Canonical Representation and Efficient Synthesis of Quantum Logic Circuits

Afshin Abdollahi and Massoud Pedram
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA

1. Abstract

Quantum information processing technology is in its pioneering stage and no efficient method for synthesizing quantum circuits has been introduced so far. This paper introduces an efficient analysis and synthesis framework for quantum logic circuits. The proposed synthesis algorithm and flow can generate a quantum circuit using the most basic quantum operators, i.e., the rotation and controlled-rotation primitives. All other quantum logic gates can be built by composition of these primitive ones. We will introduce the notion of quantum factored forms, and develop a canonical and concise representation of quantum logic circuits. The proposed quantum decision diagrams (QDD's) are amenable to efficient manipulation and optimization including recursive unitary functional bi-decomposition. Subsequently, an effective QDD-based algorithm will be developed and applied to automatic synthesis of quantum logic circuits. This representation will produce a rigorous graph-based framework for the analysis and synthesis of quantum logic circuits. Illustrative examples are provided throughout the paper to facilitate the understanding of the algorithms and methods. Comparing the synthesis results for examples provided in this paper with previous approaches demonstrates the effectiveness of the proposed approach.

2. Introduction

We are beginning to reach the fundamental limits of the materials used in the planar CMOS process, the process that has been the basis for the semiconductor industry for the past 30 years. Further improvements in the planar CMOS process can continue for the next decade or so by introducing new materials into the basic CMOS structure. However, as we look forward 10-15 years, it becomes clear that even with the introduction of new materials, most of the known technological capabilities of the CMOS device structure will have reached their limits [1]. In order to continue to drive information technology advances, it becomes essential to investigate new "beyond CMOS" devices and structures, appropriate models of computation, and algorithms that may provide a more cost-effective alternative to CMOS.

Quantum computation is the extension of classical computation to the processing of quantum information, using quantum systems such as individual atoms, molecules, or photons. It has the potential to bring about a spectacular revolution in computer science and engineering. Current-day electronic computers are not fundamentally different from purely mechanical computers: the operation of either can be described fully in terms of classical physics. By contrast, computers could in principle be built to profit from actual quantum phenomena that have no classical analogue, such as entanglement and interference, sometimes providing exponential speed-up compared with classical computers. Quantum computing by inheriting powerful properties from Quantum Mechanics, allows building computers much more compact, efficient and less power consuming than the existing computers [2]. Whereas a classical computer obeys the well understood laws of classical physics, a quantum computer is a device that harnesses physical phenomenon unique to quantum mechanics (especially quantum interference) to realize a fundamentally new mode of information processing. Quantum computers can evolve a superposition of quantum states - each following coherently distinct computational paths until the final output is obtained. Such "quantum parallelism" could potentially outstrip power of classical computers [3]. Certain problems for which there is no polynomial solution in classical domain can be solved in polynomial time in quantum domain (e.g., the factoring problem). Similarly, the complexity of some other problems (e.g., database search and Boolean satisfiability) can be reduced by transforming them into the quantum domain [4]. Indeed, quantum circuits have the ability to perform massively parallel computations in a single time step [5][6]. Hence quantum computing has become a very attractive research area, which is expected to play an increasingly critical role in building more efficient computers.

Conventional computers are mostly built from logically irreversible blocks, which results in information loss during each logic operation. According to Landauer's principle [7][8], irreversible logic computations, necessarily dissipate $log(2)kT$ J energy (in the form of heat), for every lost bit of information, regardless of its implementation technology, where k is the Boltzmann's constant and T is the temperature. Hence, for zero power dissipation (producing zero heat), it is a necessary (but not sufficient) condition for a computing system to perform reversible computations and be constructed from reversible logic components. Indeed, computation can utilize a series of steps, each logically reversible, and this in turn allows physical reversibility [9]. A reversible circuit has the same number of inputs and outputs and is a one-to-one mapping between vectors of input and output values [10]. Since all quantum operations are essentially reversible (see section 3), quantum computers are capable of not dissipating any energy due to this effect.

Quantum computation has its origins in highly specialized fields of theoretical physics, but its future unquestionably lies in the profound impact it will have on the lives of all mankind. At this very moment, obstacles are being surmounted that will provide the knowledge needed to propel quantum computers up to their just position as the fastest computational machines in existence. Error correction has made promising progress to date, nearing a point now where we may have the tools required to build a computer robust enough to adequately endure the effects of quantum decoherence. Quantum hardware, on the other hand, remains an up-and-coming field, but the work done thus far suggests that it will only be a matter time before we have devices large enough to test Shor's factoring algorithm and other quantum algorithms. According to the results of [11]-[15] building quantum computers is feasible and the progress toward these realizations is fast-paced. Consequently, quantum computers will emerge as superior computational devices.

Quantum mechanics and quantum computing are established research areas; however, systematic design methods and logic design for quantum circuits and systems is at a primitive stage. Computer aided design of quantum circuits is even less developed, which motivates rigorous research aimed at developing CAD techniques and tools for quantum circuits. Nearly all quantum algorithms (such as Shor's factoring and Grover search algorithms) require the implementation of a quantum oracle (logic circuit i.e., a circuit that for binary inputs only generates binary outputs), whose function is to recognize solutions to a given problem. To completely exploit the "quantum parallelism," this oracle should be realized by using quantum gates because it must be able to handle an arbitrary superposition of basis vectors (quantum states.) A key problem is thus how to construct a minimum-cost realization of this kind of quantum logic circuit. Automated synthesis of standard Boolean logic circuits is a well-studied area with many efficient algorithms. However, no efficient method for synthesizing quantum circuits has been introduced so far. Previous work on quantum logic synthesis is mostly based on search-based approaches, which require enormous computational complexity (e.g., matrix decomposition, local circuit transformations, spectral techniques, and evolutionary approaches.) In this paper a canonical decision diagram based representation of quantum circuits is presented and a CAD methodology and novel techniques for synthesis of quantum logic circuits based on these decision diagrams are described. As stated before, every quantum circuit is reversible, so classical binary reversible synthesis and quantum synthesis are closely related research areas. Feasibility of reversible logic circuits has been technologically demonstrated [16]; the proposed approach is also applicable to synthesis of such circuits. The reminder of this paper is organized as follows: In section 3, some fundamental aspects of quantum mechanics required to understand the proposed approach is presented. Section 4, summarizes the previous work on quantum circuit synthesis.

In section 5, the proposed technique is presented which includes the introduction of quantum factored forms, quantum decision diagrams (QDD's) and QDD-based quantum circuit synthesis. The conclusions are provided in section 6.

3. Fundamentals of Quantum Computing

In quantum computation quantum bits (qubits), derived from the states of micro-particles such as photons, electrons or ions are used instead of classical binary bits to represent information. For example, two possible spin rotations of an electron are represented as $\begin{bmatrix}1\\0\end{bmatrix}$ and $\begin{bmatrix}0\\1\end{bmatrix}$, which are the basis states (basis vectors) of this computational quantum system [17][18]. Each particle in a quantum system is represented by a wave function inheriting the powerful concept of superposition of states. For example, the state of a particle p_1 may be represented by a wave function $\Psi_1 = \alpha_1\begin{bmatrix}1\\0\end{bmatrix} + \beta_1\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha_1\\\beta_1\end{bmatrix}$ where the coefficients α_1 and β_1 are in general complex and $|\alpha_1|^2+|\beta_1|^2=1$. If another particle p_2 with the wave function $\Psi_2 = [\alpha_2 \quad \beta_2]^T$ is added to the system, then the resulting wave function of the system comprising of these two particles will be: $\Psi_1 \otimes \Psi_2 = [\alpha_1\alpha_2 \quad \alpha_1\beta_2 \quad \beta_1\alpha_2 \quad \beta_1\beta_2]^T$. However, the state of a two-qubit system in general is a vector $\Gamma = [\gamma_1 \quad \gamma_2 \quad \gamma_3 \quad \gamma_4]^T$ of four complex values with $|\Gamma|=1$. The quantum state Γ is a '*product state*' if it can be expressed as $\Gamma = \Psi_1 \otimes \Psi_2$ where Ψ_1 and Ψ_2 are one-qubit states. However, it is not always possible to express a two-qubit state as the tensor product of two one-qubit states. In this case, the state is called *entangled*. An example of an entangled state is $[1/\sqrt{2} \quad 0 \quad 0 \quad 1/\sqrt{2}]^T$.

In general, the wave function of a quantum system with n qubits represents an arbitrary superposition of 2^n states while in a classical system n bits represent only 2^n distinct states. Therefore the space of quantum systems is exponentially larger than that of the classical binary systems. Analysis (and by extension, synthesis) of quantum logic circuits is more difficult than that of the digital logic circuits because the former requires manipulation of matrices and bases in Hilbert space whereas the latter requires binary, or at most multi-valued, logic operations.

Quantum operators over a set of qubits are modeled as matrix operations. As an example, for a quantum system comprising of a single particle p_1, a quantum operator (gate) is represented by a 2×2 (in general complex) unitary matrix U which transforms state $\Psi_1 = [\alpha_1 \quad \beta_1]^T$ to state $\Psi_2 = U\Psi_1$. Recall that a matrix U is unitary exactly if $UU^+=I$ where U^+ is the hermitian (complex conjugate transpose) of U. Since matrix U is unitary, the inverse of this gate is matrix U^+, which is the inverse of U.

An important class of quantum operators is the *rotation operator*. For example, a θ rotation around the X axis in Bloch sphere representation [4] is defined by:

$$R_x(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}.$$

The following relation shows that rotation operators around X are commutative with respect to matrix multiplication:

$$R_x(\theta_1)R_x(\theta_2) = R_x(\theta_2)R_x(\theta_1) = R_x(\theta_1+\theta_2).$$

In general for an n-qubit system, a quantum operation (or gate) is represented by a $2^n\times2^n$ unitary matrix. An example of a 2-qubit gate is the controlled-U gate depicted in Figure 1. For a 2×2 unitary matrix U, the controlled-U gate works as follows: when the control signal a is $[1 \quad 0]^T$, $q=b$ and when it is $[0 \quad 1]^T$, then $q=Ub$. For both cases, $p=a$.

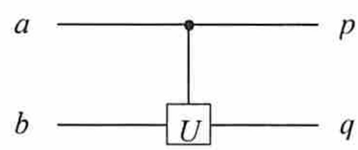

Figure 1. Schematic diagram of a controlled-*U* gate.

