

A Hierarchical Approach to Computer-Aided Design of Quantum Circuits

Marek Perkowski,+* Martin Lukac,* Pawel Kerntopf, & Mikhail Pivtoraiko,* Michele Folgheraiter *, Dongsoo Lee, + Hyungock Kim,+ Woong Hwangbo,+ Jung-wook Kim+ and Yong Woo Choi.+

*Department of Electrical and Computer Engineering, Portland State University, Portland, OR, 97201. USA, +Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, 373-1, Kusong-dong, Yusong-gu, Taejeon, 305-701, Korea, & Institute of Computer Science, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland.

Abstract: A new approach to synthesis of permutation class of quantum logic circuits has been proposed in this paper. This approach produces better results than the previous approaches based on classical reversible logic and can be easier tuned to any particular quantum technology such as nuclear magnetic resonance (NMR). First we synthesize a library of permutation (pseudo-binary) gates using a Computer-Aided-Design approach that links evolutionary and combinatorics approaches with human experience and creativity. Next the circuit is designed using these gates and standard 1×1 and 2×2 quantum gates and finally the optimizing tautological transforms are applied to the circuit, producing a sequence of quantum operations being close to operations practically realizable. These hierarchical stages can be compared to standard gate library design, generic logic synthesis and technology mapping stages of classical CAD systems, respectively. We use an informed genetic algorithm to evolve arbitrary quantum circuit specified by a (target) unitary matrix, specific encoding that reduces the time of calculating the resultant unitary matrices of chromosomes, and an evolutionary algorithm specialized to permutation circuits specified by truth tables. We outline interactive CAD approach in which the designer is a part of feedback loop in evolutionary program and the search is not for circuits of known specifications, but for any gates with high processing power and small cost for given constraints. In contrast to previous approaches, our methodology allows synthesis of both: small quantum circuits of arbitrary type (gates), and permutation class circuits that are well realizable in particular technology.

1. Introduction

While quantum mechanics and quantum computing are established research areas, automated quantum circuit synthesis is still only at the beginning of its exploration [2,4,6,7,8]. In quantum computation we use quantum bits (qubits) instead of classical binary bits to represent information. This gives the advantage of being able to perform massively parallel computations in one time step. The design of quantum circuits of practical size is still technologically impossible (the maximum number of qubits in year 2002 is 7), but the progress is fast and there are no arguments based on physics against the possibility of building powerful quantum computers in the future. Therefore quantum computing area of research is recently flourishing.

Finding an effective and efficient method of designing quantum circuits can have three application areas:

- (1) Optimizing quantum circuits for NMR [26,27,28,29], ion trap, quantum dot, cavity quantum electrodynamics or Si-based nuclear spin quantum computer technologies that exist already in practice. Each of these technologies has different minimization requirements and all of them require minimizing both the width and the length of scratchpad register (number of qubits processed). Width is absolutely critical and length is also important because of decoherence. Circuit reduction is very important for present technology, so exact or sub-minimal methods should be developed for small circuits, less than 7 variables. This is a very small number from the standard CAD algorithms point of view, but the synthesis problem for quantum logic is more difficult.
- (2) Modeling quantum computers in FPGA-based reconfigurable hardware for speeding-up computations that are very inefficient on standard computers [9]. Since for current FPGA technologies only small quantum circuits can be emulated, the requirements are similar to point (1)
- (3) Designing new optimized gates and circuits for theoretical investigations and for use in future quantum computers. If we believe that such computers will exist, efficient CAD methods for large number of variables, tens or hundreds, should be developed, but these algorithms will be non-optimal. This is a theoretical research for non-existing technology in year 2002.

2. Quantum Computing

The major difference between quantum logic and binary logic is the concept of the information itself. While the classical (binary or multi-valued) representations of information are precise and deterministic, in Quantum Computing (QC) the concept of bit is replaced by the qubit. Unlike classical bits that are realized as electrical voltages or currents present on a wire, quantum logic operations manipulate qubits [7]. Qubits are microscopic entities such as a photon or atomic spin. Boolean quantities of 0 and 1 are represented by a pair of distinguishable different states of a qubit. These states can be a photon's horizontal or vertical polarization denoted by $|\rightarrow\rangle$ or $|\downarrow\rangle$, or an elementary particle's spin denoted by $|\uparrow\rangle$ or $|\downarrow\rangle$ for spin up and spin down, respectively. After encoding these distinguishable quantities into Boolean constants, a common notation for qubit states is $|0\rangle$ and $|1\rangle$.

Qubits exist in a linear superposition of states, and are characterized by a wavefunction ψ . As an example, it is possible to have light polarizations other than

purely horizontal or vertical, such as slant 45° corresponding to the linear superposition of $\psi = \frac{1}{\sqrt{2}}[\sqrt{2}|0\rangle + \sqrt{2}|1\rangle]$. In general, the notation for this superposition is $\alpha|0\rangle + \beta|1\rangle$. These intermediate states cannot be distinguished, rather a measurement will yield that the qubit is in one of the basis states, $|0\rangle$ or $|1\rangle$. The probability that a measurement of a qubit yields state $|0\rangle$ is $|\alpha|^2$, and the probability is $|\beta|^2$ for state $|1\rangle$. The absolute values are required since, in general, α and β are complex quantities.

