

Developing Automatic Synthesis Methodologies for Quantum Circuits using Genetic Algorithms

Ph.D. Project

Ph.D. student: Eng. Cristian Ruican, MSc.
Scientific advisor: Prof.Dr.Eng. Mircea Vladutiu

Advanced Computing Systems and Architectures Laboratory
University Politehnica Timisoara, 2 V. Parvan Blvd., Timisoara 300223, Romania
cristian.ruican@cs.upt.ro
<http://www.acsa.upt.ro>

Abstract. The task of this report is to present the proposed research program, and to review the literature describing the main characteristics of quantum computing, genetic algorithms and circuit synthesis. We also present selected parts from a submitted article, which proposes a solution for quantum circuit synthesis using genetic algorithms. Our goal is to perform automatic quantum circuit synthesis for a given functional description using rippled steps. We propose and will pursue the usage of a parser which will create an internal data structure for the provided description. Also, we employ simulation in order to determine the final quantum state, and we encourage the usage of an optimizer that will allow a better handling of data, and in the last step foster the usage of genetic algorithms in order to perform the synthesis tasks.

1 Introduction

Civilization has advanced, as people discovered new ways of exploiting various physical resources such as materials, forces and energies. In the twentieth century information was added to the list, when the invention of computers allowed complex information processing to be performed outside human brains. The history of computer technology has involved a sequence of changes from one type of physical realization to another — from gears to relays, to valves to transistors to integrated circuits and so on. Today's advanced lithographic techniques can squeeze fraction of micron wide logic gates and wires onto the surface of the silicon chips. Soon even smaller parts will be yielded and inevitably reach a point where logic gates are so small that they are made out of only a handful of atoms. On the atomic

scale, matter obeys the rules of quantum mechanics, which are quite different from the classical rules that determine the characteristics of conventional logic gates. Therefore, if computers are to become smaller in the future, new quantum technology must replace or supplement what we have now. The point is, however, that quantum technology can offer much more than cramming more and more bits to silicon and multiplying the clock-speed of microprocessors. It can support entirely new kind of computation with qualitatively new algorithms based on quantum principles [14].

Quantum information science is analyzing how the principles of quantum physics can be used in order to improve the acquisition, transmission, and processing of information. A quantum computer would be a new type of machine that, by exploiting the unusual quantum properties of information, could perform certain types of calculations far more efficiently than any foreseeable classical computer [13].

In the quantum computational framework, there are polynomial time solving algorithms for problems having exponential classical solutions. The quest is - on one hand - to search if there are other possible effective quantum algorithms and - on the other hand - to be able to produce efficient implementations for the already known algorithms. The most feasible implementation of quantum algorithms is based on the quantum circuit (gate network) model [15].

1.1 Thesis proposed name

We propose the following title for our research: "Developing Automatic Synthesis Methodologies for Quantum Circuits using Genetic Algorithms". This title describe all the three domains which we are trying to connect in an amprehensive and coherent manner. Our target is to find an adequate methodology, able to perform quantum circuit synthesis.

1.2 Scientific domain

For our research as specified above, there are some domains that, we believe, are strong connected. The merge between quantum computing and genetic algorithms was already made. The field of Evolvable Quantum Information (EQI) has significantly grown over the last years [8]. At first glance, the merging of quantum computation and

evolvable computation seems natural and benefic. Indeed, relevant progress has been signaled in the EQI subfield of Quantum-Inspired Genetic Algorithms (QIGA) including the so-called evolvable quantum hardware or the automatic synthesis of quantum circuits by evolvable means [8]. Ongoing developments concerning the other EQI subfield of Quantum Genetic Algorithms (QGA) have been presented previously [9].

The genetic algorithm (GA) has been introduced by John Holland in the '70 [3]. A basic genetic algorithm shall have the following structure:

- Choose initial population
- Repeat
 - Evaluate the individual fitness of a certain proportion of the population
 - Select pairs of best-ranking individuals, in order to reproduce
 - Breed new generation through crossover and mutation
- Until terminating condition

GA's are search algorithms, based on the genetic evolution of chromosomes. A genetic algorithm start point is the creation of an initial population based on a chromosome, that encode some specific characteristic proprieties as defined by a problem. Then a fitness function is defined in order to evaluate each individual, in order to evaluate if its evolution is closer to what we search. The fitness function will associate a grade to each individual, grade which is in turn used by the selection algorithm that keeps in the generation only the individuals who show a real progress, allowing solution convergence. After applying the selection, some individuals may be removed from the population, their place being taken by new individuals, randomly generated. This new generation will suffer some genetic operations (i.e. mutation, crossover) in order to increase the solution convergence and avoid focusing to the search on a local maximum. The genetic algorithms are successfully used where a large space of possible solutions is defined.