Similar to controlled-*U* operator, one can easily define a significant class of 2-qubit operators as the *controlled-rotation operator*. Rotation operators are elementary and easily realizable in most implementations of quantum computation [4], for example, nuclear magnetic resonance and ion trap realizations. We will also show that rotations and controlled-rotations around *X* axis are *universal*. (A set of gates is *universal* if every quantum logic function can be constructed with this set of gates.) These reasons are precisely why this paper will focus on rotation and controlled-rotation operators as elementary building blocks for synthesis of quantum circuits. A new concise and canonical data structure, called quantum decision diagrams or QDD's, will be introduced and subsequently used for conducting quantum operations and synthesizing quantum logic circuits. More precisely, the QDD's are designed to have the ability to express the functionality of every quantum circuit composed of controlled-rotation operators assuming that all rotations are about a single *axis* and a '*binary control signal'* constraint is enforced.

4. Previous Work on Synthesis of Quantum Logic Circuits

Reversible logic synthesis and quantum logic synthesis are closely related. However, for quantum circuits it is much more efficient to focus on logic synthesis with quantum gates. One method for quantum circuit synthesis is to decompose the corresponding unitary matrix of the circuit into unitary matrices of quantum gates, or alternatively, composing the matrices of elementary gates to achieve the unitary matrix of the circuit. Because for an n-input, n-output reversible circuit, size of the unitary matrix is $2^n \times 2^n$, this is not a practical method for synthesizing a general quantum circuit. Since dealing with quantum gates is so much more difficult than dealing with reversible binary gates, most researchers have been working on reversible logic synthesis using reversible binary gates. The synthesis of reversible circuits differs significantly from synthesis by using traditional irreversible gates. Several approaches for reversible logic circuit synthesis have been presented in [19]-[23]. These approaches resort to exhaustive combinatorial search or methods such as matrix decomposition, local transformations, spectral approaches, and on adaptations of EXOR logic decomposition, Reed-Muller representations, and other classical combinational circuit design methods.

Toffoli [24] provided an algorithm for implementing an arbitrary function with the "CNTS" library, comprising of controlled-NOT, NOT, Toffoli gate, and SWAP gate (see section 5). This implementation is however not optimized. Many other researchers have worked on reversible logic synthesis. Kerntopf [25] proposed exhaustive search methods to perform synthesis of small-scale circuits. In [26] a synthesis method based on manipulating the truth tables is presented. The algorithm produces a circuit composed of $n \times n$ Toffoli gates.[1] The method provided is a constructive approach based on the truth tables, which makes it computationally expensive and intractable for average and large circuits. In [27], Shende et al generate a library of small optimal circuits based on branch-and-bound and exploiting the property that any sub-circuit of an optimal circuit is itself optimal. This work does not provide a synthesis approach for a general logic and is limited to synthesizing reversible logic circuits with a small number of inputs and gates. Agrawal and Jha [28] presented a RM-expansion based technique for optimizing a circuit that is mapped to reversible gates. In [29] an algorithm for synthesis of quantum circuits using reversible Davio expansion was proposed. However these algorithms are intrinsically incapable of generating near optimal circuits and may require a large number of *temporary storage channels*, i.e., input-output wire pairs other than those on which the function is computed. In [30], Shende et al presented a top-down structure using

[1] An $n \times n$ Toffoli gate has n-1 control lines which pass through the gate unaltered and a target line on which the value is inverted if all the control lines have value '1'.

the Cosine-Sine decomposition and introduced and used the quantum multiplexer for recursive implementation of quantum gates. Group theory has also been employed as a tool to analyze reversible gates [31] and investigate generators of the group of reversible gates [32].

Few researchers have investigated the synthesis problem of quantum circuits by using quantum gates. In [33], Hung et al transform the synthesis problem into a satisfiability problem. They in fact use a SAT solver instead of employing an exhaustive search. This method is practical only for very small circuits since the reported run-time of the algorithm for optimal synthesis of a single-bit adder with 6 quantum gates is 7 hours on a 850MHz Pentium III processor running Linux. Other researchers have turned to evolutionary algorithms to reduce the CPU time [34]. However, applying evolutionary algorithms or similar techniques (such as simulated annealing and branch and bound) for solving a Boolean satisfiability problem does not help much with the quantum circuit synthesis task itself since these techniques can be applied to any combinatorial optimization problem and tend to only provide marginal improvement in terms of quality and runtime over semi-exhaustive or local neighborhood search methods.

It can be inferred that developing a practical synthesis algorithm for quantum circuits is extremely difficult because of the fast increase of data sizes. Indeed to-date there are no counterparts in quantum logic of such useful tools as algebraic decomposition methods, decision diagram based synthesis, or other standard logic synthesis techniques such as reduction to covering/coloring combinational approaches. In this paper we introduce an efficient data structure based on decision diagrams for representation, analysis and synthesis of quantum circuits and provide a synthesis approach based on the proposed decision diagrams.

5. Quantum Logic Synthesis with Rotation-based Quantum Operators

In this section, it will be shown that rotations and controlled-rotations around the *X* axis (i.e., $R_x(\theta)$ and controlled-$R_x(\theta)$) form a universal gate library. In this section, we will address the problem of automatically synthesizing a given Boolean function, *f*, by using $R_x(\theta)$ and controlled-$R_x(\theta)$ operators as the elementary operations (gate primitives.)

In a synthesized quantum circuit, the quantum states representing binary (basis states) values $\hat{0}$ and $\hat{1}$ will be:

$$\hat{0} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \hat{1} = R_x(\pi)\hat{0} = R_x(\pi)\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -i \end{bmatrix}.$$

With this definition of $\hat{0}$ and $\hat{1}$, the basis states remain orthogonal, and hence, they can be completely distinguished with proper quantum measurements. We adopt this definition because inversion from one basis state to the other is simply obtained by a π rotation around the *X* axis. At the end of this section, we will present another set of orthogonal basis states (i.e., $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$) as the basis binary values. [2]

With these assignments (i.e. $\hat{0}$ and $\hat{1}$ as the basis binary states,) the $R_x(\pi)$ operation acts as the quantum NOT gate (since $R_x(\pi)R_x(\pi) = R_x(2\pi) = I$.) Subsequently, the controlled-NOT (CNOT) gate can be described by using the controlled-$R_x(\pi)$ operator (cf. Figure 2(i).) In addition, the Toffoli gate, also known as the 3×3 Feynman gate or Controlled-Controlled-NOT gate, may be described by using the controlled-controlled-$R_x(\pi)$ operator (cf. Figure 2(ii).) Notice that the Boolean functions for each output

---

[2] Notice that the inversion from $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ to $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and vice versa, requires at least two rotation operators, a rotation around *X* axis and a rotation around *Z* axis [4].

of the CNOT and Toffoli gates are also shown in this figure, where '.' and '⊕' denote binary 'AND' and 'XOR' operators.

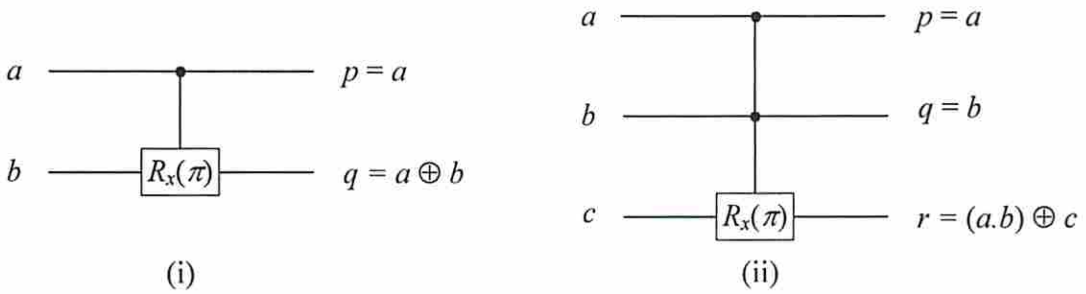

Figure 2. (i) CNOT gate (ii) Toffoli gate.

Toffoli [24] proved that NOT, CNOT and Toffoli gates are universal. Toffoli gate can be implemented using controlled-rotation operators as demonstrated in Figure 3. Therefore $R_x(\alpha)$ and controlled-$R_x(\alpha)$ operators are universal.

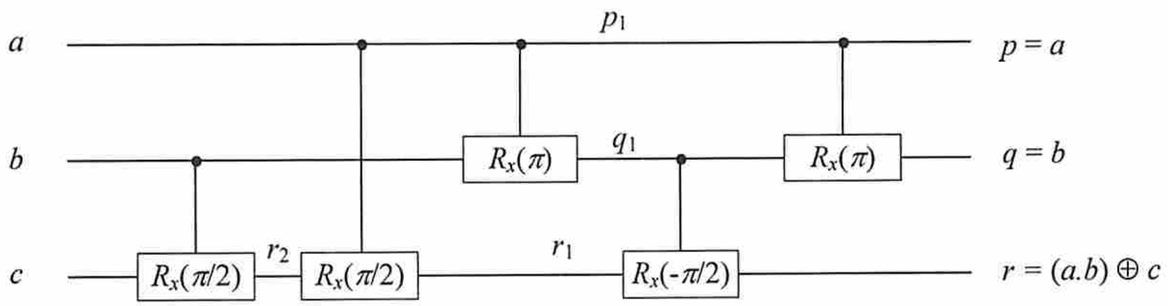

Figure 3. Synthesized Toffoli gate by using $R_x(\theta)$ and controlled-$R_x(\theta)$ operators.

In this paper, we focus on rotation-based quantum gates, which are directly realizable in quantum hardware [11][12]. In contrast, coarse-grained quantum gates (such as those in the CNTS library) may be used to synthesize an arbitrary quantum logic circuit. This disadvantage of the latter is that some of the basic gates in these libraries (e.g., the Toffoli and SWAP gates in the CNTS library) have complex realizations in quantum hardware. We believe working directly with the most primitive universal gates for quantum logic provides a higher degree of flexibility and freedom in synthesizing efficient quantum hardware, and thus, produces more efficient and compact hardware realization of quantum logic circuits. As an example, it was shown in [35] that, compared to CNTS-based realization, the implementation cost of realizing Fredkin [36] and Miller [37] gates is significantly lower when using even a restricted subset of the rotation-based operators (i.e., those corresponding to π/2, π, and −π/2 rotations.)