Pairs of qubits are capable of representing four distinct Boolean states, $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$, as well as all possible superpositions of the states. This property is known as “entanglement”, and may be mathematically described using the Kronecker product (tensor product) operation \otimes [7]. As an example, consider two qubits with $\psi_1 = \alpha_1|0\rangle + \beta_1|1\rangle$ and $\psi_2 = \alpha_2|0\rangle + \beta_2|1\rangle$. When the two qubits are considered to represent a state, that state ψ_{12} is the superposition of all possible combinations of the original qubits, where

$$\psi_{12} = \psi_1 \otimes \psi_2 = \alpha_1 \alpha_2 |00\rangle + \alpha_1 \beta_2 |01\rangle + \alpha_2 \beta_1 |10\rangle + \beta_1 \beta_2 |11\rangle. \quad (1)$$

Superposition property allows qubit states to grow much faster in dimension than classical bits. In a classical system, n bits represents 2^n distinct states, whereas n qubits corresponds to a **superposition** of 2^n states. Observe also that in the above formula some coefficient can cancel, so there exist a constraint bounding the possible states in which the system can exist. These all contribute to difficulty in understanding the concepts of quantum computing and creating efficient analysis, simulation, verification and synthesis algorithms for QC. Generally, however, we believe that much can be learned from the history of Electronic Computer Aided Design and the lessons learned should be used to design efficient CAD tools for quantum computing.

In terms of logic operations, anything that changes a vector of qubit states can be considered as an operator. These phenomena can be modeled using the analogy of a “quantum circuit”. In a quantum circuit wires do not carry Boolean constants, but correspond to pairs of complex values, α and β . Quantum logic gates of this circuit map the complex values on their inputs to complex values on their outputs. Operation of quantum gates is described by matrix operations. Any quantum circuit is a composition of parallel and serial connections of blocks, from small to large. Serial connection of blocks corresponds to multiplication of their (unitary) matrices. Parallel connection corresponds to Kronecker multiplication of their matrices. So, theoretically, the analysis, simulation and verification are easy and can be based on matrix methods. Practically they are tough because of the problem dimension exponential growth of matrices. Synthesis problem can be formulated as decomposing hierarchically a given unitary matrix to serial and parallel connections of smaller matrices, until basic directly realizable quantum primitives are reached. This problem is very difficult in such basic formulation and therefore several special methods have been and are being developed, especially in the last 5 years.

Probabilistic calculations based on this representation are used in only very small quantum computers so far (most with 3 bits), but it was verified that information can be represented as a superposition of states of single qubits, and that in one time step operations can be performed on several qubits. Beside this useful effect of quantum computing, various other effects resulting from qubit encoding emerge, such as qubit entanglement. Moreover it was shown [7] that any quantum computing has to be reversible, which affects all synthesis methods. Concluding, building quantum

computers becomes more and more technical rather than only scientific issue, and the methods developed to design them, such as formal representation, modeling and synthesis will have applications not only to quantum computing but also to DNA and other nano-technologies because of their reversible nature.

In this paper we focus only on the synthesis of arbitrary quantum circuits (and quantum gates in particular) of small size, less than five variables. We concentrate on designing the circuits from a class of permutation gates which have unitary matrices being permutation matrices. Such circuits correspond to classical Boolean functions. However, our gate design methods are for general quantum circuits, with arbitrary unitary matrices. The presented methods can be specialized to some classes such as f-controlled-phase-shift gate circuits [25], and generalized to multiple-valued quantum logic [30] and mixed multiple-valued logic.

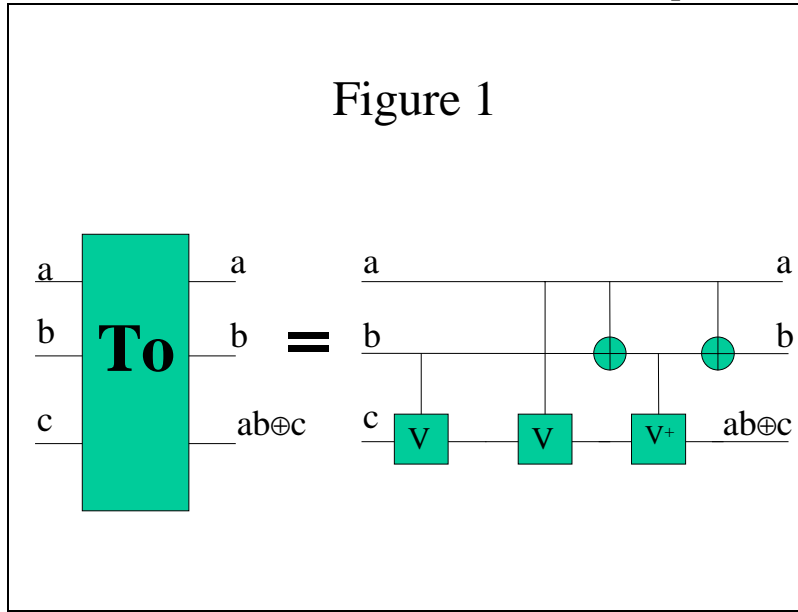
The paper is organized as follows. First, we discuss some important issues related to quantum logic circuits. **By discussing them we want to prove that it is not true that binary reversible logic synthesis methods can be directly applied to permutation quantum circuits.** (This is a common belief, partially true, but only a first step). By presenting this discussion, we would like to encourage researchers with logic synthesis background to create new improved methods that will be more practically useful to optimizing sequences for current existing technologies such as NMR.

