Journal of Advances in Computer Research
Quarterly ISSN: 2008-6148
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 3, No. 3, August 2012), Pages: 85-96
www.jacr.iausari.ac.ir

Efficient Genetic Based Methods for Optimizing the Reversible and Quantum Logic Circuits

Majid Mohammadi

International Center for Science, High Technology & Environmental Sciences, Kerman, Iran,
Department of Computer Engineering, Shahid Bahonar University of Kerman, Kerman, Iran

mohammadi@mail.uk.ac.ir

Received: 2012/05/15; Accepted: 2012/08/07

Abstract

Various synthesis methods have been proposed in the literature for reversible and quantum logic circuits. However, there are few algorithms to optimize an existing circuit with multiple constraints simultaneously. In this paper, some heuristics in genetic algorithms (GA) to optimize a given circuit in terms of quantum cost, number of gates, location of garbage outputs, and delay, are proposed. The proposed methods can optimize an existing circuit with a given truth table, including don't care values, for different aspects of optimality. The results show good enhancements in the optimization of benchmark circuits compared to the previously published methods.

Keywords: *Optimization, Genetic algorithms, Reversible logic, Quantum circuit, Heuristic method*

1. Introduction

A circuit is reversible if and only if its inputs can be calculated from its outputs and vice versa [1, 2]. Reversible logic is useful both in lossless classical computing and in quantum computing, in which all ideal computations are inherently reversible [3].The synthesis of reversible and quantum circuits differs significantly from the synthesis of traditional irreversible circuits. A considerable amount of work has already been done on the synthesis and optimization of reversible and quantum logic circuits [4-8]. Methods for automated reversible and quantum logic circuit synthesis, based on genetic algorithms (GA) and evolutionary algorithms (EA),have also been developed [9-11]. In these approaches, the global optimization characteristics of the algorithm are used to synthesize reversible and quantum circuits to obtain near optimal circuits. In GA-based synthesis methods, the optimization is a part of the synthesis process. They can also synthesize incompletely-defined functions [12].

Optimizing an existing circuit is important because many synthesis methods do not offer an optimum circuit in terms of quantum cost (QC) or other measures of reversible or quantum circuits. In [7], a template matching method for optimization of circuits including Toffoli gates is proposed; however, it finds only a local minimum for the number of gates of the circuit. Exact synthesis method [8] based on the satisfiability (SAT) technique can synthesize a reversible function using a locally minimal number of Toffoli gates; however, it cannot handle multi-objective optimizations.

In this paper, heuristics applicable in GA-based algorithms, for optimizing an existing reversible circuit in various aspects of optimality, considering the don't care values, are proposed. There are several figures of merit (FoMs) to evaluate, compare, and optimize different logic designs[13]. In the proposed method, one can minimize the number of gates (NoG), QC, number of garbage outputs (NGout), location of garbage outputs, and delay.

The proposed method can also be used for a set of quantum circuits using V, $V^{\dagger}$ and CNOT gates. This set of gates is used in many papers to synthesize the quantum and reversible logic circuits [12, 14, 15]. Our optimization algorithm can provide an optimization for quantum circuits that are represented by truth tables. However, by finding a proper measure to compare the outputs of the quantum circuits, we may extend the algorithm to a wider range of quantum circuits. In this paper, our technique is applicable to reversible or quantum circuits that contain gate libraries that are universal for classical computation.

The structure of paper is as follows. In Section 2, the background about the reversible and quantum circuits is presented. In Section 3, the main contribution of the paper, a set of heuristics for optimizing reversible and quantum logic circuits, is presented. Section 4 illustrates the experimental evaluation of these approaches by applying them to various benchmarks of reversible and quantum logic. The paper is concluded in Section 5.