1.3 Relevance

There are many quantum simulators available at this moment, but only few theoretical papers are available on the quantum circuit synthesis. This is the way we are trying to bring here our contribution.

The synthesis relevance has two views. The first one, if the progress in technology is extremely fast and it is outstripping the designers abilities to make use of the created opportunities. The second view is generated by the situation where the technology is not available on large scale and the designers can use only a small set of gates for the design. This is one of the reasons why the development and the application of new and more suitable design methodologies is very important for the modern computer system industry.

Quantum synthesis has a bigger relevance when it is related to the simulation results. Any of the final result simulation may have a physical implementation, with the help of the synthesis algorithm (we can use a quantum circuit database for the available circuit types). Therefore, starting from a program written in a high programming language we obtain the physical device with the help of synthesis.

1.4 Background

In quantum computation the qubit is the basic unit of information. In bra-ket notation a qubit is a normalized vector in a two dimensional Hilbert space $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, $|\alpha|^2 + |\beta|^2 = 1$, ($\alpha, \beta \in \mathbb{C}$) where $|0\rangle$ and $|1\rangle$ are the basis states [7]. The quantum system is described by a superposition of the basis states whereas a classical binary system can settle in one of the basis states '0' or '1' only [2]. The qubits can be organized in linear structures called quantum registers, encoding a superposition of all possible states of the classical register. For a n -qubit quantum register, its corresponding state is a normalized vector in a \mathcal{H}^{2^n} space, $|\psi_n\rangle = \sum_{i=0}^{2^n-1} \alpha_i|i\rangle$, where $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$, $i \in \mathbb{N}$. Quantum circuits are constrained networks of gates with no cloning and no feedback allowed [7]. The quantum gate is a physical device implementing an unitary operator that represents the quantum state transformation. Due to the unitary property, all quantum circuits are reversible; this model is considered as the most feasible implementation for quantum algorithms.

Physically, the qubit may be under the form of two state polarization of a photon, or two energy levels of an electron, or two spin directions, etc. Two or more qubits can create a quantum gate, and more quantum gates can create a quantum circuit. The quantum circuit implements a quantum algorithm.

A quantum computer will give us an incredible processing power, allowing us to solve problems in real time, which on classical computers necessitate exponential time. We have the examples of integer factorization and the discrete-logarithm problem which are experimental today on classical devices, but can be solved polynomially in quantum computation context.

In this presentation we will adopt the matrix representation for the quantum gates, this representation being helpful for the implementation of our synthesis approach. We explain below some of the main operators, which are applied on quantum states as unitary transformation matrices.

- Tensor product: given a $m \times n$ matrix A and a $p \times q$ matrix B, their direct product $C = A \otimes B$, also called their Kronecker product, is a $(mp) \times (nq)$ matrix with elements defined by $c_{\alpha\beta} = a_{ij}b_{kl}$, where $\alpha \equiv p(i-1) + k$ and $\beta \equiv q(j-1) + l$.

Example:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \otimes \begin{bmatrix} e & f \end{bmatrix} = \begin{bmatrix} ae & af & be & bf \\ ce & cf & de & df \end{bmatrix} \quad (1)$$

- Transpose: the object obtained by replacing all a_{ij} elements with a_{ji} . For a second-tensor rank a_{ij} , the tensor transpose is simply a_{ji} . The matrix transpose, most commonly written A^T , is the matrix obtained by exchanging A's rows and columns, thus satisfying the identity $(A^T)^{-1} = (A^{-1})^T$.

Example:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}^T = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \quad (2)$$

- Unitarity: is the property of an operator (or a matrix) that is unitary. A square matrix U is a unitary matrix if $U^H = U^{-1}$, where U^H denotes the conjugate transpose and U^{-1} is the matrix inverse.
- Matrix product: the product C of two matrices A and B is defined as $c_{ik} = a_{ij}b_{jk}$, where j is summed over for all possible values of i and k and the notation above uses the Einstein summation convention. The implied summation over repeated indices without the presence of an explicit sum sign is called Einstein summation, and is commonly used in both matrix and tensor analysis.

Therefore, in order for matrix multiplication to be defined, the dimensions of the matrices must satisfy $(n \times m)(m \times p) = (n \times p)$, where $(a \times b)$ denotes a matrix with a rows and b columns.

Example:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix} \quad (3)$$

There are not so many known quantum gates. Also, there are only few such elementary gates. Fig.1 presents several quantum gates [1] that may be used by our algorithm's database. For description we use the matrix and graphics representation. The graphic representation is important in understanding the circuit diagram obtained with this approach, whereas the matrix representation (shown only for some gates) is important in understanding how to manipulate these gates.

