in Table 5. Note that if mf_1 and mf_2 are considered to be Boolean functions like qf_1 in Table 2, this operator corresponds to the EXOR of the two Boolean functions.

We introduce another operator "$*$" as follows:

[Definition 5] Let $mf(x_1, x_2, \cdots, x_n)$ and $f(x_1, x_2, \cdots, x_n)$ be a matrix function and a Boolean function with respect to x_1 to x_n, respectively. Then $f(x_1, x_2, \cdots, x_n) * mf(x_1, x_2, \cdots, x_n)$ is a matrix function which equals to $mf(x_1, x_2, \cdots, x_n)$ when $f(x_1, x_2, \cdots, x_n) = 1$, and equals to I when $f(x_1, x_2, \cdots, x_n) = 0$.

Consider the example in Table 6. Note that if mf is considered to be a Boolean function like qf_1 in Table 2, this operator corresponds to the AND of the two Boolean functions.

2.3 Decision Diagrams for Matrix Functions

A matrix function mf for a quantum function qf can be expressed efficiently by using a binary decision diagram structure as in the case of Boolean functions [6]. For this purpose, we introduce *a (DDMF) Decision Diagram for a Matrix Function* as follows:

[Definition 6] A Decision Diagram for a Matrix Function (DDMF) is a directed acyclic graph with three types of nodes:

- a single terminal node corresponding to the identity matrix I,
- a root node with an incoming edge having a weighted matrix M, and
- a set of non-terminal (internal) nodes.

Each internal and the root node are associated with a Boolean variable x_i, and have two outgoing edges which are called 1-edge (solid line) leading to another node (the 1-child node) and 0-edge (dashed line) leading to another node (the 0-child node). Every edge has an associated matrix.

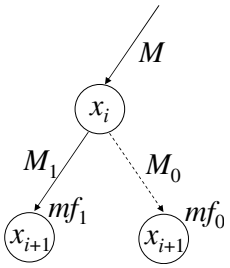

Fig. 3 An internal node in a DDMF

The matrix function represented by a node is defined recursively as follows:

- The matrix function represented by the terminal node is the constant matrix $CM(I)$.
- The matrix function represented by an internal node (or the root node) whose associated variable is x_i is defined as follows: $x_i * (CM(M_1) \oplus mf_1) \oplus \overline{x_i} * (CM(M_0) \oplus mf_0)$, where mf_1 and mf_0 are the matrix functions represented by the 1-child node and the 0-child node, respectively, and M_1 and M_0 are the matrices of the 1-edge and the 0-edge, respectively. (See an illustration of this structure in Fig. 3.)
- The root node has one incoming edge that has a matrix M. Then the matrix function represented by the whole DDMF is $CM(M) \oplus mf$, where mf is a matrix function represented by the root node.

Like conventional BDDs, we achieve a canonical form for DDMF if we impose the following restriction on the matrices on every edge.

[Definition 7] A (DDMF) is canonical when

- all the matrices on 0-edges are I,
- there are no redundant nodes: i.e. no node has 0-edge and 1-edge pointing to the same node with I as the 1-edge matrix,
- common sub-graphs are shared: there are no two sub-graphs that have the same structure.

Any DDMF can be converted to canonical form by using the following transformation from the terminal node to the root node.

Suppose the matrices on incoming edge, 0-edge and 1-edge of a node be M, M_0 and M_1, respectively. Then, if M_0 is not I, we modified these three matrixes as follows:

- The matrix on the incoming edge changed to be MM_0.

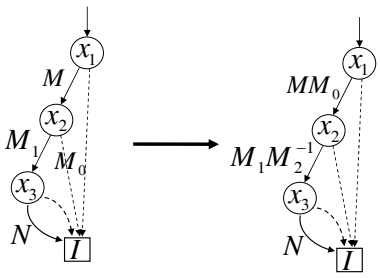

Fig. 4 Conversion to the canonical form

- The matrix on the 1-edge changed to be $M_1M_0^{-1}$.
- The matrix on the 0-edge changed to be I.

It is easily verified that this transformation does not change the matrix function represented by the DDMF. See the example in Fig. 4 where the matrix on 0-edge of the node x_2 is converted to I. In the example, the matrices I on edges are omitted.

Note: The decision diagram structure is similar between DDMFs and the quantum decision diagrams discussed in [1]. However, quantum decision diagrams are used to represent conventional Boolean functions whereas DDMFs are used for representing matrix functions; the terminal node of a DDMF is a matrix I. Also a weight on an edge in DDMFs is generalized to any matrix. Thus, DDMFs can be considered as a generalization of quantum decision diagrams to treat matrix functions rather than Boolean functions. (As we have seen in Table 2, Boolean functions can be seen as a special case of quantum functions.)