Our theses in the first part are the following:

1. Toffoli (CCNOT) and Fredkin are not always the best gates for quantum computing. (Toffoli gate is described by equations $P=a, Q=b, R= ab \oplus c$, Fredkin gate by equations $P=a, Q= a'b+ac, R=ab+a'c$). Multi-input Toffoli gate look simple in a diagram, but take a lot of gates when redrawn to 3×3 gates with auxiliary constants - they are not primitives. The synthesis should be performed at one hand on a lower level of quantum primitives such as CNOT (Feynman), controlled square-root-of-not and Hadamard, and on the other hand on the level of more powerful reversible gates, such as those introduced in [perk], to simplify the search by increasing gate granularity.
2. The transformation from optimal reversible circuit (on any level of gates) to the minimal quantum sequence for NMR programming is far from being trivial, and so far no combinatorial optimization algorithms have been created for them. The problems of gate ordering, input variable ordering, local transformations, removal of swap gates (planarization) and others, are quite similar to technology mapping and physical design areas in classical CAD, but so far they are solved only ad hoc by physicists.
3. All gate cost assumptions that can be found in general quantum papers (and not in specialized papers about NMR technology) are far too approximate and can lead to highly non-optimal sequences.

Second, after presenting these difficulties in more detail, we outline our CAD tools, in which evolutionary and human-oriented interactive methods are combined. We propose a generalized approach to the problem of quantum computing (QC) CAD by using a simple encoding and a generic genetic algorithm (GA) without any problem-specific operators. Our results show that, in contrast to published work [4,6], any kind of genetic operators can be used. We were able to synthesize completely automatically more complex circuits than those by previous programs, for instance the Margolus gate.

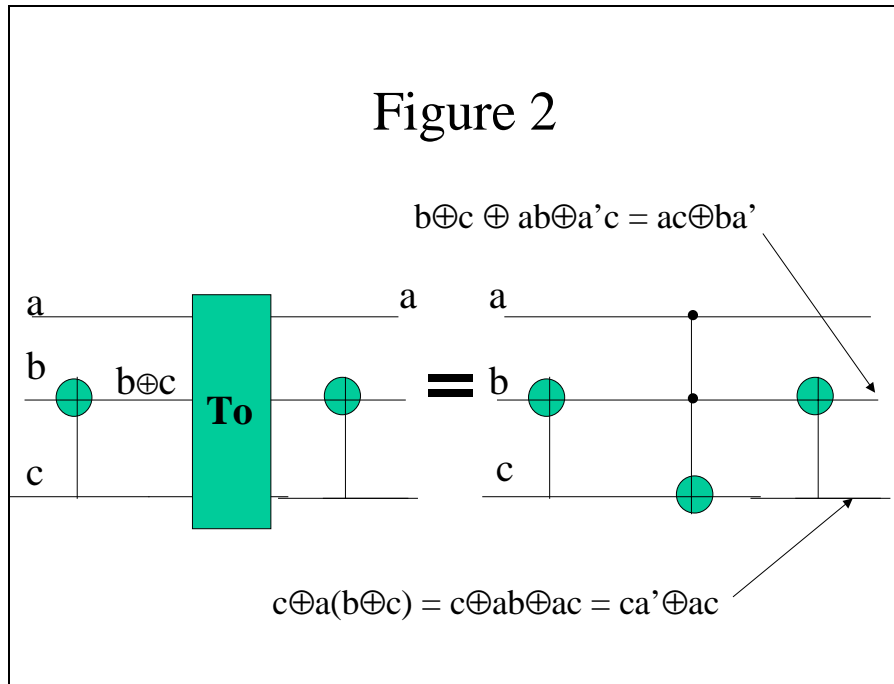
3. New Gates and their Cost Functions for the Optimization Algorithms



An important problem, not discussed so far by other authors, is the selection of the cost functions to evaluate the quantum circuit designs. Although the detailed costs depend on any particular realization technology of quantum logic, so that the cost relations between for instance Fredkin and Toffoli gates can differ in NMR and ion trap realizations, the assumptions used in several previous papers; that each gate costs the same, or that the cost of a gate is proportional to the number of inputs/outputs, are both far too approximate.

In this paper we will illustrate more precise cost functions for gates that are used in our optimization algorithms (even more accurate costs will be presented in the forthcoming paper, here we just want to signalize the important problem). We will follow in the footsteps of previous papers in quantum computing [12,smolin,lee] and we will realize all gates from 1×1 and 2×2 gates. Moreover, according to [15] we will assume that the cost of every 2×2 gate is the same. We will assume also that a 1×1 gate costs nothing, since it can be always included to arbitrary 2×2 gate that precedes or follows it. Thus, in first approximation, every permutation quantum gate will be build from 1×1 and 2×2 quantum primitives and its cost calculated as a total sum of 2×2 gates used.

Using the well-known realization of Toffoli gate with truly quantum 2×2 primitives, shown in Figure 1 [15], the cost of Toffoli gate is 5 2×2 gates, or simply, 5. In Figure 1, V is an square-root-of-NOT gate (unitary matrix V) and V^+ its hermitian. Thus $V V$ creates a unitary matrix of NOT gate and $V V^+ = I$ (an identity matrix, describing just a quantum wire). The reader can analyze correctness of this construction by analyzing all possible values of inputs signals. (the generalization of this gate to n -inputs without constant wires is shown in [12]). Now we will realize the Fredkin gate from the Toffoli gate. Using GA [16] or synthesis methods from this paper, we can synthesize the Fredkin gate using two Feynman and one Toffoli gate as in Figure 2.