2. Motivation

Several methods have been proposed for synthesis of reversible and quantum circuits. Though, there are few algorithms for optimizing an existing circuit. Optimizing such a circuit is important because many synthesis methods do not propose an optimum circuit expressed by the figures of merit of reversible or quantum circuits. In [7], a method for optimization of circuits including Toffoli gates is proposed; nevertheless, it offers only a local minimum for the number of gates of the circuit. Exact synthesis method, proposed in [8], can synthesize the reversible functions using a locally minimal number of Toffoli gates; however, it cannot guarantee that other measures such as quantum cost of the circuit are optimum. In the incompletely defined functions, the above methods cannot use don't care values in the optimization process. On the other hand, our GA-based optimization and synthesis algorithm have several advantages respect to the other existing algorithms:

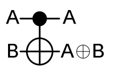
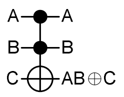
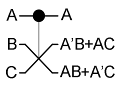
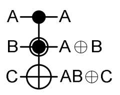
1. Various universal or non-universal reversible logic gates can be used.
2. Many parameters of the circuit are controllable. For example, one may limit the maximum gate size to two or three or limit the algorithm to use special types of gates (also user defined gates).
3. Don't care conditions and garbage outputs can be efficiently used to obtain an optimized circuit.
4. Because of the population-based behavior of the GA in optimization, the results are usually optimized based on the defined cost function.
5. Other optimality measures can be added to the cost function to optimize the circuit with respect to those measures (e.g., delay of the circuit).

3. Background on reversible and quantum circuits and their figures of merit

A reversible logic function maps each possible input vector to a unique output vector. Classical reversible logic gates can be implemented in various technologies, such as CMOS, optical circuits, and nanotechnology based techniques. On the other hand, quantum circuits, based on quantum computing theory, cannot be realized in traditional CMOS technology [3]. A reversible or quantum gate has an equal number of inputs and outputs. With *n* inputs, there exist $2^n!$reversible$n\times n$ gates, where agate is $n\times n$ if it has *n* inputs and *n* outputs [1]. The well-known 2×2 Feynman gate [15] operates as a controlled NOT (CNOT). If the control input of a CNOT is set to '0', then the gate acts as a Buffer gate; otherwise, it acts as a NOT gate. The Feynman gate can be used as a fan-out circuit to clone a logical signal. The Feynman, Toffoli [1], Fredkin [4], and Peres [18] gates are well-known reversible gates, shown in Table 1. The Fredkin and Toffoli gates are both universal, i.e., any logical reversible circuit can be implemented using one of these gates. The Toffoli gate can be generalized to an$n\times n$ gate (TOF*n*) with *n-1* control inputs [1]. If all *n-1* control inputs are '1,' then the target output is the NOT of the target input. In the same way, the Fredkin gate can be generalized to an$n\times n$ gate [4].

Classical reversible logic gates act on binary digits or bits. Quantum gates, on the other hand, act on quantum bits or qubits [3,15]. A qubit is a unit of quantum information. All quantum gates are reversible. They are represented by unitary matrices. The most common quantum gates operate on spaces of one or two qubits. These are the primitive gates of quantum technology. Generally, an *n*-qubit quantum gate can be described by a $2^n\times 2^n$unitary matrix, with complex components and orthonormal rows. Some examples of quantum gates are the Hadamard, Phase Shifter, V ($\sqrt{\text{NOT}}$), Feynman, Toffoli, and Fredkin gates [3]. Some quantum gates, such as the Feynman, Toffoli, and Fredkin gates, have counterparts in reversible logic circuits.

Table 1. Important reversible or quantum gates and their sizes, quantum costs, and delay specifications.

Gate	Size	Reference	QC	Delay	Symbol & Operation
Feynman	2×2	[15]	1	Δ	
Toffoli	3×3	[1]	5	4Δ	
Fredkin	3×3	[4]	5	4Δ	
Peres	3×3	[18]	4	3Δ	
One qubit quantum	1×1	[15]	1	Δ	$\begin{bmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{bmatrix}$
V	1×1	[15]	1	Δ	$\frac{1+i}{2}\begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}$

A quantum or reversible circuit is usually depicted using a series of connected gates, on a number of parallel lines. These lines are the inputs/outputs of the circuit. This method of representation of circuits is called the music line style [12].The fan-out of the reversible logic gates is limited to one. That is, each output of a reversible logic gate can drive only one input of another gate. This limitation, as well as the dissimilarity between reversible and irreversible gates, makes the synthesis of reversible functions more complicated than that of their irreversible counterparts. The synthesis of quantum circuits is even more complicated than that of reversible circuits because of the unitary matrix representation of quantum gates and the use of complex numbers.

