Enhancing the quantum cost of Reed-Muller Based Boolean quantum circuits using genetic algorithms

M Shaban\textsuperscript{1,a}, A Younes\textsuperscript{2,3,b} and A Elsayed\textsuperscript{2,c}

\textsuperscript{1} Department of Mathematics, Faculty of Education, Alexandria University, Alexandria, Egypt
\textsuperscript{2} Department of Mathematics and Computer Science, Faculty of Science, Alexandria University, Alexandria, Egypt
\textsuperscript{3} School of Computer Science, University of Birmingham, Birmingham, B15 2TT, United Kingdom

E-mail: \textsuperscript{a} mshaban002@gmail.com, \textsuperscript{b} ayounes@alexu.edu.eg, \textsuperscript{c} ashrafaf73@yahoo.com

Abstract. There is a direct equivalence between Boolean functions represented in Reed-Muller logic and Boolean Quantum Circuits. Different polarity Reed-Muller expansions will give different Boolean quantum circuits with different cost for the same Boolean function. For a given Boolean function with \(n\) variables there are \(2^n\) possible expansions. Searching for the expansion that gives a Boolean quantum circuit with minimum quantum cost within the search space is a hard problem for large \(n\). This paper will use genetic algorithms to find the fixed/mixed polarity Reed-Muller expansion that gives a Boolean quantum circuit with minimum quantum cost to optimize the circuit realization of a given Boolean function.

1. Introduction

Digital circuits\textsuperscript{1} process data in a digital form represented as binary values (bits) using irreversible logic gates such as AND and OR gates. Irreversible logic gates cause heat dissipation due to the information loss per operation. Irreversible gates should be replaced by reversible gates to build reversible circuits\textsuperscript{1} [1]. By using reversible gates that maintain the number of outputs equals to the number of inputs and bijection between inputs and outputs, the reversibility can be introduced in conventional computing\textsuperscript{2, 3, 4, 5}. The reversible logic has applications in fields such as quantum computing, DNA computing, VLSI design, nanotechnology, optical computing, ...etc.

Lukac et al. used genetic algorithm for automated synthesis of reversible circuits\textsuperscript{6}. Chen et al. proposed a method to design reversible arithmetic circuits using an efficient graph-based evolutionary optimization technique\textsuperscript{7}. Li et al. proposed a best-path search algorithm for reversible logic synthesis based on ACO (Ant Colony Optimization)\textsuperscript{8}. Eftakhar et al. used genetic programming with subtree mutation for evolving common RL arithmetic circuits\textsuperscript{9}. Ahmed et al. proposed a novel algorithm to optimize the quantum cost of reversible circuits by minimizing the multi-calculation of common parts by reordering the terms in a Boolean function\textsuperscript{10}.

The aim of this paper is to propose two genetic algorithms to optimize the quantum cost of Boolean quantum circuits represented in Reed-Muller logic. The idea is based on the fact that there is a direct equivalence between Boolean functions represented in Reed-Muller logic and Boolean quantum circuits\textsuperscript{11}. Different Boolean quantum circuits with different quantum cost for the same Boolean function will arise from different polarity Reed-Muller expansions\textsuperscript{11}.
The structure of the paper is as follows: Section 2 reviews the basic concepts of reversible circuits synthesis, and the equivalence between Boolean functions represented in Reed-Muller logic and quantum Boolean circuits. Section 3 proposes a genetic algorithm to optimize Boolean quantum circuits. Section 4 shows the analysis of the proposed algorithm. Section 5 concludes the paper.

2. BOOLEAN QUANTUM CIRCUITS

Reversible circuits are circuits in which the number of outputs is equal to the number of inputs and also have one-to-one and onto mapping between the vectors of inputs and outputs (bijection) [12]. Reversible circuits are composed of one or more reversible gates. The common reversible gates that are used in literature are NOT gate, CNOT gate, Toffoli gate and General CNOT gate. The function of these gates is shown in figure 1.

![Figure 1: General CNOT gate and special cases.](image)

Reed-Muller logic (RM) is a paradigm of digital logic design which use the operations AND, XOR and NOT to represent Boolean functions [11],

\[ f(x_0, \ldots, x_{n-1}) = \bigoplus_{i=0}^{2^n-1} b_i \psi_i, \]  

where

\[ \psi_i = \prod_{k=0}^{n-1} x_k^{i_k}, \]

where \( x_k = x_k \) or \( x_k' \) and \( x_k, b_i \in \{0,1\} \) and \( i_k \) represents the binary digits of \( k \). \( \psi_i \) are known as product terms and \( b_i \) determines whether a product term is presented or not. \( \bigoplus \) indicates the XOR operation and multiplication is assumed to be the AND operation.