Substituting the Toffoli design from Figure 1 to Figure 2 we obtain the circuit from Figure 3a. After using the obvious EXOR-based transformation shown in the right (change of order of Feynman gates), the final circuit from Figure 3b is obtained. Observe that a cascade of two 2×2 gates is another 2×2 gate, so following [15] we obtain a circuit from Figure 3c with the cost of 5. Thus, the cost of Toffoli gate is exactly the same as the cost of Fredkin gate, and not half of it, as some authors assumed and as may be suggested by classical binary schemata of such gates, where Toffoli gate includes single a Davio gate and a Fredkin gate - two multiplexers.

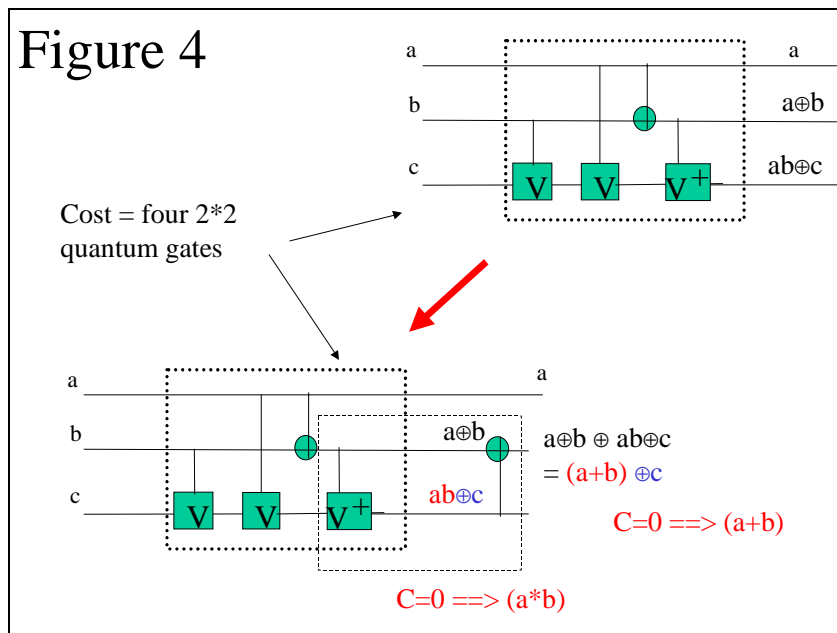
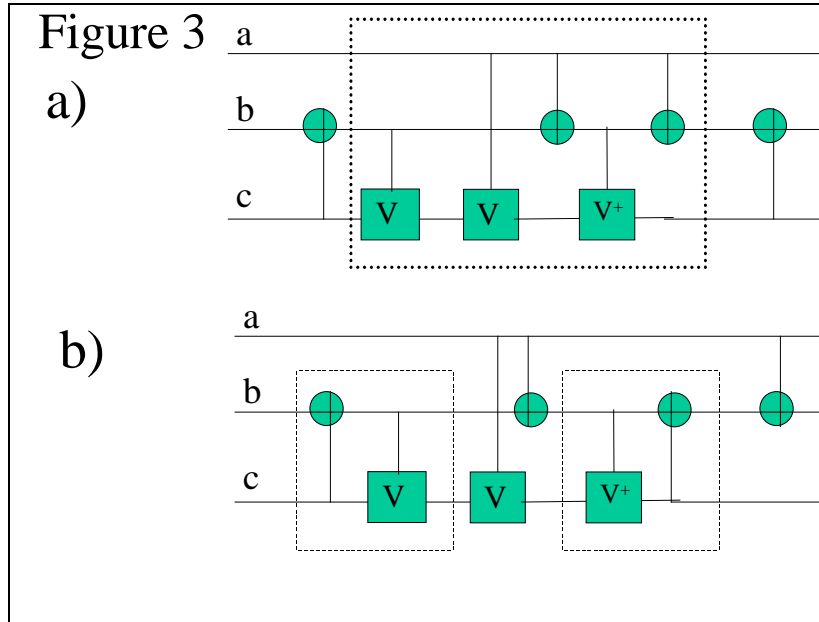
Encouraged with this observation, let us calculate costs of other known gates. It was shown in [31] that the cost of Miller's Gate is 7 and not 9, as might be expected from its binary schematics using Feynman and Toffoli gates. Interestingly, another realization of Miller's gate has an even smaller cost of 6. Observe that since swap gate can be realized by a cascade of three Feynman gates, according to the cost evaluation method from [15] its cost is also 1 and not 3 as assumed previously.

Similarly the costs of 3×3 gates by Kerntopf [xx], Margolus [21], De Vos [xx], Khan [xx], and Maslow [xx], costs of all 4×4 Perkowski's gates [BP 2002], and other gates from [21] can be calculated, which is left to the reader. Next observe that a new permutation quantum gate with equations:

$$\begin{aligned}
 P &= a \\
 Q &= a \oplus b \\
 R &= ab \oplus c
 \end{aligned}$$