To optimize a reversible or quantum circuit, one can use various notions of optimality. The number of gates, the quantum cost, the number of constant inputs, the number of garbage outputs, and the delay are important constraints for the optimization problem. The definitions and a brief review of these measures can be found in the literature [13]:

1. *Number of gates* (NoG): The number of reversible or quantum gates needed to realize a circuit is the NoG of that circuit. Reversible or quantum circuits with similar gates (size and type) can be compared using the NoG criterion.
2. *Quantum cost* (QC): The QC of a reversible or quantum logic circuit is the number of primitive (1×1 or 2×2) reversible or quantum logic gates needed to implement the circuit [15, 16]. The QC of 1×1 and 2×2 gates is considered to be a unit cost, regardless of their internal structure. QC is widely used as a measure to evaluate a reversible design. The QC of six well known reversible and quantum gates is depicted in Table 1.
3. *Number of constant inputs* (NCin): Constant inputs are inputs of a reversible or quantum circuit with arbitrary constant values. Constant inputs are sometimes necessary for the reversible realisation of an irreversible function. Since the values of constant inputs are considered “don’t care”, they can be used to optimize a reversible function efficiently.
4. *Number of garbage outputs* (NGout): These are outputs whose values are not important. Increasing the number of garbage outputs increases the information loss of a reversible circuit.
5. *Delay:* The delay time is the number of stages in a quantum circuit. In this paper, the definition and method proposed in [13] for calculating the delay is used. The delay of 1×1 and 2×2 gates is considered as a unit delay or Δ. Table 1 shows the delay of other reversible or quantum gates respect to the unit delay.

4. Heuristics to optimize reversible and a set of quantum logic circuits

In this section, an algorithm and some heuristics to optimize a given quantum or reversible circuit using the GA is proposed. The optimization will be more efficient if the truth table of the function has don’t care conditions and don’t care outputs (garbage outputs).Our proposed algorithm is comparable to the GA-synthesis algorithm in [12]. The GA-synthesis algorithm, which is explained in detail in [12], can handle the don’t care values in the synthesis process. In this paper, a similar algorithm for optimization purposes is proposed.

The proposed algorithm can be used for optimizing the quantum circuits including V, $V^\dagger$ and CNOT gates. This set of gates is used in literature to synthesize the quantum and reversible logic circuits [12, 14, 15]. In optimization algorithm the quantum logic function have to be represented by a truth table (including don’t care values).

4.1 Coding a circuit

Before using the GA to optimize a reversible or quantum circuit, it has to be coded as a bit string, which is called a chromosome. To code a circuit, we have to select a representation scheme for it. The music line style [12] is the most popular method to show a reversible or quantum circuit. In this method an *n*-input/*n*-output circuit with *m* gates is shown by *n* parallel lines with *m* gates on them. In this research, the music line style for circuit representation is used.

Coding a circuit for synthesis, as shown in [12], is the same as coding it for optimization. Fig.1.a shows the details of coding a gate. The first field encodes the gate, which is assumed, for example, “00” for the generalized Toffoli gate. We define a parameter, the maximum gate length (MGL),which is constant in the optimization algorithm. This parameter is the maximum number of inputs or outputs of a gate in the circuit. Next to the code field, the MGL is the number of fields that represent the locations of the inputs/outputs of the gate on the parallel lines. Different combinations of the locations of inputs comprise different gates in the circuit. In Fig.1.b, the locations of the main input and a control input are the same. In this case, the algorithm ignores the control input, and the resulting gate is a CNOT gate. Fig.1.c shows another combination, in which all the inputs have the same location. In this case the result is a NOT gate. Fig.1.d shows the case in which two control inputs have the same location, resulting in a CNOT gate. Therefore, using this coding method, with a constant value of MGL, other smaller gates can be generated in the optimization process. It is notable that this coding scheme can redundantly encode the same gate in multiple ways (e.g. figure 1.b and 1.d).