A RM function \( f(x_0, \ldots, x_{n-1}) \) is said to have fixed polarity if throughout the expansion each variable \( x_k \) is either \( x_k \) or \( x_k' \) exclusively, then there will be \( 2^n \) possible expansions. If for some variables \( x_k \) and \( x_k' \) both occur when the function is said to have mixed polarity, and there will be \( 2^{nm} \) possible expansions [13]. The polarity of a function can be changed by replacing any variable \( x_i \) by \( (x_i' \oplus 1) \) or any variable \( x_i' \) by \( (x_i \oplus 1) \) [11].

When replacing a variable \( x_i \) with \( (x_i' \oplus 1) \) or \( x_i' \) with \( (x_i \oplus 1) \), it should be replaced in the whole expression if fixed polarity expansion is required, and it can be replaced in some product terms if a mixed polarity expansion is required.

Once a Reed-Muller expansion is expressed, we can build a circuit for this expansion [11]. Different polarity RM expansions will give different quantum circuits with a different cost for the same Boolean function as shown in figure 2.

3. PROPOSED GENETIC ALGORITHM

Two algorithms will proposed in this paper. For the first algorithm, to find the best-fixed polarity of a given Boolean function, the polarities of the Boolean function represented as binary chromosomes of \( n \) bits, where \( n \) is the number of variables of the Boolean function.

For the second algorithm, to find the best-mixed polarity of a given Boolean function, the polarities of the Boolean function represented as binary chromosomes of \( nm \) bits, where \( n \) is the number of variables of the Boolean function, and \( m \) is the number of the product terms that exist in the 0-polarity of the given function.
3.1 Chromosome Encoding
For a Boolean function in fixed polarity, the polarity of a function of \( n \) inputs is represented in binary form on \( n \) bits. Consider \( f(x_0, x_1, x_2) \) defined as follows:

\[
f(x_0, x_1, x_2) = x_0 x_1 x_2 \oplus x_0 \oplus 1
\]

which is in 0-Polarity, it can be represented as a chromosome 000, and then the chromosome represents the 1-Polarity expansion of the same function will be 001.

For a Boolean function in mixed polarity, the encoding of the chromosome depends on the number of variables and the number of product terms that exist in 0-polarity expression, then the length of the chromosome will be \( nm \) where \( n \) is the number of variables and \( m \) is the number of product terms in 0-polarity, each \( n \) consecutive bits forms a block then there will be \( m \) blocks. Consider the function in equation (3), the length of the chromosome will be 9. Every 3 bits (Block) is related to a single product term in 0-polarity, and each bit in the block determines whether a variable is in the true or complemented form in this product term, then the chromosome 100000000 represents the expression,

\[
f = x'_0 x_1' x_2' \oplus x'_0 x'_2 \oplus x'_1 x'_2 \oplus x_0' x_1' \oplus x_1' \oplus x'_2 \oplus 1.
\]

3.2 Genetic Algorithm Operators
The genetic operator is an operator used in genetic algorithms to guide the algorithm towards a solution of a given problem. There are three main operators to be applied to the current population to evolve the next population [14] which are selection, crossover and mutation.

3.2.1 Selection. Selection operator is used to select the individuals to be parents that will contribute to the population at the next generation. The proposed algorithm used roulette wheel selection with a few numbers of elitism individuals which gives better results because it keeps the best individual during generations and large variation. Chromosomes are evaluated based on their fitness values. The fitness value of a chromosome is the quantum cost of the quantum circuit that can be built for the expansion represented by the chromosome. The quantum cost can be directly computed from a Reed-Muller expansion using equation (5).

\[
TotalCost = \sum_{i=1}^{m} (2^{n_i+1} - 3),
\]
where $m$ is the number of product terms in the expansion, and $n_i$ is the number of variables that appeared in a given product term.