It is critical to point out that, for all ***input basis (binary) vectors***, control inputs of the controlled-$R_x(\theta)$ operators in the circuit of Figure 3 only take $\hat{0}$ or $\hat{1}$ values. This condition, which we shall refer to as the ***binary control signal constraint***, is set as a design constraint in the synthesis process. For the purpose of representing quantum logic circuits this constraint does not affect the expressive power and universality of $R_x(\theta)$ and controlled-$R_x(\theta)$ operators and has also been adopted by other researchers in the field (cf. [33][34].) (This constraint does not imply that a control signal can never adopt a superposition value, i.e., it may be possible that a control signal adopt a superposition value when (and only when) the inputs of the circuit are not binary. In the reminder of this paper whenever we constraint a variable to binary values we implicitly mean that when binary inputs are applied to the circuit that constraint is set.) Moreover, to the best of our knowledge, there is no evidence that relaxing this constraint, can improve the optimality of the synthesis result for quantum logic circuits.

5.1 Quantum Factored Forms

In any quantum circuit synthesized with binary control signal constraint, the first output, *p*, of any controlled-$R_x(\theta)$ operator is equal to the control input *a*. However, the second output depends on both inputs. We use the notation $q = aR_x(\theta)b$ to describe the second output *q*. With this new notation $R_x(\theta)$ can also be regarded as a two-operand operator with the following functionality: if $a = \hat{0}$, then $q = b$ else $q = R_x(\theta)b$. (The left operand, *a*, can only assume $\hat{0}$ or $\hat{1}$ values.)

Definition: *Quantum Factored Form.* $\hat{0}$ is a quantum factored form. Every variable is a quantum factored form. If *h* is a factored form, then $f = R_x(\theta)h$ is a quantum factored form. Moreover, if *g* and *h* are factored forms and *g* only takes $\hat{0}$ and $\hat{1}$ values, then $f = gR_x(\theta)h$ is a quantum factored form.

In a quantum circuit synthesized with $R_x(\theta)$ and controlled-$R_x(\theta)$ operators (with binary control signal constraint), any output (or internal signal) of the circuit can be described as a quantum factored form. For example, the output function *r* in Figure 3 may be described as:

$$r = [aR_x(\pi)b]R_x(-\pi/2)[aR_x(\pi/2)[bR_x(\pi/2)c]].$$

The following two commutative and associative relations are useful for manipulating quantum factored forms: $aR_x(\pi)b = bR_x(\pi)a$ and $aR_x(\theta_1)[bR_x(\theta_2)c] = bR_x(\theta_2)[aR_x(\theta_1)c]$.

A sub class of quantum factored forms is quantum cascade forms defined as follows.

Definition: *Quantum Cascade Form.* $\hat{0}$ is a quantum cascade form. Every variable is a quantum cascade form. If *h* is a cascade form and *v* is a variable not present in *h*, then $f = vR_x(\theta)h$ is a quantum cascade form. ($f = R_x(\theta)h$ is also considered a quantum cascade form.)

A general quantum cascade form is expressed as:

$$f = R_x(\theta_0)[v_1R_x(\theta_1)[v_2R_x(\theta_2) \dots [v_nR_x(\theta_n)\hat{0}]]]$$

Note that if $\theta_n = \pi$ then $v_nR_x(\theta_n)\hat{0} = v_n$. It can be verified that this cascade form expression can be rewritten as:

$$f = R_x(\theta_0)[v_{p_1}R_x(\theta_{p_1})[v_{p_2}R_x(\theta_{p_2}) \dots [v_{p_n}R_x(\theta_{p_n})\hat{0}]]]$$

Where $(p_1, p_2, \dots, p_n)$ is a permutation of $(1,2,\dots,n)$

The problem of realizing a function using $R_x(\theta)$ and controlled-$R_x(\theta)$ operators is equivalent to finding a quantum factored form for the function. To do this, we must first introduce a graph-based data structure in the form of a decision diagram for representing quantum logic functions.

5.2 Reduced Ordered Quantum Decision Diagrams (QDD)

The concept of BDD's was first proposed by Lee [38] and later developed by Akers [39] and then by Bryant [40], who introduced Reduced Ordered BDD's (ROBDD) and proved its canonicity property. Bryant also provided a set of operators for manipulating ROBDD's. From now on, we shall drop the prefix RO and simply use BDD to mean ROBDD. Using complement edges can further reduce the size of the BDD [41]. (A complement edge points to the complement of a BDD node. To maintain canonicity, complement edges should be assigned to only 0-edges (or only 1-edges.) Lai et al. [42] proposed Edge-Valued BDD's (EVBDD), which can represent and manipulate integer functions and can be used for functional decomposition.

In this section, we describe a new decision diagram for the representation of quantum functions. To the best of our knowledge, this is the first such canonical graph-based representation for quantum functions.

Definition: A QDD is a directed acyclic graph with three types of nodes: a single terminal node with value $\hat{0}$, a weighted root node, and a set of non-terminal (internal) nodes. Each internal node represents a quantum function. It is associated with a binary decision variable and has two outgoing edges: a weighted $\hat{1}$-edge (solid line) leading to another node (the $\hat{1}$-child) and a non-weighted $\hat{0}$-edge (dashed line) leading to another node (the $\hat{0}$-child.) The weights of the root node and $\hat{1}$-edges are in the form of $R_x(\theta)$ matrices. Since all the weights in a QDD are in the form of $R_x(\theta)$, the value θ is sufficient to specify the weight. We assume that $-\pi < \theta \leq \pi$. Furthermore, when the edge or root node weight is the identity matrix (i.e., $R_x(0) = I$), it will not be shown in the diagram.

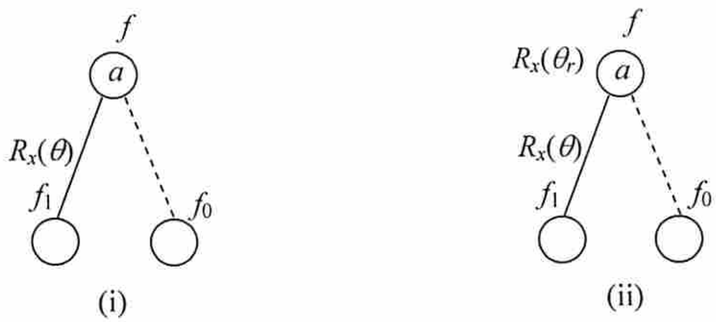

Figure 4. Structure of a QDD.

Figure 4(i) shows an internal node, f, in a QDD with decision variable, a, the corresponding $\hat{0}$ and $\hat{1}$ edges, and child nodes, f_0 and f_1. This relation between the QDD nodes in this figure is as follows. If $a = \hat{1}$, then $f = R_x(\theta) f_1$ else $f = f_0$. In addition, if f is the weighted root node of a QDD (cf. Figure 4(ii)), then the following relation holds. If $a = \hat{1}$, then $f = R_x(\theta_r) R_x(\theta) f_1 = R_x(\theta_r + \theta) f_1$ else $f = R_x(\theta_r) f_0$.

Similar to BDD's, in QDD's isomorphic sub-graphs (nodes with the same quantum function) are merged. Additionally, if the $\hat{0}$-child and the $\hat{1}$-child of a node are the same and the weight of the $\hat{1}$-edge is $R_x(0) = I$, then that node is eliminated. Using these two reduction rules and given a total ordering on input variables, the QDD will be uniquely constructed for a quantum function.

Consider a quantum function with n variables $f(v_1, v_2, ..., v_n)$. Each binary value assignment to the variables $v_1, v_2, ..., v_n$ corresponds to a path from the root to the terminal node of the QDD of f. Assuming the variable ordering $v_1 < v_2 < ... < v_n$, the corresponding path can be identified by a top-down traversal of the QDD starting from the root node. For each node that is visited during the traversal, we select the edge corresponding to the value of its decision variable v_i. (i.e., if $v_i = \hat{1}$ select the $\hat{1}$-edge; otherwise, select the $\hat{0}$-edge) and continue with the node at the end of the selected edge until the terminal node is visited. During such a traversal for every variable v_i, only one node with decision variable v_i will be visited specifying a path from the root to the terminal node with a total number of n-1 edges. Let's denote the weight of the root node by w_0 and the weight of the selected edges by $w_1, w_2, ..., w_{n-1}$. The value of the function f for assigned values to $v_1, v_2, ..., v_n$ is:

$$f(v_1, v_2, ..., v_n) = w_0 w_1 ... w_{n-1} \hat{0} = w_0 w_1 ... w_{n-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Clearly, if, during this graph traversal, a $\hat{0}$-edge is selected for variable v_i (i.e., if $v_i = \hat{0}$), then the corresponding edge weight will be $w_i = I$.

We have shown that QDD's provide a concise and canonical representation for a quantum function. Notice that QDD's can be regarded as an extension of BDD's i.e., each BDD can also be regarded as a QDD (A QDD is a BDD exactly if all the weights of the QDD are either $R_x(0) = I$ or $R_x(\pi)$.) As will be shown later, the synthesis process starts with the QDD of the given logic function (which is also a QDD) and decomposes the given QDD to realizable QDD's.

The QDD structure has some useful properties. One important property, i.e., the *linear topology property*, is demonstrated in Figure 5.

The idea is that when the $\hat{0}$-child and the $\hat{1}$-child of a node, f, are the same node, g, then that node can be directly realized by a controlled-$R_x(\theta)$ operator in terms of its child i.e., $f = aR_x(\theta)g$.

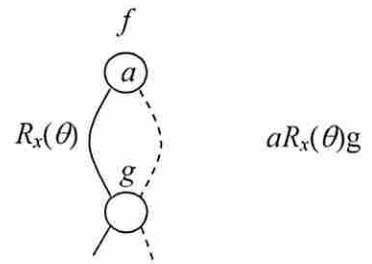

Figure 5. The linear topology property of a QDD.

As an example, Figure 6 shows the QDD's of functions p_1, q_1 and r_1 in Figure 3 .