- HADAMARD (Fig. 1 (a)): this gate operates on a single qubit.

$$H(q) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (4)$$

- CNOT (Fig. 1 (b)): this gate operates over two qubits. The first qubit is considered a controlling qubit, which complements the second controlled qubit if it is 1, otherwise leaves it unchanged.

$$CNOT(q1, q2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5)$$

- TOFFOLI (Fig. 1 (c)): this gate operates over three qubits. The first two qubits are considered as controlling qubits, which complement the third controlled qubit if they are both 1, otherwise leaves it unchanged.

$$Toffoli(q1, q2, q3) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

- SWAP (Fig. 1 (d)): this gate operates over two qubits, swapping their values

$$SWAP(q1, q2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

- FREDKIN (Fig. 1 (e)): this gate operates on three qubits. The last two qubits are swapped if the first qubit is 0 or are left unchanged otherwise.

$$Fredkin(q1, q2, q3) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

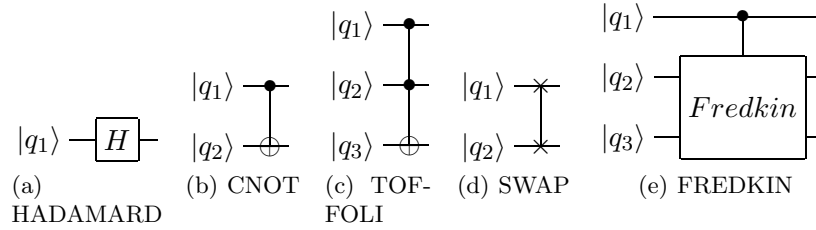


Fig. 1. Elementary gates.

2 Related work

An important step in quantum computing simulation was made by Bernhard Omer [21] which implemented a Quantum Computation Language. He consider that one of the reasons for the slow adoption of quantum computers by the computer science community is the confusing variety of formalisms (Dirac notation, matrices, gates,

operators, etc.), none of which has any similarity with classical programming languages, as well as the rather physical terminology in most of the available literature. He proposed the QCL (Quantum Computation Language) which tries to fill this gap: QCL is a high level, architecture independent programming language for quantum computers, with a syntax derived from classical procedural languages like C or Pascal. This allows for the complete implementation and simulation of quantum algorithms (including classical components) in one consistent formalism. Other simulations are written in common used programming languages. They are available on Mathematica (QuCalc, QDENSITY, qmatrix), on C/C++/Java (Quantum Computer Language, Open Qubit, libquantum (C++), libquantum (C), Qubiter, QCSim, etc), on Perl/PHP (Quantum::Superpositions, Quantum::Entanglement), on Matlab/Octave (Quack!, Qubit4matlab, etc), etc [22].

In the synthesis field, some scientific papers are available. As proposed previously, a four-phase design flow assists computations by transforming a quantum algorithm from a high-level language program into precisely scheduled physical actions as presented in the Fig.2 [4]. The first three phases are implemented by a compiler and

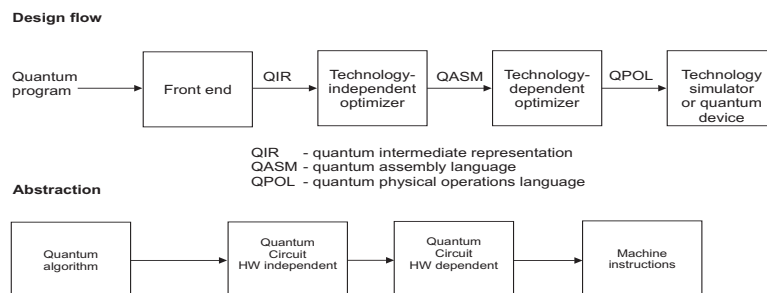


Fig. 2. Proposed design flow [4].

the last phase is hardware dependent (in some cases can be just a simulator). The simulation and layout tools incorporate details of the emerging quantum technologies that would ultimately implement the algorithms described by higher level languages [4].

In the second and third phases, the envisaged situation is to optimize the program representation of the circuit using procedures similar to those used in the present for digital circuits. Also, in refer-

ence [4], the proposal is to employ commercial circuit synthesis tools, and to use only some type of circuits for synthesis (from a library).

When developing reusable software for automating quantum circuit design, reducing technological dependence is desirable. The former step uses an abstract gate library, such as AND-OR-NOT, and emphasizes the scalability of synthesis algorithms that capture the given computation's global structure[4].