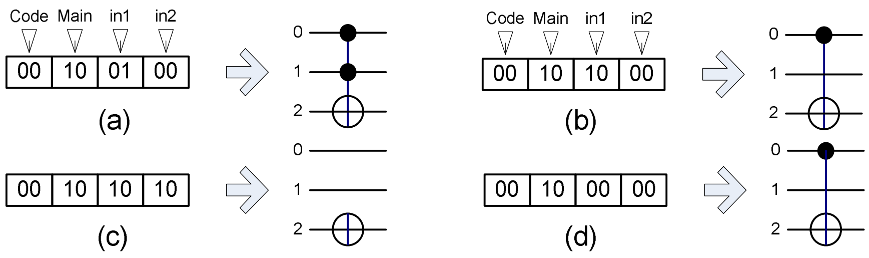


Figure1. The method used for coding a reversible gate and different combinations of locations of inputs: (a) distinct locations, (b) the locations of a main input and a control input are the same, (c) all inputs have the same location, (d) all control inputs have the same location.

4.2 Error Function

In GA-based optimization methods, the algorithm is designed in such a way that it maximizes a function called the fitness function. The algorithm can also be designed to minimize a function, called the error function (EF). In the process of evaluating a circuit, an error function is calculated, which can show our target of optimization. To have an acceptable circuit, the Hamming distance (HD) of the desired truth table (*des_row*array in Fig.2) and outputs of synthesized circuit (*syn_row* array), have to be zero. If one impose the constraint EF = HD, the optimization algorithm finds only acceptable circuits. To obtain an optimized circuit in terms of QC, the QC of the synthesized circuit has to be added to the error function, i.e., the condition EF = HD + QC must hold. The algorithm tries to minimize this EF, which results in a circuit with minimum QC. The QC and HD are both integer values; thus, when a specific value for EF is obtained, one cannot determine which of QC or HD is zero. If the HD is not zero, then the circuit obtained is not acceptable for the desired truth table. To guarantee that HD = 0, we normalize the QC of each synthesized circuit to QCN such that $0 < QCN < 1$. To normalize the QC, a maximum QC value (QC max) for all chromosomes is obtained. In the algorithm, the maximum gate length (MGL) and number of gates (NOG), and the gate library are specified. The gate library shows the type of gates used in the synthesis or optimization process. For example, the CNOT, NOT, and Toffoli gates (CNT library), V, V^+, and CNOT gates (VVC library), or Fredkin gates may be used as the gate library. In the GA-based algorithms, combinations of gate libraries are also acceptable [12]. For example, we assume the CNT gate library. Considering the MGL for the length of a generalized Toffoli gate (when MGL > 2), the QC of the gate is given by Equation 1 [15].

$$Gate\_QC_{max} = 2^{MGL} - 3 \quad (1)$$

The MGL parameter is the maximum value for the size of a gate in the synthesis or optimization algorithm. Therefore, this cost (Equation 1) is the maximum possible value for QC of the generalized Toffoli gate in the circuit. Given the NOG, the maximum QC of the circuit is calculated as Equation 2.

$$QC_{max} = NOG(2^{MGL} - 3) \quad (2)$$

Thus, the normalized QC for each chromosome is calculated using Equation 3.

$$QCN = \frac{QC}{QC_{max}} = \frac{QC}{NOG(2^{MGL}-3)+1} \quad (3)$$

In this case (EF = HD + QCN), if the EF is less than 1, then the circuit is acceptable for our truth table. Since the HD is an integer value, it can be multiplied by an integer number to expand the range of QCN in the error function. For example, if 10*HD is used, then QCN can vary in the open interval of]0,10[. Then, to have a valid circuit for a given truth table, the EF must be less than 10. Multiplying the HD by an integer is useful when you want to change the weights of the optimization measures in the algorithm.

It is possible to also add the delay to the EF to obtain a circuit with minimum delay. Again, the normalized delay (Delay N) is used in the EF expression (Equation 4).

$$EF = \alpha.HD + \beta.QCN + \gamma.DelayN \quad (4)$$

The definition of delay in [13] is used to calculate the delay of each gate or circuit. To calculate the Delay N parameter, the maximum delay (Max Delay)has to be calculated. In [13],it is shown that the unit delay of a gate is less than or equal to its QC. The delay of a circuit is also less than or equal to the sum of the delays of each of the gates in the circuit. Therefore, the Equation 5 and Equation 6 can be used for Max Delay for the normalized delay.