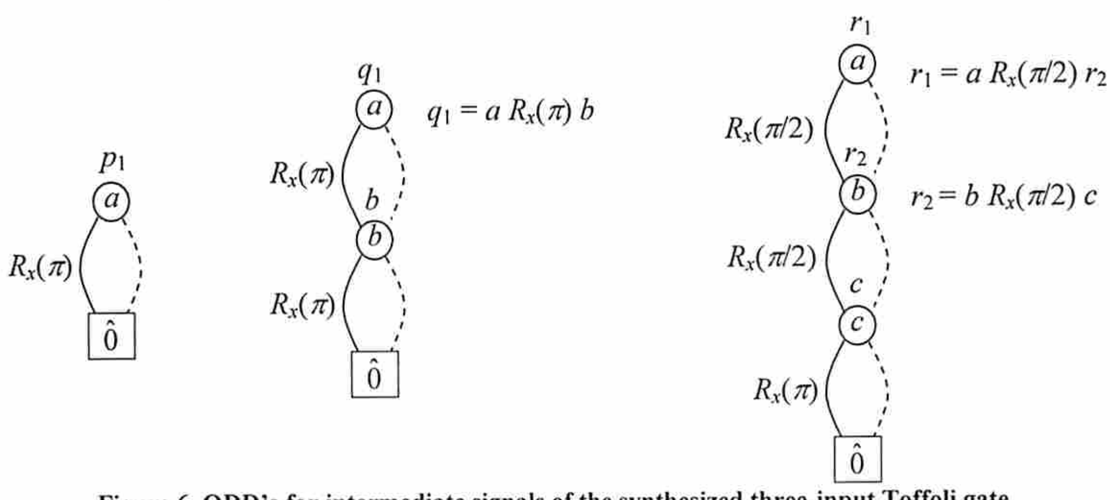

Figure 6. QDD's for intermediate signals of the synthesized three-input Toffoli gate.

The QDD's in Figure 6 are associated with functions that have a quantum cascade form representation. For example function r_1 can be represented as: $r_1 = aR_x(\pi/2)[bR_x(\pi/2)c]$ which is a cascade form. Generally every QDD with a chain structure (such as QDD's in Figure 6) is associated with a cascade form and can directly be realized with the rotation and controlled-rotation operators. This property will extensively be used in the synthesis algorithm.

5.3 The Quantum Apply Operation

In this section we explain how to apply rotation and controlled-rotation operators to QDD's. Suppose the QDD for a function, f, is given. The QDD for $h = R_x(\gamma)f$ can simply be obtained by multiplying the weight of the root node of f by $R_x(\gamma)$. To obtain $h = fR_x(\gamma)g$ for given QDD's f and g (assuming f only takes $\hat{0}$ and $\hat{1}$ values,) we use the *quantum apply operation* (q-apply) an extension to the apply operation first introduced by Bryant [40].

The q-apply is implemented by a recursive traversal of the two QDD operands. For each pair of nodes that are visited during the traversal, an internal node is added to the resultant QDD by utilizing the rules stated in Figure 7.

Consider performing q-apply to obtain $h = fR_x(\gamma)g$. q-apply takes two QDD nodes f and g as arguments and compares the corresponding decision variables of the nodes and adds a new node to the resulting QDD, h, using one of three rules after including the weights of the root node and $\hat{1}$-edge in the corresponding $\hat{1}$-child and $\hat{0}$-child as shown in Figure 7. Assume that the corresponding variables for QDD nodes f and g are a and b, respectively. The new node generated by q-apply depends on the variable ordering of a and b as demonstrated in Figure 7. For example, suppose that $a<b$. Rule 1 is invoked, generating a new node in the resulting QDD (h) containing variable a. Rule 1 directs the q-apply operation to recursively call itself.

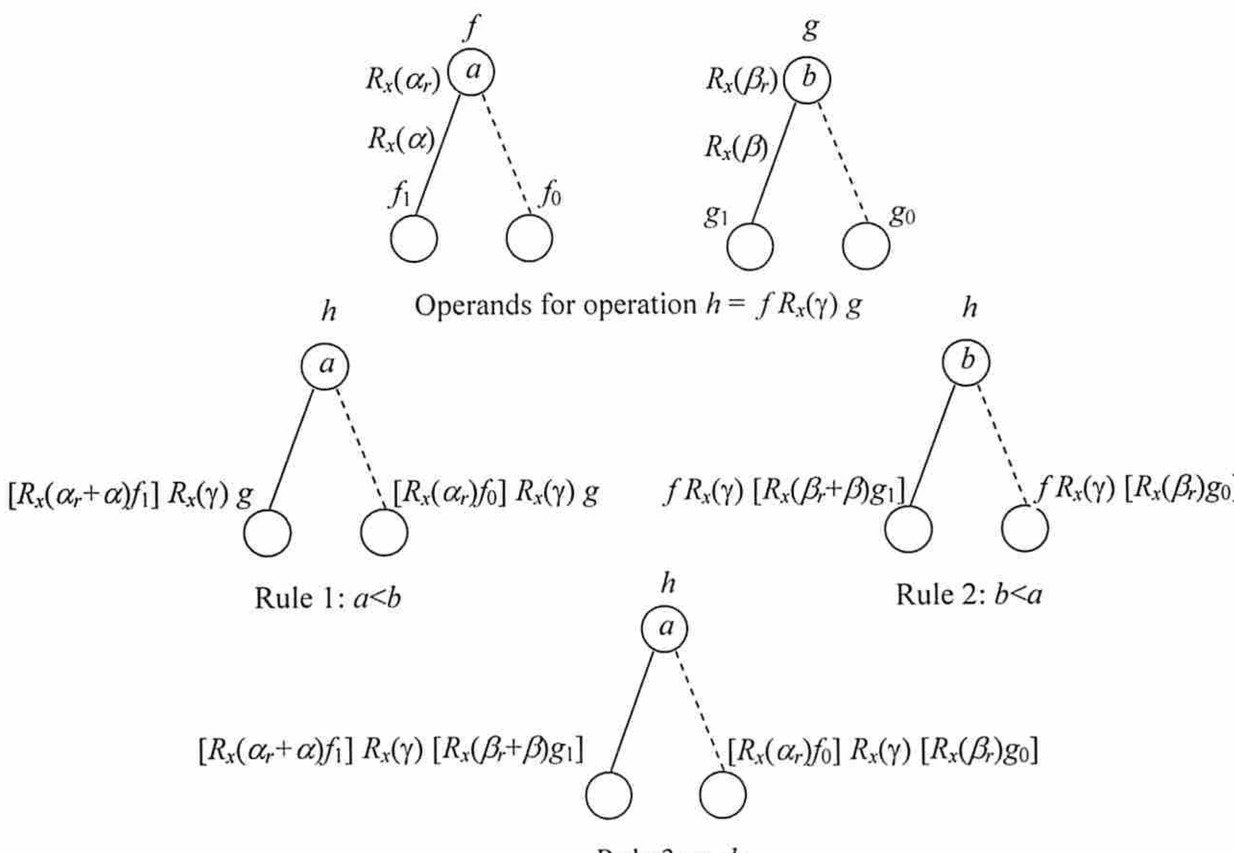

Figure 7. Rules for implementing the q-apply operator on two QDD's.

For terminal conditions the following relation are used: $\hat{0}R_x(\theta)v = v$ and $\hat{1}R_x(\theta)v = R_x(\theta)v$. Since *f* only assumes $\hat{0}$ and $\hat{1}$ values, these are the only possible terminal conditions.

After the recursive computation of $\hat{1}$-child and $\hat{0}$-child of *h*, in order to maintain the canonicity of the resulting QDD, isomorphic sub-graphs are merged and if the $\hat{0}$-child and the $\hat{1}$-child of a node are the same and the weight of the $\hat{1}$-edge is $R_x(0) = I$, then that node is eliminated. Also the resulting weights for the nodes ($\hat{1}$-child and $\hat{0}$-child of *h*) are modified as demonstrated in Figure 8 to make QDD of *h* canonical.

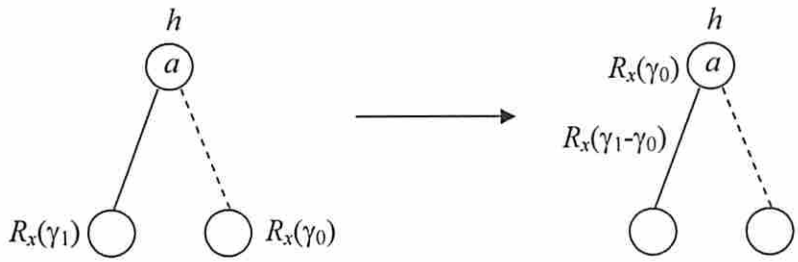

Figure 8. Weight modification during q-apply to maintain canonicity of the resulting QDD.

Figure 9 demonstrates the result of performing q-apply operation on q_1 and r_1 from Figure 6 to obtain $r=q_1R_x(-\pi/2)r_1$.

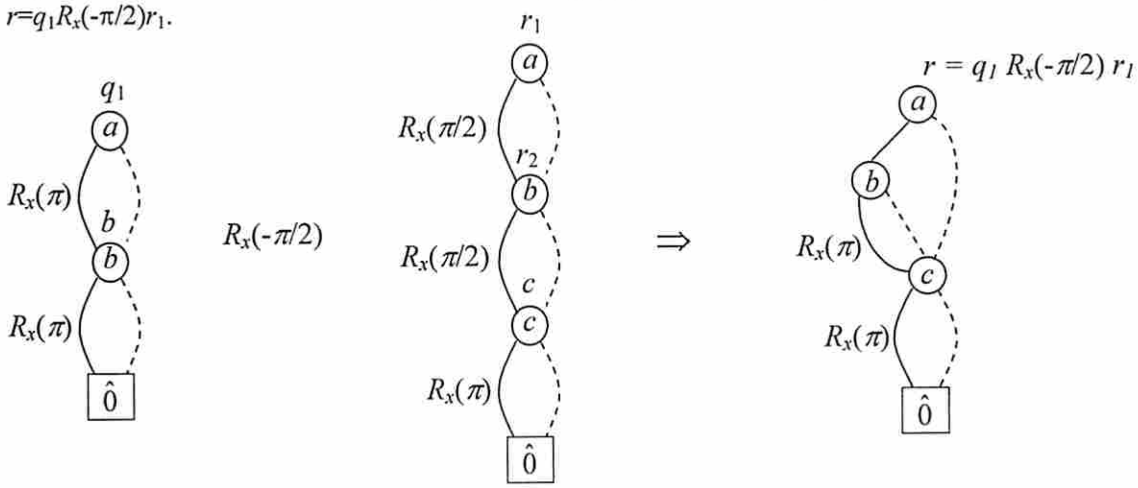

Figure 9. An example of performing q-apply on two QDD's.

It is noteworthy that the commutative property of matrix multiplication for $R_x(\theta)$ matrices is critical for the q-apply to generate the correct result i.e., performing q-apply as described may not generate the correct result for decision diagrams with weights that are not commutative.

5.4 QDD-based Functional Decomposition and the Notion of Q-Linearity

As mentioned earlier, the problem of realizing a function, *f*, using $R_x(\theta)$ and controlled-$R_x(\theta)$ operators is equivalent to finding a quantum factored form for the function, which can in turn be performed by recursive bi-decomposition of the given function *f*. Before delving into the details, we provide a brief review of prior work related to functional decomposition in general, and bi-decomposition in particular.