In [4] some challenges are identified:

1. Design a high-level programming language in order to create quantum algorithms that encapsulates the principles of quantum mechanics such as superposition and entanglement in a natural way, for physicists and programmers.
2. Find efficient technology-independent optimization algorithms, that work well on realistic classes of quantum circuits, and develop strategies for adapting generic circuits to specific implementation technologies.
3. Develop simulation techniques for quantum circuits and high-level programs, that will allow designers to evaluate meaningful design blocks.
4. Identify fault-tolerant architectural strategies that can be used with emerging quantum device technologies, such as ion traps.
5. Find efficient optimization algorithms for fault-tolerant circuits that minimize the number of fault paths, size of code and the number of gates.

An application of a genetic algorithm for evolving quantum computing circuits has been proposed in [5]. The genetic algorithm automatically searches for the appropriate circuit design that yields the desired output state. The fitness function compares the current output with the desired output, the search being stopped when a close match is found. The presentation briefly discusses the operation of a quantum gate over the multi-qubit system. The paper also presents some examples of the evolved circuits, using the algorithm. Prashant [5] has proposed a simple program on two qubits as shown in the Fig.3.

Shende et al. proposed a top-down structure and effective computation by employing the Cosine-Sine Decomposition [6]. With the help of an optimized quantum multiplexor, a quantum analog Shannon decomposition (see the Fig.4) of Boolean functions is derived,

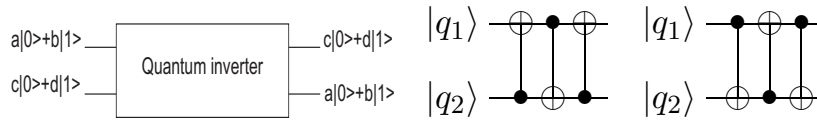


Fig. 3. Sample program as presented in [5].

by applying this decomposition recursively to quantum operators, which leads to a circuit synthesis algorithm in terms of quantum multiplexors.

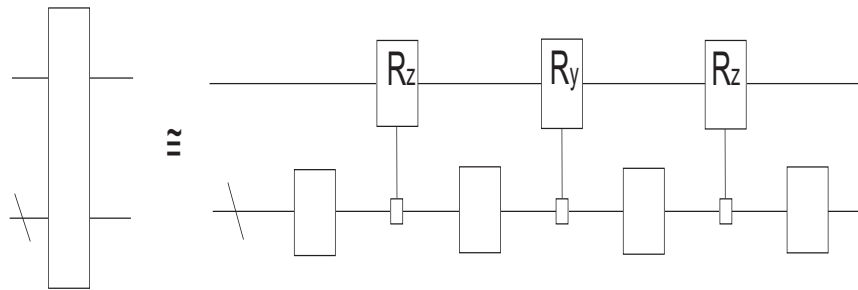


Fig. 4. The Quantum Shannon Decomposition.

Mihai Udrescu in [15] proposed a quantum hardware interpretation of Gajsky and Kuhns Y-diagram [23] (Fig.5). He modified the classical diagram [20] in quantum terms, therefore having three abstracting levels: architectural, unitary and particle. The architectural level corresponds to the algorithm data flow encoded in the overall quantum states. The unitary level is concerned with quantum gate networks (quantum circuits at the gate level), basic quantum unitary transforms, and unitary operators. Finally, the particle level is a technological interpretation related to the physical implementations of the quantum gates, networks and circuits [7]. These abstracting levels could be seen from three different perspectives, or views. The views in the Y-diagram are behavioral, structural and physical. From a behavioral (or functional) view, the quantum circuit is a functional description, without taking into account the implementation issues. In the structural view, the circuit is just the sum of interconnections between basic components, while the physical view is concerned with

the physical aspects of circuit implementation. At the architectural and unitary (logical for classical computation) levels, HDLs (Hardware Description Languages) are able to describe classical and quantum circuits from both behavioral and structural views. Moreover, existing software tools could assist both architectural and unitary synthesis.