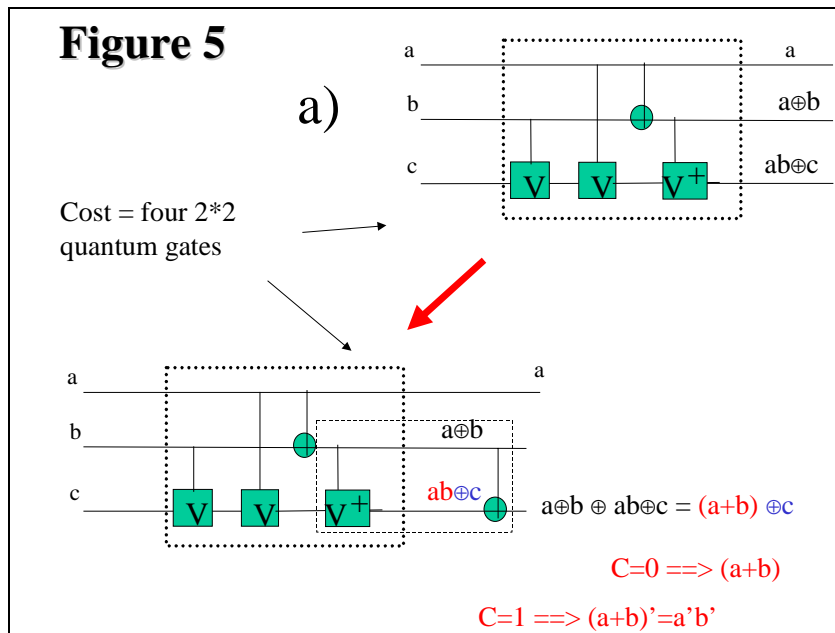
can be realized with cost 4. It is just like a Toffoli gate from Figure 1 but without the last Feynman gate from right. This is the cheapest quantum realization known to us of a complete (universal) permutation gate and it is thus worthy further investigations.

We found that this gate was invented by Peres [21], but it has been not used in reversible or quantum computing. It is now a challenge to other researchers to find a cheaper universal 3*3 gate, that would use only 2*2 and 1*1 gates (it is known that several such universal sets of gates exist. [12]).



Observe in Figure 4, that the Peres Gate with Feynman gates located in all possible points at its periphery creates powerful and inexpensive new gates. For instance by locating Feynman gate with EXOR up on output wires b and c, generates a gate with equations: $a = a$, $b = (a+b) \oplus c$, $c = ab \oplus c$. Again, the Feynman gate can be combined with the output V^+ gate so the cost of the combined new gate is again only four. Another possibility of connecting Feynman gate to the outputs b, c of the Peres gate is

shown in Figure 5. There exist also two possibilities of connecting Feynman gate to inputs b, c, which similarly lead to two new gates, and four possibilities of connecting Feynman gates to both inputs b, c and outputs b, c, which leads to four new gates.



4. Frame-based search generation and genetic algorithms

Let us observe that there exist very many combinatorial possibilities of connecting Feynman (and other) gates to inputs or outputs of basic gates, as shown in section 3. We came thus to a conclusion that such procedure should be automated, for exhaustive and partial search, and that it should be interactively called by the user, who would declare structure, gates, constraints and search parameters like depth, search type (depth first, breadth first, OR vs AND/OR, etc), number of node children, etc. [ZOBRIST].

The user selects some seed gate and defines symbolically all possible structures that can be built by connecting additional gates to it. The types of these gates, the places of their addition and the simplifying (combining) transformations that can be applied, as well as costs of gates, are the parameters of the program. These gates are like slots in a frame structure used in AI programming. The sequence of slots is like a chromosome. This method combines evolutionary programming and frames. The details about evolutionary algorithms for both truly quantum circuits and permutation circuits can be found in [xx] and [dd], respectively. All automatically created new gates, or gates satisfying some criterion (like small cost or (partial) satisfaction of unitary matrix) are displayed on screen to the user, together with their cost functions. This way a library of powerful complex permutation gates is created that on one hand are geared towards structured design methods (such as Kerntopf gate used in regular structures called Nets [Warsaw, 21], and on other hand the gates that have good realizations (short and completely realizable sequences) in any particular quantum technology, such as especially NRM [25 - 29].

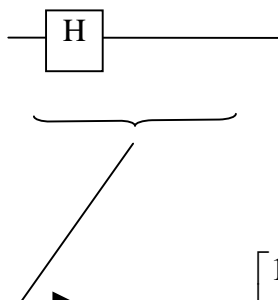
Applying the macrogenerations from abstract quantum gates to quantum gate primitives and tautology and equivalence transformations not on the level of permutation gates but on the level of truly quantum 1×1 and 2×2 primitives allows to optimize the quantum circuits further. For instance, a n -input Toffoli gate can be build without constant inputs using only 2×2 primitives as a result of macrogeneration [Barenco]. These transformations are presented in detail in [Markov, Shigeru, 25-29] and will be not discussed here. Many of them are based on obvious rules of EXOR algebra, such as some illustrated above.

Encoding for the frame generator is the same as for the genetic algorithm. Also, the same matrix-operation based verification scheme from the fitness evaluation subroutine of GA is used. An example of our encoding is shown in Figure 6.

Figure 6: Transformation of a QC from the encoded chromosome (on the left) to a final quantum circuit notation representation of this QC (on the right). Here S is a Swap gate, H is a Hadamard gate, W is a wire. In the middle there is one CCNOT (Toffoli) gate.

On the left side of Figure 1 it is shown how the circuit on the right of the same figure is encoded. As can be seen, there is no free space in the proposed encoding. Each place in the circuit is presented as a symbol of a unitary matrix of certain elementary quantum gate. A wire has a unitary identity matrix representation. While evaluating the fitness function, Kronecker products (tensor products) are executed on matrices of parallel gates (blocks, circuits), and standard matrix multiplications are performed on serial connections of gates. Each QC is parsed in parallel blocks, evaluated separately and finally multiplied together to give the final unitary matrix representation of the QC. This final matrix is next compared with the target matrix to evaluate their distance as a part of fitness function calculation. The example shown in Figure 1 illustrates a common inconvenience of encoding quantum circuits for genetic algorithms. A quantum gate CCNOT can be placed over three different arbitrary wires in a quantum circuit. However with the encoding used, there is no information indicating what gates are connected to what wires, beside the order of the gates. To solve this problem we insert two Swap gates (one before and one after) the CCNOT. This implies that outside of the Swap gates the CCNOT seems like being on wires 2,4 and 5, but the real CCNOT gate uses wires 3,4 and 5. In order of being able to encode a QC without any additional parameters, the circuit is split into parallel blocks where

each block can be evolved separately.