Multi-level logic synthesis based on algebraic optimization techniques [43] are commonplace. An alternative synthesis method is based on Boolean division and decomposition. Functional decomposition, systematically investigated by Ashenhurst [44] and Curtis [45], can be defined as expressing the function F(X) as F(X)=f(G(Y),Z) where Y∪Z=X. The decomposition methods provided in [44][45] and some other works are based on decomposition charts, which makes them computationally inefficient since the size of the chart grows exponentially with the number of variables. Therefore, BDD-based decomposition methods have been developed that use BDD's as platform to carry out functional decomposition. Lai et al. [46] used BDD's instead of decomposition charts to perform functional decompositions. Other approaches based on the technique provided in [46] have been reported in [47]-[49]. Reference [42] considered decomposition of multiple-output functions, where the multiple-output Boolean function is first transformed into an EVBDD and then decomposed by using methods developed in [46]. Bidecomposition [51], which is an important special case of functional decomposition, is a decomposition of type F(X)=G(Y)ΘH(Z) where Y∪Z=X and Θ stands for any logic operation. A class of quasi-algebraic decomposition, which in turn is a special case of bidecomposition, has been introduced in [52]. Files and Perkowski [53] used Multivalued Decision Diagrams (MDD) to perform multi-valued functional decomposition. Karplus [54] proposed a method which performs functional decomposition directly on BDD's. He introduced the concept of a 1- and 0-dominator and showed their relationship to algebraic AND/OR decomposition. Bertacco and Damiani [55] presented a method which performs recursive decomposition directly on a BDD. Stanion and Sechen [56] described a Boolean division and factorization method using a specialized BDD operator, called interval cofactor. Yang and Ciesielski [57] introduced the concepts of x-dominator and generalized x-dominator to perform XOR decomposition directly on BDD's.

Definition: Quantum bi-decomposition of *f* is defined as finding functions *g* and *h* and value γ such that $f = gR_x(\gamma)h$ where *g* only assumes values $\hat{0}$ and $\hat{1}$.

Next we provide an algorithm for quantum bi-composition which can be used to bi-decompose a given function f to $gR_x(\gamma)h$. Subsequently, *g* and *h* are recursively bi-decomposed, which will eventually result in a quantum factored for *f*. The bi-decomposition algorithm is based on the notion of *quantum linear (q-linear) variables.*

In the reminder of this paper, while expressing a function as $f(v_1, v_2, ..., v_n)$, it is implicitly assumed that *f* depends on all variables $v_1, v_2, ..., v_n$ (i.e. *f* is not invariant with respect to any variable among $v_1, v_2, ..., v_n$).

Definition: For a given function $f(v_1, v_2, ..., v_{i-1}, v_i, v_{i+1}, ..., v_n)$, variable v_i is '*q-linear*' if there exists a rotation value, θ_i, such that for every value assignment to $v_1, v_2, ..., v_{i-1}, v_{i+1}, ..., v_n$: $f_{v_i} = R_x(\theta_i) f_{\bar{v}_i}$, where we define $f_v = f(v_1, v_2, ..., v_{i-1}, \hat{1}, v_{i+1}, ..., v_n)$ and $f_{\bar{v}} = f(v_1, v_2, ..., v_{i-1}, \hat{0}, v_{i+1}, ..., v_n)$. A variable is called *q-nonlinear* if it is not q-linear.

Next we present a number of key results.

Lemma 1: Consider function $f(v_1, v_2, ..., v_n)$ with variable ordering $v_1 < v_2 < ... < v_n$. If (and only if) variables $v_{k+1}, v_{k+2}, ..., v_n$ are q-linear (i.e., for each v_i, $k+1 \leq i \leq n$, there is a θ_i that for all $v_1, v_2, ..., v_{i-1}, v_{i+1}, ..., v_n$ values, $f_{v_i} = R_x(\theta_i) f_{\bar{v}_i}$,) then for each variable v_i, $k+1 \leq i \leq n$, there is only one QDD node, n_i, with decision variable v_i. The weight of the $\hat{1}$-edge of n_i will be $R_x(\theta_i)$. Also no edge originating from nodes above n_j (i.e., nodes with decision variable v_j, $j<i$) will end at a node below n_i (i.e., a node with decision variable v_j, $j>i$.)

Proof: The proof is by induction on v_n, v_{n-1}, v_{n-2}, …, v_{k+1} starting from v_n. Details are straight-forward and omitted here. ■

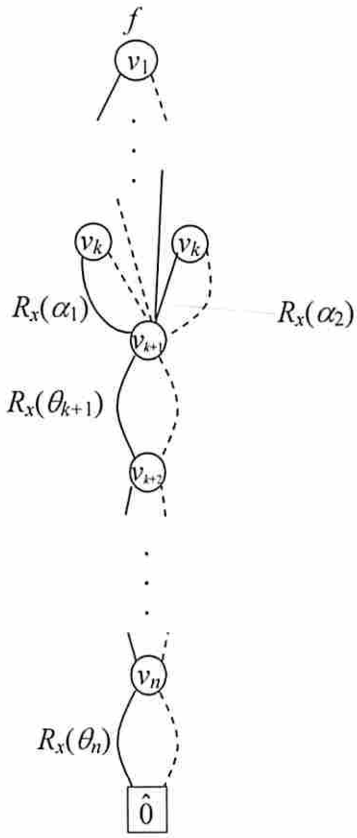

Figure 10. A general QDD structure with q-linear variables v_{k+1} … v_n.

Let v_k be the lowest indexed q-nonlinear variable after which $v_{k+1}, v_{k+2}, \ldots, v_n$ are q-linear variables of *f*. From Lemma 1, $f_{v_j} = R_x(\theta_j) f_{\bar{v}_j}$, $k+1 \leq j \leq n$ where θ_j is fixed independent of the input combination of $v_1, v_2, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n$. As shown in

Figure 10, every path from the root node of the QDD to its terminal node will either go thru an internal node with decision variable v_k or it will skip any such node and directly go the single QDD node with decision variable v_{k+1}. For the latter case, $f_{v_k} = R_x(0) f_{\bar{v}_k} = f_{\bar{v}_k}$ and for any former case, $f_{v_k} = R_x(\alpha_i) f_{\bar{v}_k}$ where there will be as many different rotation angles (e.g., α_1, α_2) for variable v_k as there are internal nodes with decision variable v_k in the QDD.

Definition: The *degree of q-nonlinearity* of variable v_k is *m*-1 where *m* denotes the number of different rotation angles α_i (including 0 if any) that $f_{v_k} = R_x(\alpha_i) f_{\bar{v}_k}$ for some $v_1, v_2, \ldots, v_{k-1}, v_{k+1}, \ldots, v_n$. For q-linear variables the degree of q-nonlinearity is zero.

Lemma 2: Let m' denote the number of internal node with decision variable v_k, then if all paths from the root node of the QDD to its terminal node go thru an internal node with decision variable v_k, the degree of q-nonlinearity of variable v_k will be equal to $m'-1$ otherwise (if there is a path that skips any node with decision variable v_k,) the degree of q-nonlinearity of variable v_k will be equal to m'.

Proof: The proof follows from structural properties of QDD's and the definition of q-nonlinearity. Details are straight-forward and omitted here. ■

Theorem 1: Consider function $f(v_1, v_2, ..., v_n)$ with variable ordering $v_1 < v_2 < ... < v_n$. Assume that:

- $v_{k+1}, v_{k+2}, ..., v_n$ are q-linear variables of f.
- v_k is a q-nonlinear variable of f with degree of q-nonlinearity m-1:

  For each value assignment to variables $v_1, v_2, ..., v_{k-1}, v_{k+1}, ..., v_n$ exactly one of the following m relations holds:

  $$f_{v_k} = R_x(\alpha_1) f_{\bar{v}_k}, \quad f_{v_k} = R_x(\alpha_2) f_{\bar{v}_k}, \quad \ldots \quad , f_{v_k} = R_x(\alpha_m) f_{\bar{v}_k}.$$

- Let function g be defined as follows:

  If $f_{v_k} = R_x(\alpha_1) f_{\bar{v}_k}$ then $g = \hat{1}$ else $g = \hat{0}$.

Then:

- Function f can be bi-decomposed as: $f = g_1 R_x(\gamma) h$ where:

  $$g_1 = v_k R_x(\pi) g, \qquad \gamma = (\alpha_2 - \alpha_1)/2, \qquad h = g_1 R_x(-\gamma) f.$$

- g_1 will be a function of $v_1, v_2, ..., v_k$ (i.e. g_1 will be invariant of $v_{k+1}, v_{k+2}, ..., v_n$) and v_k will be q-linear in function g_1.
- h in will be a function of $v_1, v_2, ..., v_n$ and $v_{k+1}, v_{k+2}, ..., v_n$ will be q-linear in function h.
- Degree of q-nonlinearity of v_k in h will be less than or equal to m-2.

Proof: First we prove that function g is invariant with respect to $v_{k+1}, v_{k+2}, ..., v_n$ i.e., $g_{v_i} = g_{\bar{v}_i}$ for $k+1 \le i \le n$.

It follows from the definition of g that:

$$\text{If } f_{v_i v_k} = R_x(\alpha_1) f_{v_i \bar{v}_k} \text{ then } g_{v_i} = \hat{1} \text{ else } g_{v_i} = \hat{0}.$$

$$\text{If } f_{\bar{v}_i v_k} = R_x(\alpha_1) f_{\bar{v}_i \bar{v}_k} \text{ then } g_{\bar{v}_i} = \hat{1} \text{ else } g_{\bar{v}_i} = \hat{0}.$$

Since v_i is q-linear, there exists a θ_i such that for all $v_1, v_2, ..., v_{i-1}, v_{i+1}, ..., v_n$ values, $f_{v_i} = R_x(\theta_i) f_{\bar{v}_i}$ resulting in:

$$f_{v_i v_k} = R_x(\theta_i) f_{\bar{v}_i v_k}$$

$$f_{v_i \bar{v}_k} = R_x(\theta_i) f_{\bar{v}_i \bar{v}_k}.$$

By combining these relations, we have:

$$f_{v_i v_k} = R_x(\alpha_1) f_{v_i \bar{v}_k} \quad \Leftrightarrow \quad R_x(\theta_i) f_{\bar{v}_i v_k} = R_x(\alpha_1 + \theta_i) f_{\bar{v}_i \bar{v}_k} \quad \Leftrightarrow \quad f_{\bar{v}_i v_k} = R_x(\alpha_1) f_{\bar{v}_i \bar{v}_k}$$

which proves that $g_{v_i} = g_{\bar{v}_i}$.