We will use the same operators for DDFMs as for matrix functions. More concretely, for matrix functions mf, mf_1 and mf_2, we will say

- (DDMF for mf) = (DDMF for mf_1) $\oplus$ (DDMF for mf_2) if $mf = mf_1 \oplus mf_2$, and
- (DDMF for mf) = (DDMF for mf_1) $*$ (DDMF for mf_2) if $mf = mf_1 * mf_2$.

3. Quantum Circuit Design from Matrix Functions

As in conventional logic design, if we have already constructed quantum circuits for mf_1 and mf_2, then we can construct quantum circuits for $mf_1 \oplus mf_2$ and $mf_1 * mf_2$. Thus, we can construct a quantum circuit for a given matrix function mf if we can decompose it to $mf_1 \oplus mf_2$ or $mf_1 * mf_2$ where mf_1 and mf_2 are easier to implement than mf; we may consider the quantum circuit design in a similar way as conventional logic synthesis.

We will now show the two types of quantum matrix decomposition.

3. 1 Decomposition of Type 1

When we want to make a circuit for a matrix function that can be decomposed to $mf_1 \oplus mf_2$, we simply concatenate two circuits for mf_1 and mf_2. This is easily verified by the property of the operator $\oplus$. Formally, we have the following observation.

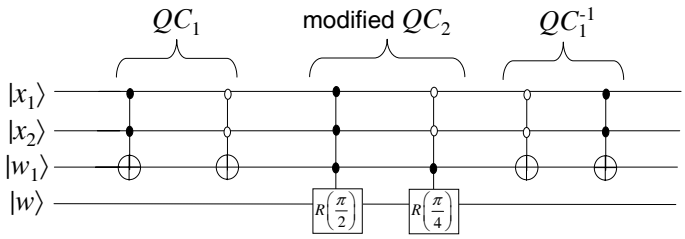

Fig. 5 A Quantum Circuit for $mf_1 * mf_2$

[Proposition 1] Let quantum circuits QC_1 and QC_2 be the circuits corresponds to the transformations by matrix functions mf_1 and mf_2, respectively. Then the circuit obtained by concatenating QC_1 and QC_2 corresponds to the transformation by a matrix function $mf_1 \oplus mf_2$.

Intuitively, the transformation by the circuit corresponding to $mf_1 \oplus mf_2$ is just a combination of the transformations for mf_1 and mf_2.

3. 2 Decomposition of Type 2

If we have already designed two quantum circuits QC_1 and QC_2 corresponding to the transformations by matrix functions mf_1 and mf_2, respectively, we can obtain the quantum circuit corresponding to the transformation by a matrix function $mf_1 * mf_2$ in the following way.

Step. 1 Construct QC_1. Let the qubit where QC_1 calculates the quantum function be w_1.

Step. 2 Construct a circuit identical to QC_2 except that w_1 is added to the control bit of every gate in the circuit. Add this circuit after the circuit obtained at Step. 1.

Step. 3 Construct an inverse circuit of QC_1. Add this circuit after the circuit obtained at Step. 2.

This approach is illustrated by the example in Fig. 5 where QC_1 (which corresponds to mf_1) and QC_2 (which corresponds to mf_2) correspond to the circuits in Figs. 1 and 2, respectively.

4. Quantum Circuit Design using DDMFs

In this section, we show some techniques for designing a quantum circuit from a given DDMF. Based on the presented techniques, we can construct a Boolean function with only controlled-NOT, controlled-V and controlled-V^{-1} gates. Thus, our method can be considered to be more practical than the method [1] which needs many types of rotation gates with *exponential* (with respect to the number inputs of the desired function) accuracy in the worst case.