### 3.2.2. Crossover

Crossover operator is used to exchange the genetic information of two parents to generate new offsprings for the next generation. The simplest way to do this is to randomly choose a single crossover point, then swap bits around the picked point between the two parents [14]. Consider the functions in equations (3) and (6), figure 4 explain crossover operator.

$$f = x_0'x_1x_2' \oplus x_1x_2 \oplus x_0'x_1 \oplus x_1 \oplus x_0' : 5 \text{ Polarity} \quad (6)$$

**Figure 4:** Crossover Rules (| is the crossover point).

### 3.2.3. Mutation

Mutation operator is to randomly change the gene of an individual member to form a new member. For an individual encoded in binary, we can negate a randomly chosen bit from 1 to 0 or vice versa [14]. Consider the function in equation (6), Mutation can understood as shown in figure 5.

**Figure 5:** Mutation Rules.

The proposed algorithm used a bit mutation rate instead of random bit mutation, this means that all bits are under threaten to be mutated. The complete set of GA parameters is shown in table 1.

| Table 1: GA Parameters. |
|-------------------------|
| Genetic Algorithm Parameter | Value     |
| Population               | 300-500   |
| Selection                | Roulette Wheel |
| Crossover                | Uniform   |
| Mutation                 | Bit Mutation |
| Crossover Probability    | 0.5-0.6   |
| Mutation Probability     | 0.001-0.1 |
| Elitism                  | Yes       |
| Population Replacement   | Whole Population |

### 4. RESULTS AND DISCUSSION

The algorithms proposed in this paper have been tested on the reversible circuits from RevLib [15] with a single output line. Each circuit is expressed as a Boolean function in 0-polarity Reed-Muller form, then the proposed algorithms are applied. The experimental results from the proposed algorithms have been collected on a machine with a Core-i7 processor, 8 GB RAM using Microsoft Windows 8.1. The proposed algorithms are implemented in JAVA programming language and are compiled using NetBeans IDE 8.2.
The results of the proposed GA have been compared with the results shown in RevLib [15] and the results of the reorder algorithm in [10]. The experimental results are displayed in table 2 and table 3, where the algorithm achieves a reduction in the quantum cost (QC) compared with other algorithms in literature. For example, figure 6a shows the majority_239 circuit from RevLib [15], while figure 6b is the circuit that has been evolved from the proposed algorithm.

![Figure 6: majority_239 Circuit and its Evolved Circuit, majority_239 Circuit, (a) Quantum Cost=136 [15], (b) Evolved Circuit of majority_239, Quantum Cost = 128.](image)

Table 2: The reversible Boolean function synthesized in REVLIB [15] versus using the proposed GA.

| Function    | Quantum Cost [15] | Fixed Polarity Result | Mixed Polarity Result | Best Result |
|-------------|-------------------|-----------------------|-----------------------|-------------|
| majority_239 | 136               | 128                   | 146                   | 128         |
| parity_247  | 32                | 18                    | 18                    | 18          |
| cm152a_212  | 252               | 371                   | 281                   | 281         |
| t481_263    | 237               | 228                   | 261                   | 228         |
| sym6_145    | 777               | 735                   | 595                   | 595         |
| sf_276      | 152               | 25                    | 25                    | 25          |
| sf_275      | 51                | 31                    | 36                    | 31          |
| mux_246     | 2427              | 2279                  | 1721                  | 1721        |
| 5alu-9      | 55                | 47                    | 36                    | 36          |
| 5rd53f2     | 69                | 50                    | 50                    | 50          |
| 4Mod5_8     | 9                 | 21                    | 22                    | 21          |
| 7con1f1     | 139               | 163                   | 131                   | 131         |
| 7con1f2     | 65                | 67                    | 71                    | 67          |
| 4gt4-20     | 51                | 36                    | 66                    | 36          |
| 4gt5-21     | 17                | 15                    | 14                    | 14          |
| 8newill     | 875               | 822                   | 1292                  | 822         |
| 8newtag     | 555               | 437                   | 867                   | 437         |
| **Average** | **347.000**       | **321.941**           | **331.294**           | **273.000** |
Table 3: The reversible Boolean function synthesized with reorder algorithm in [10] versus using the proposed GA.