Since $g_1 = v_k R_x(\pi) g$, g_1 will also be invariant with respect to $v_{k+1}, v_{k+2}, ..., v_n$. Moreover $g_{1v_k} = R_x(\pi) g$ and $g_{1\bar{v}_k} = g$ which results in: $g_{1v_k} = R_x(\pi) g_{1\bar{v}_k}$ i.e., v_k in g_1 is q-linear.

Now we prove that the degree of q-nonlinearity of v_k in $h = g_1 R_x(-\gamma) f$ is less than or equal to m-2. For each value assignment to variables $v_1, v_2, \ldots, v_{k-1}, v_{k+1}, \ldots, v_n$ exactly one of the following m relations holds:

$$f_{v_k} = R_x(\alpha_1) f_{\bar{v}_k}, \quad f_{v_k} = R_x(\alpha_2) f_{\bar{v}_k}, \quad \ldots \quad , f_{v_k} = R_x(\alpha_m) f_{\bar{v}_k}.$$

For each of the above cases, we examine the relation between h_{vk} and $h_{\bar{v}k}$.

(1) If $f_{v_k} = R_x(\alpha_1) f_{\bar{v}_k}$ then by definition $g = \hat{1}$ and these relations will follow:

$$h_{\bar{v}k} = [\hat{0} R_x(\pi) \hat{1}] R_x(-\gamma) f_{\bar{v}k} = R_x(-\gamma) f_{\bar{v}k} \Rightarrow f_{\bar{v}k} = R_x(\gamma) h_{\bar{v}k}$$

$$h_{vk} = [\hat{1} R_x(\pi) \hat{1}] R_x(-\gamma) f_{vk} = f_{vk} = R_x(\alpha_1) f_{\bar{v}k} = R_x(\alpha_1 + \gamma) h_{\bar{v}k}$$

$$h_{vk} = R_x(\frac{\alpha_1 + \alpha_2}{2}) h_{\bar{v}k}$$

(2) If $f_{v_k} = R_x(\alpha_2) f_{\bar{v}_k}$ then by definition $g = \hat{0}$ and the following relations will hold:

$$h_{\bar{v}k} = [\hat{0} R_x(\pi) \hat{0}] R_x(-\gamma) f_{\bar{v}k} = f_{\bar{v}k}$$

$$h_{vk} = [\hat{1} R_x(\pi) \hat{0}] R_x(-\gamma) f_{vk} = R_x(-\gamma) f_{vk} = R_x(-\gamma + \alpha_2) f_{\bar{v}k}$$

$$h_{vk} = R_x(\frac{\alpha_1 + \alpha_2}{2}) h_{\bar{v}k}$$

(3,4,…,m) For these m-2 cases, $f_{v_k} = R_x(\alpha_i) f_{\bar{v}_k}$, $3 \leq i \leq m$, by definition $g = \hat{0}$ and $h_{vk} = R_x(-\gamma + \alpha_i) h_{\bar{v}k}$.

The first two cases, (1,2), result in the same relation between h_{vk} and $h_{\bar{v}k}$: $h_{vk} = R_x(\frac{\alpha_1 + \alpha_2}{2}) h_{\bar{v}k}$. The remaining m-2 cases, (3,4,…,m), at most result in m-2 different relations between h_{vk} and $h_{\bar{v}k}$. Therefore, the total number of different relations between h_{vk} and $h_{\bar{v}k}$ i.e., the degree of q-nonlinearity of v_k in h, is less than or equal to m-2.

Finally, from $h = g_1 R_x(-\gamma) f$ it follows that:

$$g_1 R_x(\gamma) h = g_1 R_x(\gamma) [g_1 R_x(-\gamma) f] = f$$

(The second equality can be verified by checking for both cases: $g = \hat{1}$ and $g = \hat{0}$.)

Hence, with this definition of h and g_1, function f can be bi-decomposed as : $f = g_1 R_x(\gamma) h$. ■

Using the proposed bi-decomposition approach f can be bi-decomposed into $f = g_1 R_x(\gamma) h$, where g_1 and h are themselves recursively bi-decomposed until a quantum factored form is obtained.

Since g_1 is invariant of $v_{k+1}, v_{k+2}, \ldots, v_n$ and v_k in g is q-linear and degree of q-non linearity of v_k in h is at most m-2, the recursion will finally stop at terminal cases where g_1 and/or h have directly realizable QDD's, i.e., all the variables will be q-linear in the functions and hence they will have cascade forms corresponding to QDD's with a chain structure similar to QDD's in Figure 6. As a result of Lemma 1, in a function with chain structured QDD, all variables are q-linear.

The algorithm, *q-factor(f)*, uses the recursive bi-decomposition in Theorem 1 to generate a quantum factored form for a function *f*.

Algorithm: *q-factor* (f)

0- If all variables are q-linear then return the corresponding cascade form for f .

1- Find the lowest indexed q-nonlinear variable, v_k , after which $v_{k+1}, v_{k+2}, ..., v_n$ are q-linear.

2- Bi-decompose f as $f = g_1 R_x(\gamma)h$ where g_1, h and γ are given in Theorem 1 using v_k .

3- Return [*q-factor*(g_1)] $R_x(\gamma)$ [*q-factor*(h)] .

It is important to notice that all of the above steps can be directly performed on QDD's. For example if the QDD of a function, f, is a chain structure, there exists a cascade form for f (step 0). For step 1, as depicted in Figure 10, according to Lemma 1, identifying v_k is equivalent to identify the lower chain-structure part of the QDD. As for step 2, according to Lemma 2, the values $\alpha_1, \alpha_2, ..., \alpha_m$ can be obtained from the weights of the $\hat{1}$-edges of nodes with decision variable v_k . Hence, $\gamma = (\alpha_2 - \alpha_1)/2$ can also be obtained. Let n_i denote the node with decision variable v_k and $\hat{1}$-edges weight $R_x(\alpha_i)$. The QDD of g_1 can be constructed from QDD of f using the following method. Starting from the QDD of f:

- Change all the weights to $R_x(0) = I$.
- Create a DQQ node, v_k, representing v_k. as depicted in Figure 11.
- Redirect all edges toward n_1 to node v_k and make the weight of all such edges $R_x(\pi)$.
- Redirect all edges toward $n_2, n_3, \ldots, n_m$ to node v_k and make the weight of all such edges $R_x(0)$.
- Discard nodes $n_1, n_2, \ldots, n_m$.
- Merge isomorphic sub-graphs, eliminate nodes with same $\hat{0}$-child and the $\hat{1}$-child if the weight of the $\hat{1}$-edge is $R_x(0) = I$, and update weights of the QDD to make the QDD of g_1 canonical. Having the QDD's for g_1 and f, the QDD of $h = g_1 R_x(-\gamma) f$ can be obtained using the q-apply operation.

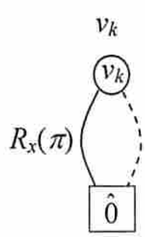

Figure 11. QDD for the node v_k.

The final factored form resulting from q-apply will be in the following form:

$$f = g_1 R_x(\gamma_1)\left[g_2 R_x(\gamma_2)\left[g_3 R_x(\gamma_3) \ldots \left[g_k R_x(\gamma_k)\hat{0}\right]\right]\right]$$

which can also be rewritten as:

$$f = g_{p_1} R_x(\gamma_{p_1})\left[g_{p_2} R_x(\gamma_{p_2})\left[g_{p_3} R_x(\gamma_{p_3}) \ldots \left[g_{p_k} R_x(\gamma_{p_k})\hat{0}\right]\right]\right]$$

where $(p_1, p_2, \ldots, p_k)$ is a permutation of $(1,2,\ldots,k)$. (Notice that g_i functions should be decomposed as well using q-apply.) In the following example, it is shown that different permutations on $(1,2,\ldots,k)$ may result in different number of gates while synthesizing the circuit.

Example 1: In this part a four-input Toffoli gate, depicted in Figure 12 (i), will be synthesized by using the *q-factor* algorithm. Figure 12 (ii) shows the QDD of the output *s* of the Toffoli gate. Throughout the synthesis process we maintain the variable ordering $a<b<c<d$.

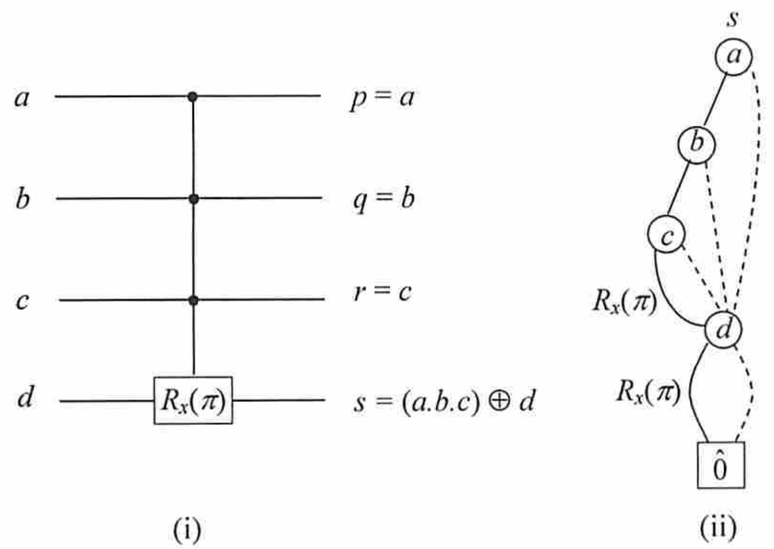

Figure 12. Four-input Toffoli gate and the QDD for 4th output, *s*.

Comparing the QDD of *s* with the general QDD structre in Figure 10, variable *c* will correspond to v_k and the degree of q-nonlinearity of *c* is 1. Also $\alpha_1 = 0$ and $\alpha_2 = \pi$ which results in $\gamma = \pi/2$. (It would also be correct to set $\alpha_1 = \pi$ and $\alpha_2 = 0$. This will generate a different circuit but with the same functionality.)

Consequently, function *s* can be bi-decomposed as: $s = g_1 R_x(-\pi/2)h$ where $g_1 = cR_x(\pi)g$. Now the QDD's for *g* and g_1 are depicted in Figure 13 (i) and (ii). It is seen that function g_1 is a 3-input Toffoli gate, which is synthesized as in Figure 3. As for function *h*, it can be derived as $h = g_1 R_x(\pi/2)s$. The QDD for *h* is depicted in Figure 13 (iii). Subsequently, *h* can be bi-decomposed as $h = g_2 R_x(-\pi/2)h_1$ where $g_2 = aR_x(\pi)b$ and $h_1 = g_2 R_x(\pi/2)h$. The resulting QDD for g_2 and h_1 are shown in Figure 13 (iv) and (v). The resulting factored form for *s* is: $s = g_1 R_x(-\pi/2)[g_2 R_x(-\pi/2)h_1]$.