4. 1 Easy Case 1

If we want to construct a circuit corresponding to a DDMF

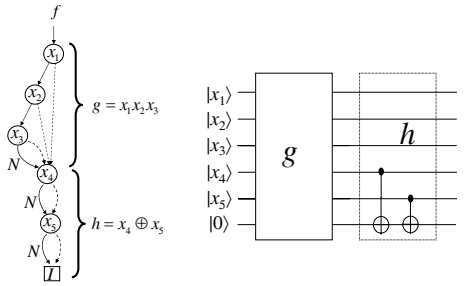

Fig. 6 Decomposition of DDMFs: Easy Cases

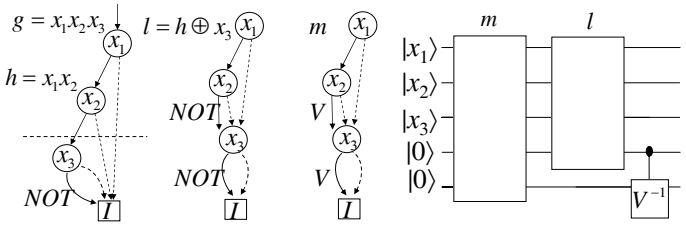

Fig. 7 Decomposition of DDMFs: Complex Case

for f where (DDMF for f) = (DDMF for g) $\oplus$ (DDMF for h), then we simply concatenate two circuits for mf_1 and mf_2 to get the desired circuit. This is exactly the same as the Decomposition Type 1 in the previous section.

See the example in Fig. 6 where we want to construct a circuit for a Boolean function $f = g \oplus h$. In this example, (DDMF for f) = (DDMF for g) $\oplus$ (DDMF for h). The desired circuit is just a concatenation of the circuits for g and h.

4.2 Easy Case 2

[Definition 8] A (DDMF) is called a chain if, for all the nodes in the DDMF, the 1-edge and 0-edge go to the same node.

If the targeted DDMF is a *chain structure*, we can construct the circuit as follows:

- For each non-terminal node where the associated variable is x_i and the matrix on 1-edge is M, we create a controlled-M_i gate controlled by x_i.
- We concatenate all the above gates to get the desired circuit.

For example, the DDMF for h in Fig. 6 is a chain structure, and thus it has a simple implementation as shown in the figure.

Note that this technique is based on a very similar idea found in [1].

4.3 Complex Case

If there is no chain structure in the bottom part of a given DDMF, we cannot use the decomposition techniques described above. Even in such a case, by using the following method, we can decompose any DDMF into simpler DDMFs.

Assume that we want to implement a Boolean function g whose corresponding DDMF does not have a chain structure in the bottom part. Let the bottom non-terminal node be v_i, and the associated variable to it be x_i. Then, we can construct the circuit as follows:

- Let a Boolean function h be a function that corresponds to all paths from the root node to v_i in the DDMF for g.
- Let a Boolean function l be $h \oplus x_i$.
- Let m be a DDMF that is obtained from the DDMF for l with replacing all NOT on edges to V.
- We can construct the desired circuit as shown in Fig. 7.

An example of this approach is shown in Fig. 7.

Although the detail is omitted, it can be easily shown (as illustrated in Fig. 7) that the DDMFs for l and m have chain structures in their bottom part, and they are easier to implement than g. Again due to the space, we omit the detail, but we can construct a quantum circuit for a given Boolean function by recursively apply the above three cases.

5. Conclusion

In this paper, we introduced a new concept "matrix functions" which are very useful to describe how logic synthesis is performed for quantum circuits. We also introduced DDMFs and their rigorous definitions. We showed that there is a systematic way to decompose a DDMF to construct the corresponding quantum circuit.

We are now developing a synthesis program based on the strategies presented in this paper. Our obvious future work is to evaluate the efficiency of the proposed method by applying the method to benchmark circuits [7]. Also, we will investigate alternative synthesis methods based on DDMFs.

References

[1] A. Abdollahi and M. Pedram: "Analysis and synthesis of quantum circuits by using quantum decision diagrams", Proc. of the Design, Automation and Test in Europe (DATE'06), pp. 317–322 (2006).

[2] D. Deutsch and R. Jozsa: "Rapid solutions of problems by quantum computation", Proceedings of the Royal Society of London, pp. 553–558 (1992).

[3] P. W. Shor: "An algorithm for quantum computation: discrete log and factoring", Proc. 35th Annual IEEE Symposium on Foudations of Computer Science, pp. 124–134 (1994).

[4] L. K. Grover: "A fast quantum mechanical algorithm for database search", Proceedings of 28th ACM Symposium on Theory of Computing, pp. 212–219 (1996).

[5] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin and H. Weinfurter: "Elementary gates for quantum computation", Physical Review A, **52**, 5, pp. 3457–3467 (1995).

[6] R. E. Bryant: "Graph-based algorithm for Boolean function manipulation", IEEE Trans. Comput., **C-35**, 8, pp. 667–691 (1986).

[7] D. Maslov, G. Dueck and N. Scott: "Reversible logic synthesis benchmarks page" (2005). http://www.cs.uvic.ca/ maslov/.