| Function  | Quantum Cost [10] | Fixed Polarity Result | Mixed Polarity Result | Best Result |
|-----------|------------------|-----------------------|-----------------------|------------|
| 5alu-9    | 24               | 47                    | 36                    | 36         |
| 7conf1f1  | 121              | 163                   | 131                   | 131        |
| 7conf1f2  | 61               | 67                    | 71                    | 67         |
| 4gt4-20   | 51               | 36                    | 66                    | 36         |
| 4gt5-21   | 14               | 15                    | 14                    | 14         |
| 8newill   | 920              | 822                   | 1292                  | 822        |
| 8newtag   | 387              | 437                   | 867                   | 437        |
| 7rd73f2   | 7                | 7                     | 7                     | 7          |
| 8rd84f2   | 8                | 8                     | 8                     | 8          |
| 8rd84f3   | 509              | 509                   | 509                   | 509        |
| 4gt10-22  | 34               | 35                    | 34                    | 34         |
| 4gt10-23  | 5                | 5                     | 5                     | 5          |
| 4gt10-24  | 43               | 36                    | 34                    | 34         |
| 4gt10-25  | 13               | 13                    | 13                    | 13         |
| 4sf_232   | 22               | 29                    | 30                    | 29         |
| lt41      | 26               | 37                    | 28                    | 28         |
| lt42      | 56               | 42                    | 42                    | 42         |
| lt43      | 28               | 39                    | 31                    | 31         |
| lt44      | 42               | 47                    | 46                    | 46         |
| lt45      | 58               | 57                    | 55                    | 55         |
| lt51      | 141              | 140                   | 177                   | 140        |
| lt52      | 128              | 100                   | 109                   | 100        |
| Average   | 122.636          | 122.318               | 163.863               | 119.272    |

5. CONCLUSION
This paper proposed two genetic algorithms to synthesis reversible circuits. The proposed algorithms work on any Boolean function represented in Reed-Muller form. The algorithms uses the 0-polarity of a given function, then apply the proposed genetic algorithms to find the fixed/mixed polarity that gives the Boolean quantum circuit with the minimum cost. The experimental results show that the best quantum cost of the circuit can be obtained from the fixed polarity or the mixed polarity of the equivalent Reed-Muller expansion of the Boolean function.
REFERENCES

[1] Dubey V, Mishra G R and Singh O P 2012 Implementation of a two-bit binary comparator using reversible logic International Journal of Scientific and Research Publications 2

[2] Bruce J W, Thornton M A, Shivakumaraiah L, Kokate P S and Li X 2002 Efficient adder circuits based on a conservative reversible logic gate Proc. IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002 pp 83–88

[3] Haghparast M, Jassbi S J, Navi K and Hashemipour O 2008 Design of a novel reversible multiplier circuit using HNG gate in nanotechnology World Appl Sci J. p 974

[4] Khan M 2002 Design of full-adder with reversible gates. Proc. International Conference on Computer and Information Technology

[5] Rentergém Y V and Vos A D 2005 Optimal design of a reversible full adder IJUC 1(4) 339–55

[6] Lukac M and Perkowski M 2002 Evolving quantum circuits using genetic algorithm. Proc. 2002 NASA/DoD Conference on Evolvable Hardware pp 177–185

[7] Cheng K W and Tseng C C 2002 Quantum full adder and subtractor Electronics Letters 38 1343–44

[8] Li M, Zheng Y, Hsiao M S and Huang C 2010 Reversible logic synthesis through ant colony optimization Proc. of the Conference on Design, Automation and Test in Europe (3001 Leuven, Belgium, Belgium: European Design and Automation Association) pp 307–310

[9] Eftekhar S M A, Habib S M and Hashem M M A 2013 Evolutionary design of digital circuits using genetic programming CoRR abs/1304.2467

[10] Ahmed T, Younes A and Elsayed A 2018 Improving the quantum cost of reversible Boolean functions using reorder algorithm Quantum Information Processing 17(5) 104

[11] Younes A and Miller JF 2004 Representation of Boolean quantum circuits as Reed–Muller expansions International Journal of Electronics 91(7) 431–44

[12] Hadjam F Z and Moraga C 2017 A hierarchical distributed linear evolutionary system for the synthesis of 4-bit reversible circuits Claudio Moraga: A Passion for Multi-Valued Logic and Soft Computing R Seising and H Allende-Cid (Cham: Springer International Publishing) pp 233–249

[13] Almaini A E A 1994 Electronic Logic Systems (3rd Ed) (Upper Saddle River, NJ, USA: Prentice-Hall, Inc)

[14] Srinivas M and Patnaik L M 1994 Genetic Algorithms: A Survey Computer 27(6) 17–26

[15] Wille R, Große D, Teuber L, Dueck G W and Drechsler R 2008 RevLib: An online resource for reversible functions and reversible circuits Int’l Symp. on Multi-Valued Logic pp 220–225 RevLib is available at http://www.revlib.org