$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & -1 \\ 0 & 0 & -1 & 1 \\ -1 & -1 & 0 & 0 \end{bmatrix}$$

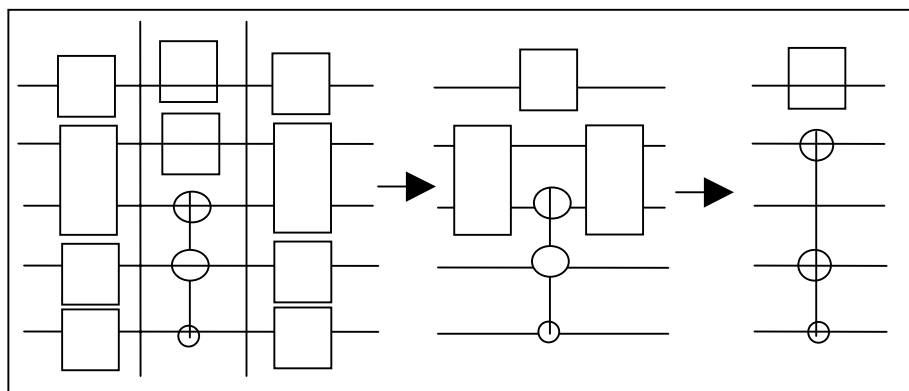


Figure 7: Examples of Kronecker product \oplus , and of Matrix product $*$ on a sample of a circuit.

The results are quite encouraging. In every case the GA found the requested gate, however in no case the automatically created chromosome was better than the circuit for which the corresponding target unitary matrix was created. Summary of results is shown in Table 1.

Number of inputs per q-gate	Number of generations	pM	pC	Real time (average 20 runs)	pM<0.2 Number of generations	Real time (average 20 runs)	Population size
1 - input	<50	0.4	0.6	< 30 seconds	<100	< 1 minute	50
2 - inputs	<50	0.6	0.4	< 30 seconds	<100	< 1 minute	50
3 - inputs	50 - 200	0.6	0.6	<1 minutes	<200	<3 minutes	60

Table 1: Results of experiments. Due to the similarity of results we grouped the results by the number of inputs/outputs of the requested q-gate. PM and PC are probability of mutation and crossover.

All circuit evolved were exact copies or at least had same number of wires, of the searched circuit. For small gates the convergence is logically faster, because of the restricted recombination between different gates. A gate with more inputs than the number of wires in the circuit was not tested. The 3 input gates test shows the same result, however with exponential time. The results are measures of average values over 20 runs. Depending on the circuit we were looking for, the times are, as

predicted, increasing very fast with the increase of the number of QC inputs. Two configurations were tested. First, high probabilities of mutation (0.4-0.7) and crossover (0.4-0.6) were applied. The results are surprising because of so high mutation probability. The size of the GA population was set in range [50,100]. The local very high probability of mutation allows a fast dangerous search. However the runs were stopped as soon as a good solution was found and the best individual examined. A large random search with mutation used on a recombination problem seems to have a positive effect in a restricted search. The solution was found also when the mutation was of a small order, however the time of search raised as well.

Next step was to test composite circuits proposed by [4,6]. We selected three of them shown in Figure 4. The first two are both circuits to produce EPR as in [4], the last is the “send” circuit originally proposed by [10] and evolved in [6].

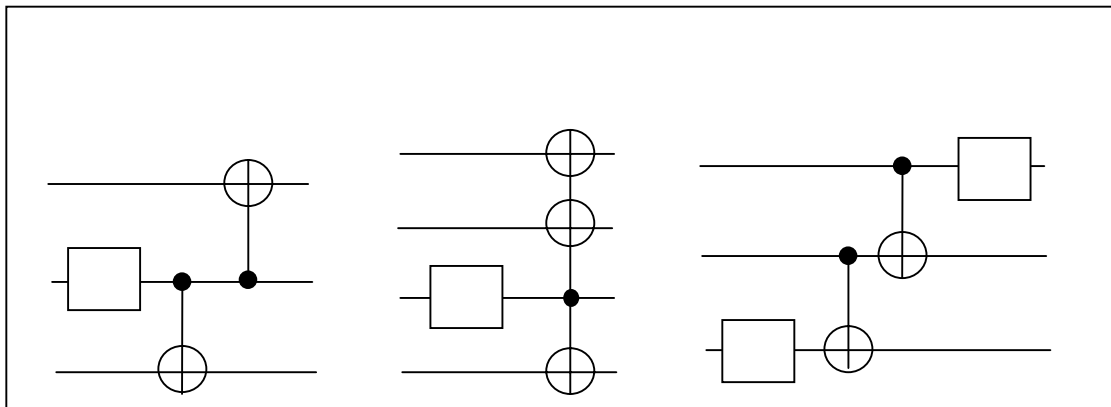


Figure 4: 3 types of circuits searched with the GA.