$$MaxDelay = QC_{max} \quad (5)$$

$$DelayN = \frac{Delay}{QC_{max}+1} \quad (6)$$

In Equation 6, the QCmaxis used instead of Max Delay. This also reduces the calculation time of the evaluation section of the optimization algorithm. The values of QCN and Delay N are both less than one. In Equation 4, α, β, and γ are constants that specify the weight of each measure in the optimization process. To guarantee that the obtained circuit satisfies HD = 0, the coefficients α, β, and γ have to satisfy Equation 7.

$$\alpha > \beta + \gamma \quad (7)$$

The condition $EF < \alpha$ is checked to determine whether the circuit is acceptable or not. Between two circuits with $EF < \alpha$, the circuit with the lower EF value is chosen.

4.3 The optimization algorithm

Figure 2 shows the optimization algorithm. First, in Line 1, a population of μ chromosomes is constructed. Each chromosome encodes a complete reversible or quantum circuit. All except the first of the chromosomes are initialized randomly in start of the algorithm (Line 2). Afterwards, in Line 3, the first chromosome is initialized with an existing circuit, which is a valid answer to the given truth table. Don’t care values have to be specified, e.g., by ‘x’ characters, in the truth table. The *while* loop, Line 5, is the beginning of the genetic optimization algorithm. In this loop, λ new chromosomes are produced by using crossover and mutation operators. The crossover operator selects two chromosomes randomly and exchanges some of their corresponding segments. The mutation operator applies random changes to the selected chromosome, with a specific probability, by randomly inverting some of its bits.

Line 8 is the start of the loop that evaluates all λ+μ chromosomes. Evaluation is based on an error function, which was explained in the previous section. For the optimum usage of don’t cares in the optimization process, the heuristic methods proposed in [12] are used. Lines 12 and 13 show how we discount the values of don’t care conditions (Line 12) and garbage outputs (Line 13) to obtain an optimized circuit for the care values. In Line 20, the algorithm selects μ better chromosomes to produce the new population. To select μ better chromosomes from a set of μ+λ chromosomes, one can sort them by ascending EF and select the first μ chromosomes. After the optimization loop, those circuits are acceptable for the desired truth table in that their EF’s are less than α. Since the first chromosome was initiated as the existing circuit, its EF is always less than α. If the optimization loop finds a circuit with a smaller EF value, it replaces the first chromosome by the new one. Therefore, the chromosome[0] is always the best circuit of the population with the minimum value of EF.

4.4 Minimising the number of gates using Buffer gates

The number of gates (NOG) used in the optimization process is a constant value. In this situation, the optimization algorithm could not find answers with gate counts lower than NOG. To allow the algorithm to change the number of gates, the Buffer gate is added to the gate library. A Buffer gate is a 1×1 gate whose output is equal to its input. The more Buffer gates a circuit contains, the fewer main gates it contains. Therefore, the optimization algorithm can also minimize the NOG of a given circuit by maximising the number of Buffer gates in the circuit. It is important to note that minimum value of NOG is not necessarily the minimum value of QC. The QC of a circuit depends on the NOG and on the size of the gates in the circuit. The QC of a gate increases exponentially with the size of the gates. Therefore, using a small value for MGL can decrease the QC of the circuit efficiently.

```
// In this algorithm, number of gates (NOG), maximum gate length (MGL), an existing circuit, truth table of the
//  circuit with don't care conditions, garbage outputs, and gate library are known.
CreateChromosome[ i ] ( $0\le i<\mu$) ; //each chromosome is a Quantum or Reversible circuit      1)
Initialize Chromosome[ i ] with random numbers for ( $1\le i<\mu$) ;      2)
Initialize the Chromosome[ 0 ] as the initial existing circuit ;      3)
Calculate QCmax ;      4)
While (max_iteration)      5)
  {      6)
Generate $\lambda$ newChromosomes using Crossover and Mutation operators ;      7)
For all of $\lambda+\mu$ Chromosome[ i ]//Evaluation loop      8)
    {      9)
For row[ j ] of truth table      10)
{      11)
If row [ j ]isnot "don't care condition"then// row [ j ] = 'x' means don't care condition      12)
13)                HD = HD + Abs ( Hamm ( (des_row [ j ]  & mask), ( syn_row [ j ] & mask ) ) );
14)          //mask bits are '1' for a care outputs and '0' for a don't care outputs
}      15)
           Calculate QCN for Chromosome [ i ];  // QCN = QC / QCmax      16)
           Calculate DelayN for Chromosome [ i ]; // DelayN = Delay/QCmax      17)
           EF[ i ] = $\alpha$.HD + $\beta$.QCN + $\gamma$.DelayN; // $\alpha$, $\beta$, and $\gamma$ are constant weighting factors      18)
    }//End of evaluation loop      19)
Select $\mu$ better Chromosome for reproduction// better circuits have less EF      20)
  } // End of while      21)
Print Chromosome[ 0 ]      22)
END      23)
```