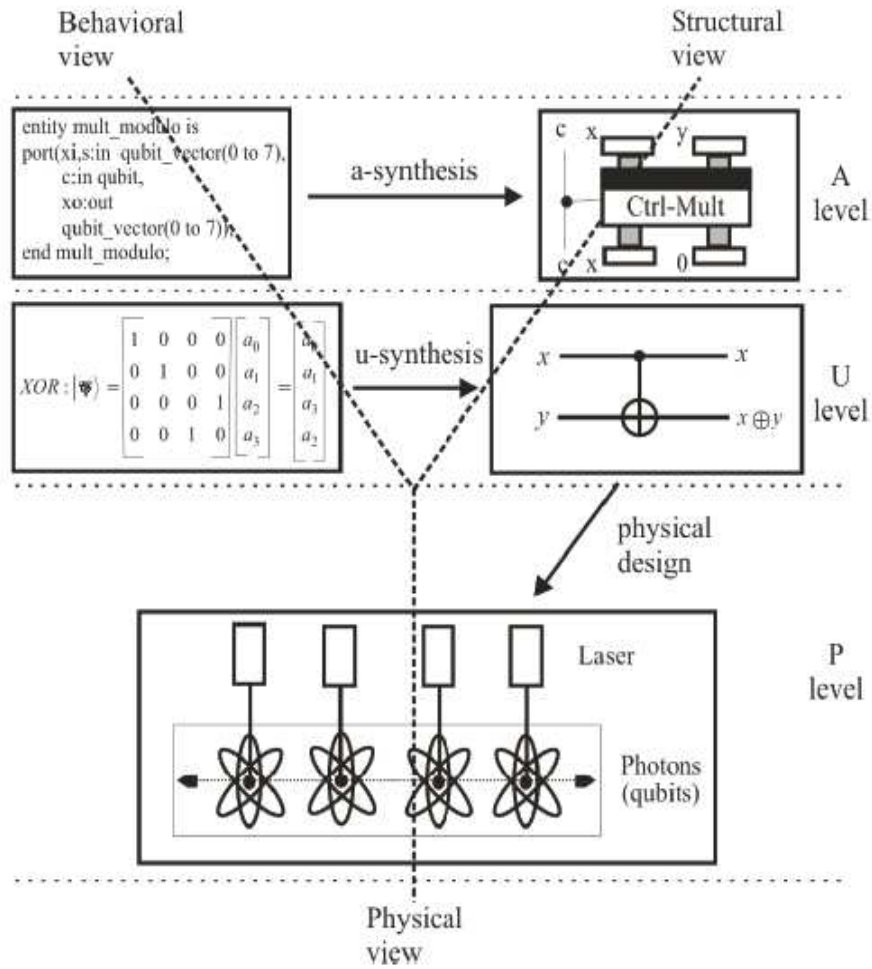


Fig. 5. Quantum interpretation of Gajsky and Kuhns Y-diagram [23].

2.1 First article - A new approach

We present a new approach that facilitates successful synthesis for different quantum circuit types, that are described with matrix elements as real numbers. The desired output function is provided to the synthesizer, and the tool computes whether a quantum circuit (composed of one or more quantum gates from our database), that implements the function exists.

The novelty of the approach lies in splitting the potential circuits in vertical levels called "sections" and horizontal levels called "planes" (Fig.6), for a consistent representation of the genetic algorithm, which will be used for the chromosome definition. Information is exchanged from left to right with the upper wires representing the most significant qubits.

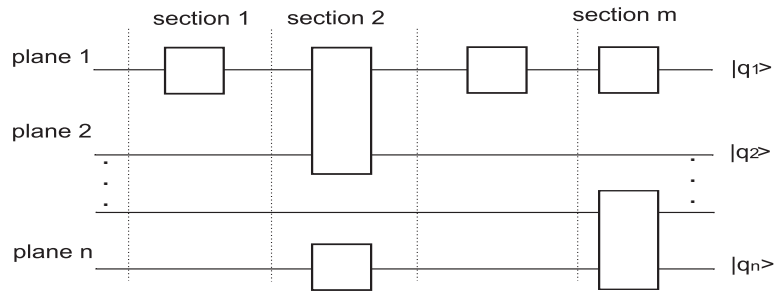


Fig. 6. The quantum circuit view that is used for the new approach.

Potential circuits are represented by individuals (Fig.7 (b)), each containing rows for possible gates on which the tensor product can be applied and one column for the section decomposition. The number of gates on a row is limited by the number of circuit qubits. The number of gate elements within the column is not limited, the level of decomposition being set interactively. Each individual represents a possible solution, which is computed by applying the tensor product for all horizontal rows and then multiplying all the results.

The new algorithm operates with a collection of quantum gates, that are properly encoded as matrices. A dynamic approach for the matrix representation (list-in-a-list) was used. This approach allowed for efficient memory usage. In order to decrease the time required by the tensor product and multiplication, we have created several

dedicated memory locations for intermediate results storage. The special-purpose algorithm [10] is used for the automated search of the best quantum gate composition, in order to implement a given quantum multi-qubit unitary transformation.