The results are shown below in Table 3. We were able to find all searched circuits, however in this part of experimentation the starting set of gates was open. Our GA found for all benchmarks at least similar, if not better, results compared to the published results of the studied cases. Even if the number of generations grows exponentially, the real time still remains reasonable.

Number of inputs per q-gate	Number of generations	pM	pC	Real time (average 20 runs)	pM<0.2 Number of generations	Real time (average 20 runs)	Population size
3 - input	<150	0.4	0.6	< 1 minutes	<300	< 2 minute	50
4 - inputs	<350	0.6	0.4	< 2 minutes	<900	< 3 minute	50

Table 2: Results of benchmark tests for assembled circuits.

The 3- and 4- input circuit search was made under similar conditions as the first part of experiments. Results from both tables shows that GA can be very successfully used to synthesize circuits. The time can be reduced by appropriate hardware and consequently used for still larger designs. Using this algorithm, we were not able to find less expensive quantum realizations of any 3*3 permutation gates than Smolin-like realizations and well-known solutions and our hand designs, but we found new gates of the same cost, and we found new realizations with the same costs as the well known ones (Toffoli and Margolus gates). For instance, only 99 individuals were created in the genetic algorithm pool to find the optimal solution to Margolus gate. Many new gates, some interesting and of small costs, have been found as a by-product of searching for known gates. These results will be analyzed elsewhere.

1. Conclusion

We have shown that the evolutionary computation can be used for automated QC development in real time using standard PC computers. Designed as shown, this algorithm can be also easily implemented on parallel computers or in classical binary FPGA-based evolvable hardware. An interesting research will be to implement evolutionary learning in future truly quantum hardware which will lead to a new area of Evolutionary Quantum Hardware (EQH). Our program found one new circuit that was earlier came across by Williams [7] and three circuits located by Rubinstein [4]. In all cases that we studied the program was faster than the results previously published. In contrast to previous works that concentrated on some particular types of circuits such as teleportation [7] and entanglement [4] our approach is fully general. For instance the optimized version of the “send” circuit found by Williams was created. We will further experiment with the algorithm trying to find various realizations for gates and circuits from [8,9,10,12,13,14].

Our algorithm and its data structure can be applied without any modification to reversible circuits from “pseudo-classical” circuits [14]. Such circuits are used for instance in the famous Grover’s Quantum Search Algorithm [11]. Reversible gates realizing Boolean operations can be realized not in quantum but in several other reversible technologies such as DNA, single-electron transistor, mechanical nano-switches, quantum dots or CMOS.

Although all benchmarks proved the convergence of our GA and results were better than previous ones, the goal of our approach is not only to benchmark a GA, but mainly to explore various evolutionary and other approaches [Alan,Andrey] to reversible and quantum circuit synthesis. Next step is to apply different Evolutionary

algorithms such as Baldwinian or Lamarckian GA, genetic engineering or Evolutionary strategies. The encoding used here fits well also the design of reversible logic, where each parallel block in the chromosome is a small sub-circuit, with same number of wires as its neighbors. Obtained results also show the problem of scalability of any designing method. Here for 5 wires the time of search is small, and as previously said hardware implementation of a GA will speed the process up.

References

1. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning Addison Wesley*, 1989.
2. Y. Z. Ge, L. T. Watson, and E. G. Collins. Genetic algorithms for optimization on a quantum computer. In *Unconventional Models of Computation*, pp. 218-227.
3. K-H Han, K-H Park, C-H Lee, and J-H Kim, "Parallel quantum-inspired genetic algorithm for combinatorial optimization problems," In *Proceedings of the 2001 Congress on Evolutionary Computation*, volume 2, pp. 1422-1429, 2001.
4. B.I.P. Rubinstein, "Evolving quantum circuits using genetic programming", *Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001)*, pp. 144-151 (2001)
5. L. Spector, H. Barnum, H. J. Bernstein, and N. Swamy, "Finding a better-than-classical quantum AND/OR algorithm using genetic programming," In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pp. 2239-2246, Washington D.C., 6.-9. July 1999. IEEE, Piscataway, NJ.
6. C.W. Williams, Gray G. Alexander, "Automated Design of Quantum Circuits", *QCQC '98*, Springer-Verlag, pp. 113-125 (1999)
7. Williams C. P., Clearwater S. H., "Explorations in Quantum Computing", *Springer-Verlag*, New York Inc. (1998)
8. T. Yabuki and H. Iba. "Genetic algorithms and quantum circuit design, evolving a simpler teleportation circuit," In *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pp. 421-425, 2000.
9. G. Negovetic, M. Perkowski, M. Lukac, A. Buller, "Evolving quantum circuits and an FPGA-based Quantum Computing Emulator," *Proc. Intern. Workshop on Boolean Problems*, 2002.
10. Brassard G., Braunstein S. L., Cleve R., "Teleportation as Quantum Computation", in *Proceedings of the Fourth Workshop on Physics and Computation*, New England Complex System Institute.
11. L.K. Grover, "A Framework for Fast Quantum Mechanical Algorithms," *ACM Symposium on Theory of Computing (STOC)*, 1998.
12. A. Barenco et al., "Elementary Gates For Quantum Computation", *Physical Review A* **52**, 1995, pp. 3457-3467
13. M. Nielsen & I. Chuang, *Quantum Computation and Quantum Information*, Cambridge Univ. Press, September 2000.
14. T. Hogg et al., "Tools for Quantum Algorithms", <http://arxiv.org/abs/quant-ph/9811073>, 1998.
15. J. Smolin, D. P. DiVincenzo, "Five two-qubit gates are sufficient to implement the quantum Fredkin gate." *Physical Review A*, Vol. 53, no. 4, April 1996, pp. 2855-2856.
16. M. Lukac and M. Perkowski, "Evolving Quantum Circuits Using Genetic Algorithm," *Proc. of NASA/DOD Workshop on Evolvable Hardware*, Washington, D.C. July 2002.
17. M. Lukac, M. Pivtoraiko, A. Mishchenko, and M. Perkowski, "Automated Synthesis of Generalized Reversible Cascades using Genetic Algorithms," *Proc. 5th Intern Workshop on Boolean Problems*, Freiberg, Germany, September 19-20, 2002. pp 33-45
18. Miller, D. M., "Spectral and Two-Place Decomposition Techniques in Reversible Logic," *Proc. Midwest Symposium on Circuits and Systems*, on CD-ROM, August