Figure2. The proposed optimization algorithm based on GA

4.5 Optimal location of garbage outputs

In general, garbage outputs and constant inputs are necessary for the reversible implementation of an irreversible function. The minimum number of garbage outputs and constant inputs can be calculated by the Maslow formula [17], shown in Equation 8; in which η is the maximum number of repetitions of output patterns.

$$\mathrm{NGout}_{\min} = \log_2 \eta \qquad (8)$$

To design a circuit with minimum NGout, one can calculate $NGout_{min}$ by Equation 8 and add this number of garbage outputs to the function. Additionally, sometimes it is necessary to add constant inputs to the function. It is important to note that the minimum number of garbage outputs and constant inputs is not necessarily the best choice. To optimize, for example, the QC of a circuit, more garbage outputs and constant inputs may result in a better circuit [12]. Actually, there are optimum values for NGout and NCinfor a reversible function, depending on the complexity of the function [12].

Another problem with garbage outputs is identifying the locations of garbage outputs with respect to the inputs in the truth table of the function. The locations of garbage outputs can highly affect the QC of the synthesized circuit. Therefore, the best locations for garbage outputs in the truth table can result in the best circuit. To find the best locations for garbage outputs, we add Swap gates to each chromosome in the algorithm. Number of Swap gates is *n/2*for an *n*×*n* circuit, to change the location of garbage outputs. Thus, the algorithm is allowed to add the Swap gates anywhere in the circuit. However, it is easier to add these gates on the input or the output side of the circuit. Since the Swap gates can be moved to either sides of the circuit (Fig.3), adding them to one side is sufficient to change the locations of the garbage outputs. In the optimization process, when the algorithm tries to optimize the circuit, it also finds the optimum location of the garbage outputs.

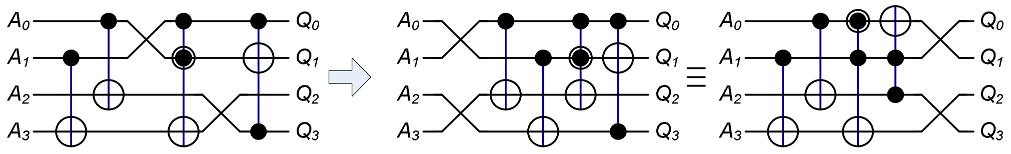


Figure3. Using Swap gates to change the location of garbage outputs: (a) Swap gates are anywhere in the circuit, (b) Swap gates are moved to the input side, (c) Swap gates are moved to the output side.

5. Experimental evaluation

This section presents experimental results for the presented methods. Fig.4 shows, for instance, the optimized circuits for the benchmark 4gt5 function. It is a four input function that determines whether the number defined by the binary encoding of the input, “bcde”, is greater than five. Fig.4.a is the best circuit found in [8], which is implemented using four gates and has a QC of 28. Fig.4.b is the optimized circuit using the proposed algorithm and the CNT gate library. It uses three gates with a QC of 11. This figure also shows that the best location of the f output is the fourth line of the circuit (from the top), on the *b* input line. Using a different gate library, the generalized Fredkin gates, an optimized circuit with only two gates and a QC of 10 is obtained. Finally, using a mixed gate library, allowing an arbitrary gate selection, the function is implemented with one Peres and one Fredkin gate, with a QC of 9. Using the VVC quantum gate library, a circuit with a QC of 7 is found (Fig.4.e).