A genetic algorithm is used to perform a search in a solution space, without human intervention. Our main focus was to define all the phases required by the algorithm, allowing interactive specification of a minimal set of parameters, mainly those necessary for defining the exit point (algorithm stop condition) and those necessary for increasing or decreasing the solution convergence (depending on mutation/crossover percentage, multiple mutation/crossover percentage). Fig.7 (a) presents our genetic algorithm. An important step in defining the fitness function (as percentage derived from the matrix comparison) was to find the best encoding for the chromosome (plane and section decomposition) by taking into account, at the same time, the matrix dynamic representation.

The new approach is coming from our way of defining the matrix representation (less memory usage), from our way of defining the auxiliary memory spaces (more performance on matrix operators), from our way of defining the chromosome codification (decomposition in planes and sections), from our way of defining the fitness function (as percent from the matrix comparing). Some of them have been already explained through the article, some of them will follow. When encoding the chromosomes (Fig.7(b)), each individual will contain one or more sections, each section containing one or more gates. The number of gates is given by the number of qubits in the circuit, which is computed from the number of matrix elements (i.e. qubits = \log_2 (number of elements)). The number of sections is specified interactively and represents the granularity level of the synthesis (a higher number of sections means that the circuit will be composed of elementary gates, and a small number of sections means that the algorithm will search in order to find more complex gates for synthesis). Depending on this parameter, we may find a solution.

When implementing mutation (Fig. 8 (a))there is less chance to generate an initial population who already contain the expected solution. Usually the algorithm tends to find a local solution and through mutation we allow to the evolution a new path, which may be a good one [3]. For this, we randomly select the chromosome, then we randomly pick a section and random a gate.

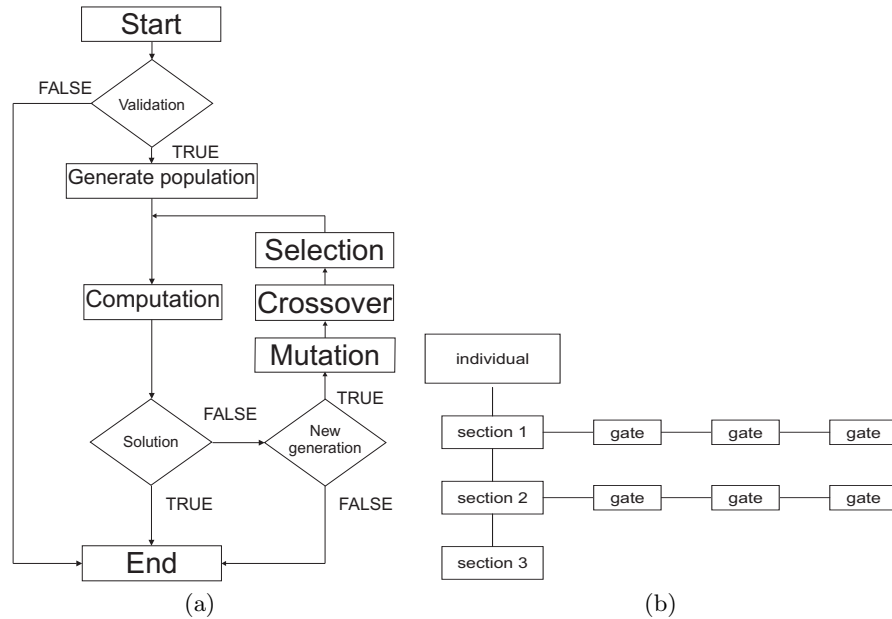


Fig. 7. Genetic algorithm (a) and chromosome encoding (b).

The approach (Fig.7 (a)) randomly selects the chromosome (Fig.7 (b)), then picks a random gate solution. This gate solution will suffer the mutation, thus being replaced by a new randomly generated gate (all gates are defined in the input database). Because we allow multiple mutations, the same chromosome may suffer another mutation within the defined probability (Fig.8 (a)).

In almost all cases, only mutation will not help to find a solution. Therefore, another genetic operator must be implemented to vary the chromosome for the next generation. This operator implements the reproduction of chromosomes.

Also, we implemented the one-point crossover (Fig.8 (b)): two parents are randomly selected together with a point on their sections, and from that point all the gates in between are swapped. The crossover results are used to improve the fitness value of the selected chromosome.