Due to the chain structure of g_2 and h_1, they may be directly realized by using controlled-rotation operators. Notice that when realizing g_1, we will also implement g_2. As a result, it is more efficient to construct s as: $s = g_2 R_x(-\pi/2)[g_1 R_x(-\pi/2)h_1]$. The resulting quantum circuit realization is depicted in Figure 14.

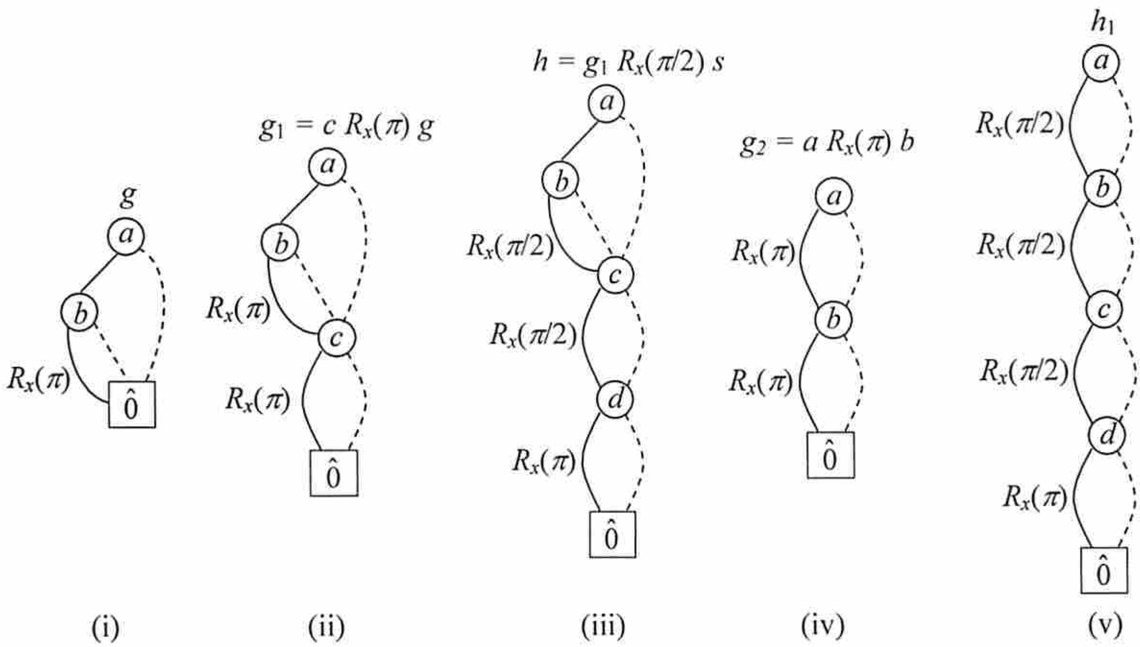

Figure 13. QDD's needed to synthesize the four-input Toffoli gate.

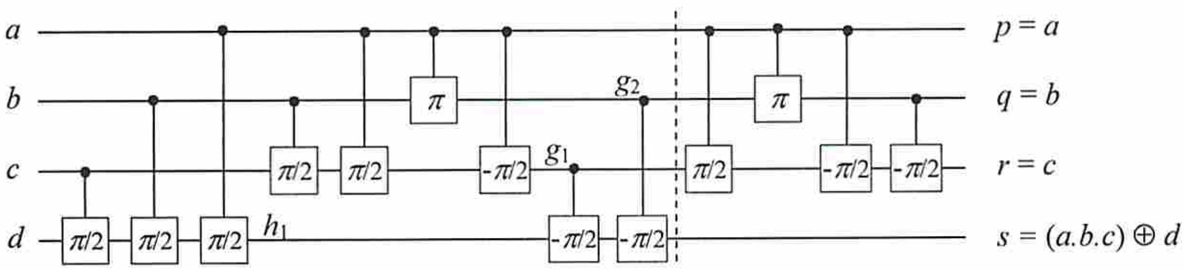

Figure 14. Automatic synthesis solution for the four-input Toffoli gate obtained by the q-factor algorithm.

In this figure only the angle of rotation is shown for controlled-rotation operators. The first part of the circuit (left of the dashed line) generates output s whereas the second part generates outputs p, q and r. This realization of the 4-input Toffoli gate can be generalized for n-input Toffoli gates. In [58] a method for synthesizing an n-input Toffoli gate (including 4-input) is provided which is similar to the synthesis result provided in this paper. However the approach in [58] is specialized for gates similar to n-input Toffoli gates while our approach automatically and without assuming any prior knowledge above the function, synthesizes the circuit.

Example 2: Consider a full adder with inputs x_1, x_2 and x_3 and outputs s (sum) and c (carry out): $s = x_1 \oplus x_2 \oplus x_3$ and $c = (x_1.x_2) + (x_1.x_3) + (x_2.x_3)$ where '+' is binary 'OR' operation. The QDD's of s and c are shown in Figure 15 (i),(ii).

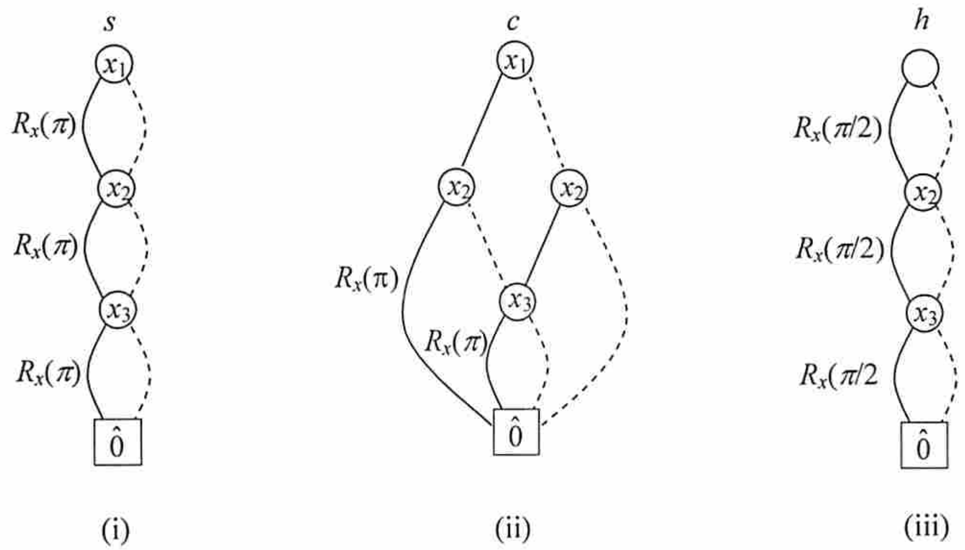

Figure 15. QDD's for a full adder.

As can be seen, the QDD of s is a chain structure which corresponds to a quantum cascade form and is directly realizable. The QDD of c however, needs to be recursively decomposed using q-factor algorithm. Using q-factor algorithm c is bi-decomposed as $c = g_1 R_x(-\pi/2)h$ where g_1 happens to be equal to s. The QDD for $h = sR_x(\pi/2)c$ is shown in Figure 15 (iii). As can be seen the QDD for h is a chain structure and can be directly realized. The resulting quantum circuit realization is depicted in Figure 16.

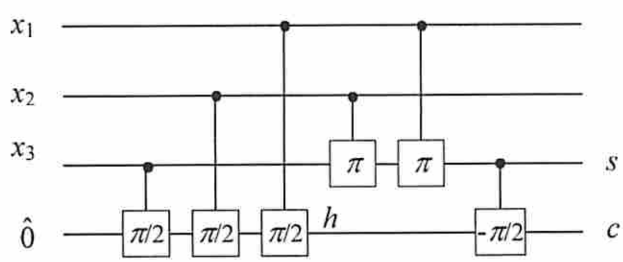

Figure 16. Quantum full adder.

The authors of [33] reported the run-time of the algorithm for optimal synthesis of a single-bit adder with 6 quantum gates as 7 hours on a 850MHz Pentium III processor running Linux. As we can observe, our method results in a circuit with the same number of quantum gates in virtually no time.

In order to create the quantum circuit from the quantum factored form generated by q-apply, in general there might be need to use the quantum fanout gate (none was required for examples provided in this paper.) Figure 17, shows the implementation of the quantum fanout gate using a controlled-rotation operator.

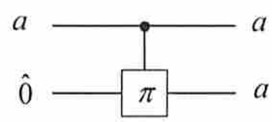

Figure 17. Quantum fanout gate

Also in some implementations of quantum logic, the binary basis values are $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. If there is need to design a circuit that perform on these basis values instead of $\hat{0} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\hat{1} = \begin{bmatrix} 0 \\ -i \end{bmatrix}$, the operator $C = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}$ can be used to transform the $|0\rangle$ and $|1\rangle$ states to $\hat{0}$ and $\hat{1}$ states and operator $C^{+} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ can be used to perform the reverse transformation. Hence to convert a circuit that works on $\hat{0}$ and $\hat{1}$ values to a circuit that performs on $|0\rangle$ and $|1\rangle$ as basis values the configuration in Figure 18 can be used. Notice that C and C^{+} are special cases of rotation operators around the Z axis [4].

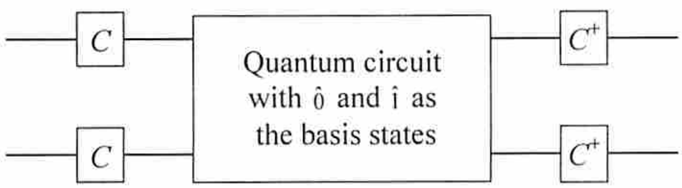

Figure 18. A quantum circuit with $|0\rangle$ and $|1\rangle$ as the basis states.

6. Conclusions

Quantum nano-devices are expected to play important roles in future information technology and multidisciplinary nanotechnology. This is because nano-scale dissipative quantum switches such as single electron transistors provide opportunities to realize Boolean logic circuits at smallest possible power-delay product values near the quantum limit with high circuit densities beyond limits of silicon roadmap devices. Nanotechnology can even realize non-dissipative quantum circuits such as in places where quantum coherence, extending over entire circuit operation, realizes massively parallel and highly sophisticated quantum computation/processing.