To show the efficiency of the proposed algorithm and heuristics, we have used them to optimize some benchmarks of reversible and quantum circuits. Table 2 shows the results for optimizing the QC. The results are compared with those of the methods proposed in [8]. The definitions and truth tables of the functions are presented in [19]. In some circuits, ex. rows 5 to 8, there are no improvements in the QC; however, in other benchmarks the improvement is more than 60 percent.

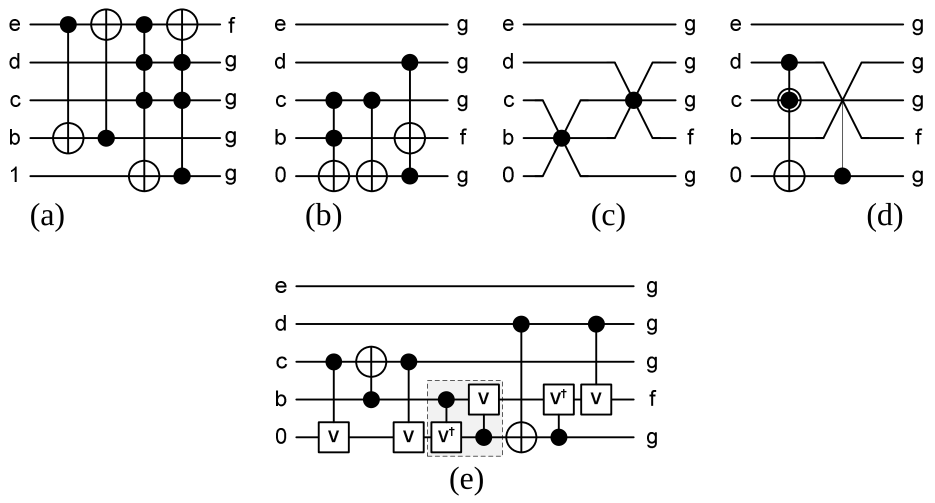


Figure4. Proposed circuits for the 4gt5 benchmark: (a) circuit synthesized in [8], (b) optimized circuit using NCT gate library, (c) optimized circuit using generalized Fredkin gate library, (d) optimized circuit using a mixed gate library, (e) optimized circuit using a quantum VVP gate library.

In the same manner, the optimization algorithm can be used for finding a circuit with a minimum delay. In some cases the algorithm can also minimize both the QC and the delay simultaneously. For example, the 4gt5 of Fig.4.e is a circuit with minimum QC of 7 and minimum delay of 7Δ.

Table 2.Results of optimization of reversible and quantum benchmarks and comparisons with [8].

row	Benchmark	Ref. [8] QC	This Research QC	Improvement
1	Mod5mils	13	11	15%
2	Ham3	9	5	44%
3	Ex-1	8	6	25%
4	Aj-el1	30	29	3%
5	Graycode3	2	2	0%
6	Graycode4	3	3	0%
7	Graycode5	4	4	0%
8	Graycode6	5	5	0%
9	3_17	14	9	36%
10	Mod5d1	11	11	0%
11	Mod5d2	20	11	45%
12	Rd32	12	6	50%
13	Decod24	18	15	16.7%
14	4gt4	54	28	48%
15	4gt5	28	7	75%
16	4gt10	37	13	65%
17	4gt11	7	5	28%
18	4gt12	41	16	61%
19	4gt13	15	12	20%
20	4mod5	9	6	33%
21	4mod7	38	24	37%
22	One-two-three	24	14	42%
23	Mini_alu	33	14	58%
24	alu	22	11	50%

The most important drawback of GA-based algorithms compared to other mathematical methods in both synthesis and optimization is the time of convergence. Because of the vast range of the searching area, the algorithm may not converge for circuits bigger than about 8×8. However, the convergence time efficiently decreases if the desired truth table includes a large number of don't care values (don't care conditions and garbage outputs).

6. conclusions

Although there are various methods to synthesize the reversible and quantum logic circuits, there are few methods to optimize an existing circuit. In this paper, an algorithm and some heuristics to optimize reversible and quantum logic circuits were proposed. The heuristics were used in a genetic algorithm-based optimization method. In the proposed algorithm, they don't care values were efficiently used. We proposed a new coding method to encode a generalized $n \times n$ controlled gate such that all gates smaller than a predefined size can be generated. Contributions including the definition of a proper error function and various aspects of optimality are offered. A method to allow the optimization algorithm to minimize the number of gates, using Buffer gates, was also proposed. An important problem involving garbage outputs, namely the optimal location of garbage outputs respect to inputs, was considered, and a new method using Swap gates was proposed. The proposed algorithm and heuristics were applied to benchmarks of reversible and quantum logic circuits. The results show good enhancements in the optimization of benchmark circuits compared to the previously published papers.