When implementing selection, it is important to eliminate the chromosomes from the generation that manifest a small solution convergence. We used a proportional selection, so that an individ-

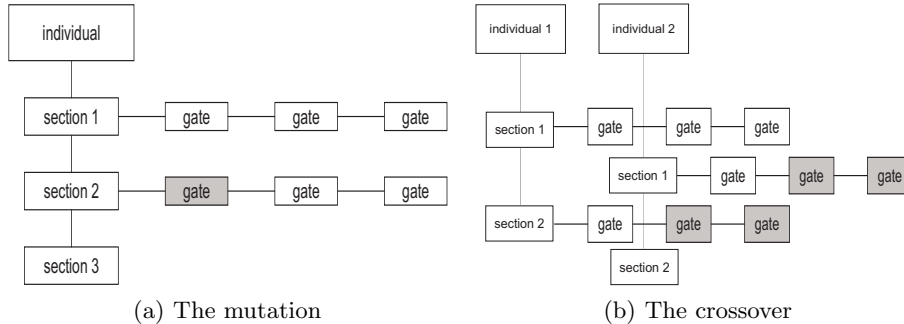


Fig. 8. Genetic operators.

ual with a fitness value smaller than a threshold will be eliminated from the population, and a new one given will be created randomly (intruder in population).

We defined our fitness function as a matching percentage with the given output function, by comparing each matrix element from our chromosome with each corresponding element from the expected solution. For instance, if our chromosome has three out of four matrix elements that are identical with the given output, the fitness function will be 75%.

Many times the evolution look like an open-end process. We cannot guaranty a solution and if a solution is found we cannot guaranty that is an optimum solution. More then this, many times, the genetic algorithm will tend to go into a local solution (premature convergence) which is far away from what is expected. Thus, this is the reason why we must force the algorithm through mutations and crossover to a global solution and that's why we decided to specify this end point as number of allowed generations.

In the quest for automated quantum circuit synthesis, our approach, using Genetic Algorithms, produced different solutions, by interactively changing the parameters of the genetic algorithm, or by just repeating the execution. Only the genetic evolution decide about which solution is found (the evolution discover for us new solutions or just discover again what was already found).

The resources employed by the algorithm (memory and CPU time) increase with the complexity of the output function. This is

due to the number of qubits needed by the algorithm and because of the mathematical operations involved (i.e. the tensor product).

The algorithm will find circuits that implement a given unitary transformation (i.e. the output functions). The tool uses, as input, a database that can be updated interactively. This update is easy to be made because the circuits are described using the matrix notation (in a quite straightforward manner). It's not necessary to describe many kind of gates, because our algorithm will combine them, allowing more complex gates to be used in synthesis.

The major difference, with respect to other approaches, is the way circuits are seen as been split in sections and planes. The optimal coding of the chromosome allows a low complexity synthesis of the quantum circuit. Therefore, our proposed genetic methodology offers superior performance, in terms of time; for instance, the methodology described in [5] is working only on two qubits.

Dynamical memory allocation allow for efficient time and memory consumption, as well as for the synthesis of different types of circuits without any prior parametrization.

Future work will focus on chromosome encoding through graph representation (m-graphs may be suitable), along with an enhanced matrix representation.

Our focus is now on using a tool chain in order to do synthesis. We will start with an algorithm implementation described in an accepted programming language, we will go through that algorithm computing the output function and then using a well defined input database we will start to perform the synthesis with an improved genetic algorithm.

As shown in Fig.9 the algorithm is convergent. If in Fig.9 (a) the number of individuals who have a high fitness is low, in Fig.9 (b) the majority of individuals have high fitness values. This shows that in the end at least a solution will be produced.

Fig.10 shows experimental values obtained by running the genetic algorithm for two/three qubits, for 15 times, at the moment when the solution was found, in terms of number of generations, number of mutations and crossover, as well as the running time (in seconds).

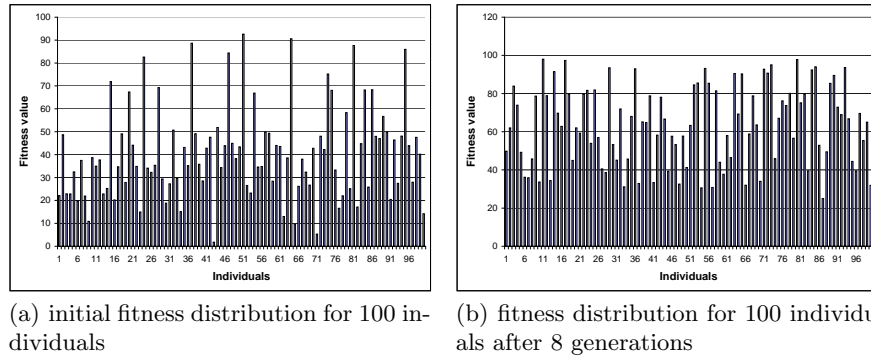


Fig. 9. 2-qubit solution convergence.