2002

19. D. M. Miller and G.W. Dueck, "Spectral Techniques for Reversible Logic Synthesis," *submitted to RM 2003*.
20. M. H. A. Khan, and M. Perkowski, "Multi-Output ESOP Synthesis with Cascades of New Reversible Gate Family," *submitted to RM 2003*.
21. Maslov, D., and G. W. Dueck, "Garbage in Reversible Designs of Multiple-Output Functions," *submitted to RM-2003*.
22. Dueck, G. W., and D. Maslov, "Reversible Function Synthesis with Minimum Garbage Outputs," *submitted to RM-2003*
23. E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, 21, pp. 219-253, 1982
24. P. Kerntopf, "Synthesis of multipurpose reversible logic gates," *Proceedings of EUROMICRO Symposium on Digital Systems Design*, 2002, pp. 259-266.
25. J-S. Lee, Y. Chung, J. Kim, and S. Lee, "A Practical Method of Constructing Quantum Combinational Logic Circuits," arXiv:quant-ph/9911053v1, 12 Nov. 1999.
26. J. Kim, J-S Lee, and S. Lee, "Implementation of the refined Deutsch-Jozsa algorithm on a three-bit NMR quantum computer", *Physical Review A*, Volume 62, 022312, 2000.
27. J. Kim, J-S. Lee, and S. Lee, "Implementing unitary operators in quantum computation," *Physical Review A*, Volume 62, 032312, 2000.
28. M. D. Price, S.S. Somaroo, A.E. Dunlop, T. F. Havel, and D. G. Cory, "Generalized methods for the development of quantum logic gates for an NMR quantum information processor," *Physical Review A*, Vol. 60, Number 4, October 1999, pp. 2777-2780.
29. M.D. Price, S.S. Somaroo, C.H. Tseng, J.C. Core, A.H. Fahmy, T.F. Havel and D.G. Cory, "Construction and Implementation of NMR Quantum Logic Gates for Two Spin Systems," *Journal of Magnetic Resonance*, 140, pp. 371- 378, 1999.
30. A. Al-Rabadi, L.W. Casperson, M. Perkowski and X. Song, "Multiple-Valued Quantum Logic," Booklet of 11th Post-Binary Ultra Large Scale Integration (ULSI) 2002 Workshop, pp. 35-45, Boston, Massachusetts, 15th May 2002.
31. G. Yang, W.N.N. Hung, X. Song and M. Perkowski, "Majority-Based Reversible Logic Gate," *submitted to RM 2003*.
32. D. Deutsch, "Quantum computational networks," *Proc. Roy. Soc. Lond. A*. 425, 1989, pp. 73-90.
33. K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation Rules for Designing CNOT-based Quantum Circuits," *Proc. DAC 2002*, New Orleans, Louisiana.
34. P. Kerntopf, "Maximally efficient binary and multi-valued reversible gates," *Proceedings of ULSI Workshop*, Warsaw, Poland, May 2001, pp. 55-58.
35. A. Khlopotine, M. Perkowski, and P. Kerntopf, "Reversible logic synthesis by gate composition," *Proceedings of IWLS 2002*. pp. 261 – 266.
36. A. Mishchenko and M. Perkowski, "Logic Synthesis of Reversible Wave Cascades", *Proc. IEEE/ACM International Workshop on Logic Synthesis*, June 2002. pp. 197 – 202
37. M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, B. Massey, "Regularity and symmetry as a base for efficient realization of reversible logic circuits," *Proceedings of IWLS 2001*, pp. 90-95, 2001.
38. M. Perkowski, L. Jozwiak, P. Kerntopf, A. Mishchenko, A. Al-Rabadi, A.

- Coppola, A. Buller, X. Song, M. M. H. A. Khan, S. Yanushkevich, V. Shmerko, and M. Chrzanowska-Jeske, "A general decomposition for reversible logic," *Proceedings of RM 2001*. pp. 119 – 138.
39. M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, B. Massey, "Regular realization of symmetric functions using reversible logic," *Proceedings of EUROMICRO Symposium on Digital Systems Design*, 2001, pp. 245-252.
 40. V.V. Shende, A.K. Prasad, I.L. Markov, J.P. Hayes, "Reversible Logic Circuit Synthesis," *Proc. 11th IEEE/ACM Intern. Workshop on Logic Synthesis*, 2002, pp. 125 – 130.
 41. A. Al-Rabadi, "Novel Methods for Reversible Logic Synthesis and Their Application to Quantum Computing," *Ph. D. Thesis*, Portland State University, Portland, Oregon, USA, October 24, 2002.