In this paper an efficient analysis and synthesis framework for quantum logic circuits was presented. We introduced the quantum factored forms, and developed a canonical and concise representation of quantum logic circuits. The focus of our approach was on the most basic quantum operators, i.e., the rotation and controlled-rotation primitives. Subsequently, an effective QDD-based algorithm (q-factor) for automatic synthesis of quantum logic circuits was introduced. Comparing the synthesis results for examples provided in this paper with previous approaches demonstrates the effectiveness and promise of the proposed approach.

References

[1] http://public.itrs.net/Files/2003ITRS/Home2003.htm.

[2] R. P. Feynman, "Simulating Physics with Computers," International Journal of Theoretical Physics, 21, 1982, pp. 467-488.

[3] D. Deutsch, "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer," Royal Society, A, 400, 1985, pp. 97-117.

[4] M. A. Nielsen, I. L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.

[5] C. P. Williams, S. H. Clearwater, Explorations in Quantum Computing, Springer-Verlag, 1998.

[6] M. Hirvensalo, Quantum Computing, Springer Verlag, 2001.

[7] R. Landauer, "Irreversibility and Heat Generation in the Computational Process," IBM Journal of Research and Development, 5, 1961, pp. 183-191.

[8] R. Keyes, R. Landauer, "Minimal Energy Dissipation in Logic," IBM Journal of Research and Development, 14, 1970, pp. 152-157.

[9] C. Bennett, "Logical Reversibility of Computation," IBM Journal of Research and Development, 17, 1973, pp. 525-532.

[10] C. Bennett, R. Landauer, "The Fundamental Physical Limits of Computation," Scientific American, 1985, pp. 48-56.

[11] J. I. Cirac, P. Zoller, "Quantum Computation with Cold Trapped Ions," Physical Review, 74, Issue 20, 1995, pp. 4091-4094.

[12] C. Monroe, D. Leibfried, B. E. King, D.M. Meekhof, W. M. Itano, D.J. Wineland, "Simplified Quantum Logic with Trapped Ions,". Physical Review A, 55, Issue 4, 1997, pp. 2489-2491.

[13] D. P. Di Vincenzo, "Quantum Computation," Science, 270, 1995, pp. 255-256.

[14] A. Ekert, R. Jozsa, "Quantum Computation and Shor's Factoring Algorithm," Review of Modern Physics, 68, Issue 3, 1996, pp. 733-753.

[15] I. L. Chuang, R. Laflamme, Y. Yamamoto, "Decoherence and a Simple Quantum Computer," Quantum Coherence and Decoherence, 1996, pp. 299-302.

[16] C. Vieri, M. J. Ammer, M. Frank, N. Margolus, T. A. Knight, Fully Reversible Asymptotically Zero Energy Microprocessor, MIT Artificial Intelligence Laboratory, Cambridge, MA 02139, USA.

[17] P. A. M. Dirac, The Principles of Quantum Mechanics, Oxford University Press, 1st Edition, 1930.

[18] J. von Neumann, Mathematical Foundations of Quantum Mechanics, Princeton Univ. Press, 1950.

[19] K. Iwama, Y. Kambayashi, S. Yamashita, "Transformation Rules for Designing CNOT-Based Quantum Circuits," Design Automation Conference, 2002, pp. 419-424.

[20] A. Khlopotine, M. Perkowski, P. Kerntopf, "Reversible Logic Synthesis by Iterative Compositions," International Workshop on Logic Synthesis, 2002, pp. 261-266.

[21] D. M. Miller, "Spectral and Two-Place Decomposition Techniques in Reversible Logic," Midwest Symposium on Circuits and Systems, Published on CD-ROM, 2002.

[22] A. Mishchenko, M. Perkowski, "Logic Synthesis of Reversible Wave Cascades," International Workshop on Logic Synthesis, 2002, pp. 197-202.

[23] M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Joswiak, A. Coppola, B. Massey, "Regularity and Symmetry as a Base for Efficient Realization of Reversible Logic Circuits," International Workshop on Logic Synthesis, 2001, pp. 90-95.

[24] T. Toffoli, Reversible Computing, Lab. for Computer Science, MIT, Cambridge, MA, Technical Memo. MIT/LCS/TM-151, 1980.

[25] P. Kerntopf, "A Comparison of Logical Efficiency of Reversible and Conventional Gates," International Workshop Logic Synthesis, 2000, pp. 261-269.

[26] D. M. Miller, D. Maslov, G. W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," Design Automation Conference, 2003, pp. 318-323.

[27] V. V. Shende, A. K. Prasad, I. L. Markov, J. P. Hayes, "Synthesis of Reversible Logic Circuits," IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 22(6), 2003, pp. 710-722.

[28] A. Agrawal, N. K. Jha, "Synthesis of Reversible Logic," Design Automation and Test in Europe, 2004, pp. 21384-21385.

[29] A. Al-Rabadi, "Quantum Circuit Synthesis Using Classes of GF(3) Reversible Fast Spectral Transforms," International Symposium on Multi Valued Logic, 2004, pp. 87-93.

[30] V. V. Shende, S. S. Bullock, I. L. Markov, "Synthesis of Quantum Logic Circuits," Asia and South Pacific Design Automation Conference, 2005, pp. 272-275.

[31] L. Storme et al., "Group Theoretical Aspects of Reversible Logic Gates," Journal of Universal Computer Science 5, 1999, pp 307-321.

[32] A. De Vos et al., "Generating the Group of Reversible Logic Gates," Journal of Physics A: Mathematical and General, vol. 35, 2002, pp. 7063-7078.

[33] W. Hung, X. Song, G. Yang, J.Yang, M. Perkowski, "Quantum Logic Synthesis by Symbolic Reachability Analysis," Design Automation Conference, 2004, pp. 838-841.

[34] M. Lukac, M. Perkowski, H. Mikhail Pivtoraiko, C. Hyo Yu, K. Chung, H. Jee, B. Kim, Y. Kim, "Evolutionary Approach to Quantum and Reversible Circuits Synthesis," Artificial Intelligence in Logic Design, Kluwer Academic Publisher, 2004, pp. 361-417.

[35] J. A. Smolin, D. P. DiVincenzo, "Five Two-Bit Quantum Gates are Sufficient to Implement the Quantum Fredkin Gate," Physical Review A, 53, 1996, pp. 2855-2856.

[36] E. Fredkin, T. Toffoli, "Conservative Logic.," Int. J. of Theoretical Physics (21): 219–253, 1982.

[37] D. M. Miller, "Spectral and Two-Place Decomposition Techniques in Reversible Logic," Proc. Midwest Symposium on Circuits and Systems, on CD-ROM, August 2002.

[38] C. Y. Lee, "Representation of Switching Circuits by Binary Decision Programs," Bell System Technical Journal, vol. 38, no. 4, 1959, pp. 985-999.

[39] S. B. Akers, "Functional Testing with Binary Decision Diagrams," Annual Conference of Fault-Tolerant Computing, 1978, pp. 75-82.

[40] R. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," IEEE Transactions on Computers, vol. 35, 1986, pp. 677-691.

[41] K. Brace, R. Rudell, and R. Bryant, "Efficient Implementation of a BDD Package," Design Automation Conference, 1990, pp. 40-45.

[42] Y.-T. Lai, M. Pedram, and S. Vrudhula, "EVBDD-Based Algorithms for Integer Linear Programming, Spectral Transformation, and Function Decomposition," IEEE Transactions on Computer-Aided Design, vol. 8, 1994, pp. 959-975.

[43] R. K. Brayton, G. D. Hachtel, and A. Sangiovanni-Vincentelli, "Multilevel Logic Synthesis," Proceedings of the IEEE , Vol. 78 , Issue 2, 1990, pp. 264-300.

[44] R. L. Ashenhurst, "The decomposition of switching functions," in Proc. Int. Symp. Theory of Switching, vol. XXIX, Ann. Computation Lab. Harvard Univ., Cambridge, MA, 1959, pp. 74–116.

[45] H. A. Curtis, "A New Approach to the Design of Switching Circuits," Boston, MA: D. Van Nostrand, 1962.

[46] Y-T. Lai, M. Pedram, and S. Vrudhula, "BDD-based decomposition of logic for functions with applications to FPGA synthesis," in Proc. Design Automation Conf., 1993, pp. 642–647.

[47] T. Sasao, "FPGA Design by Generalized Functional Decomposition," in Logic Synthesis and Optimization. Boston, MA: Kluwer, 1993.

[48] Y.-T. Lai, K.-R. Pan, and M. Pedram, "OBDD-based function decomposition: Algorithms and implementattion," IEEE Trans. Computer-Aided Design, vol. 15, pp. 977–990, Aug. 1996.

[49] S.-C. Chang, M. Marek-Sadowska, and T. Hwang, "Technology mapping for TLI FPGA's based on decomposition of binary decision diagrams," IEEE Trans. Computer-Aided Design, vol. 15, pp. 1226–1235, Oct. 1996.

[50] Y.-T. Lai, M. Pedram, and S. Vrudhula, "EVBDD-based algorithms for integer linear programming, spectracl transformation, and function decomposition," IEEE Trans. Computer-Aided Design, vol. 8, pp. 959–974, Aug. 1994.

[51] D. Bochman, F. Dresig, and B. Steinbach, "A new decomposition method for multilevel circuit design," in Proc. Eur. DAC, 1991, pp. 374–377.

[52] T. Stanion and C. Sechen, "Quasialgebraic decomposition of switching functions," in Advanced Res. VLSI, 1995.

[53] C. Files and M. Perkowski, "New multi-valued functional decomposition algorithms based on MDD's," IEEE Trans. Computer-Aided Design, vol. 19, pp. 1081–1086, Sept. 2000.

[54] K. Karplus, "Using if-then-else DAG's for multi-level logic minimization," Univ. California, Santa Cruz, UCSC-CRL-88-29, 1988.

[55] V. Bertacco and M. Damiani, "The disjunctive decomposition of logic functions," in IEEE Int. Conf. Computer-Aided Design, 1997, pp. 78–82.

[56] T. Stanion and C. Sechen, "Boolean division and factorization using binary decision diagrams," IEEE Trans. Computer-Aided Design, vol. 13, pp. 1179–1184, Sept. 1994.

[57] C. Yang and M. Ciesielski, "BDS: A BDD-Based Logic Optimization System," Trans. CAD. July 2002.

[58] A. Barenco et al., "Elementary Gates for Quantum Computation," Physical Review A, 52, 1995, pp. 3457-3467.