Acknowledgement: This work was supported by: Iran National Science Foundation (INSF).

7. References

[1] Toffoli, T.: Reversible computing. In: Tech memo MIT/LCS /TM-151, MIT Lab for Computer Science (1980).

[2] Landauer, R.: Irreversibility and heat generation in the computing processes. In: IBM J. Res. Develop. July 1961.

[3] Kaye, P., Laflamme, R., Mosca, M.: An Introduction to Quantum Computing. In: Oxford University Press. Jan 2007eBook-LinG, ISBN 0-19-857000-7

[4] Fredkin, E., Toffoli, T.: Conservative logic. In: Int. J. Theo. Phys., 21 (1982) 219.

[5] Gupta, P., Agrawal, A., Jha, N.K.: An Algorithm for Synthesis of Reversible Logic circuits. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 25, Number 11, November 2006, 2317-2330

[6] Kerntopf, P.: A New Heuristic Algorithm for Reversible Logic Synthesis. In: Annual ACM IEEE Design Automation Conference Proceedings of the 41st annual conference on Design automation San Diego, CA, USA, 2004, Pages: 834 – 837, ISBN:1-58113-828-8

[7] Miller, D.M., Dueck, G.W., Maslov, D.: A transformation based algorithm for reversible logic synthesis. In: Proceedings of the 40th Design Automation Conference, Anaheim, CA, pp. 318–323 (2003)

[8] Große, D., Wille, R., Dueck, G.W., Drechsler, R., "Exact Multiple-Control Toffoli Network Synthesis With SAT Techniques", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, VOL. 28, NO. 5, MAY 2009.

[9] Lukac, M.; Perkowski, M., Evolving quantum circuits using genetic algorithm, In Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware, 2002.

[10] Lukac, M., Perkowski, M., Gol, H.: Evolutionary Approach to Quantum and Reversible Circuits Synthesis. In: Artificial Intelligence Review, Vol. 20, Issue 3-4, December 2003, 361–417

[11] Khan, M.H.A.; Perkowski, M., Genetic algorithm based synthesis of multi-output ternary functions using quantum cascade of generalized ternary gates, In Proceedings of the 2004 Congress on Evolutionary Computation, 2004.

[12] Mohamadi, M. Eshghi, M.: Heuristic Methods to use don't cares in automated design of reversible and quantum logic circuits. In: Quantum Information Processing Journal, Springer, Volume 7,Issue 4 (August 2008) pp.175 - 192.

[13] Mohammadi, M. Eshghi, M. "On figures of merit in reversible and quantum logic designs", Quantum Information Processing Journal, Springer, Volume 8, Issue 4 (August 2009) pp.297 - 318.

[14] Mohammadi, M., Eshghi, M., "Behavioral Model of V and V+ Gates to Implement the reversible Circuits Using Quantum Gates", IEEE TENCON08 conference, Heydarabad, India, November 18-21, 2008.

[15] Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., Sleator, T., Smolin, J.A., Weinfurter, H.: Elementary gates for quantum computation. In: Phys. Rev. A 52 (5) (1995) 3457–3467.

[16] Lee, S., Lee, S.J., Kim, T., Jae-Seung Lee, Biamonte, J., Perkowski, M.: The Cost of Quantum Gate Primitives. In: Journal of Multi-valued Logic and Soft Computing, 12(5–6) 2006.

[17] Maslov, D., Dueck, G.W.: Garbage in reversible design of multiple output functions. In: 6th International Symposium on Representations and Methodology of Future Computing Technologies, pages 162-170, March 2003.

[18] Peres, A.: Reversible Logic and Quantum Computers. In: Physical Review, 1985, A 32: 3266–3276.

[19] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in Int. Symp. Multi-Valued Logic, 2008, pp. 220–225. [Online]. Available: http://www.revlib.org