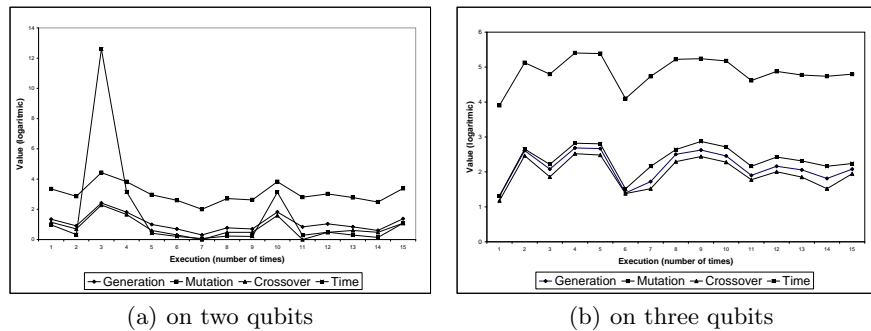


Fig. 10. Experimental results.

3 Project presentation

Writing a thesis is tough work. A thesis is the acquisition and dissemination of new knowledge. In order to acquire this, we shall demonstrate that we understand what the relevant state of the art is, and what the strengths and weaknesses of the SoA are. For someone's work to be acknowledged, there must be a demonstration that suitable and systematic methods were used in order to evaluate the chosen hypothesis. It is important that "new" is not just new to the researcher, but also new to the community.

3.1 Objectives

The main objective is to create a tool chain for the quantum circuit synthesis. We are motivated by the wish to bring together genetic al-

gorithms and quantum computing. This association is created with the scope of performing optimum synthesis. The objective of this thesis is also related to the Advanced Computing Systems and Architectures (ACSA) Laboratory, which aims at fostering the new computing technologies.

Another objective is to continue the work of my colleague Mihai Udrescu, which is a advisor for this particular project within *ACSA* and to create a solid base for quantum technologies in our university (Fig.11).

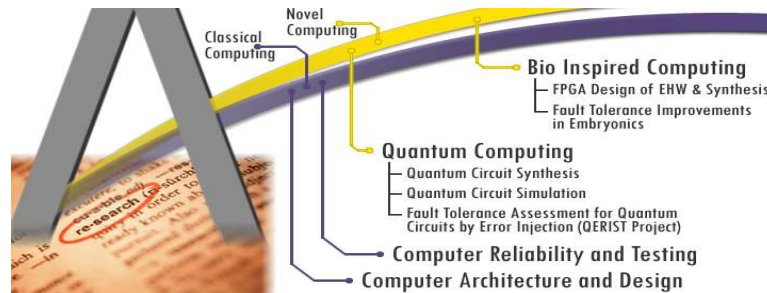


Fig. 11. ACSA laboratory overview.

We want to open a discussion about the quantum synthesis involving also scientists from genetic algorithms, which could be of great importance for computer science in general.

The motivation presented in this section may seem theoretical and pretty much detached from the actual industry problems. But the fact is that the industry is seriously taking into consideration the aspects related to the emerging technologies, and quantum circuits in particular. The industry representatives have quickly reacted to these emerging problems, and founded a global organization called ITRS (International Technology Roadmap for Semiconductors), which is jointly sponsored by European Semiconductor Industry Association, Japan Electronics and Informational Technology Industries Association, Korea Semiconductor Industry Association, Taiwan Semiconductor Industry Association, and Semiconductor Industry Association from U.S.A. As this organization defines its documents, they are about a continuous evaluation of the semiconductor technology requirements, aimed at increasing the performance of the integrated circuits. This effort is supported by industry, suppliers,

academia, research groups, and governments [16]. The results of the ITRS assessments are published as ITRS reports, which are annually updated (Fig.12).

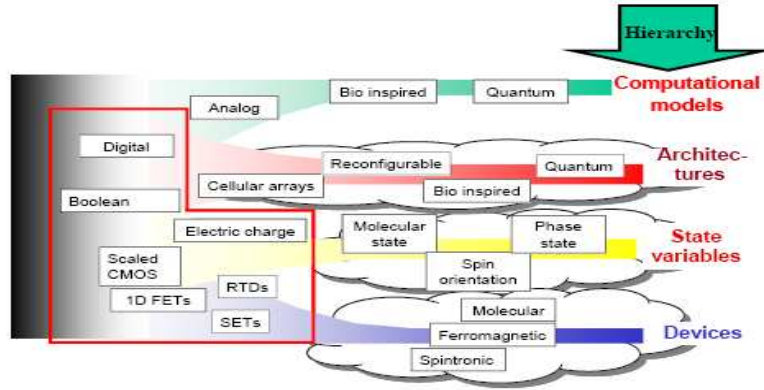


Fig. 12. A Taxonomy for Nano Information Processing.

The device choices in the future will be determined by the following hierarchy: applications \rightarrow computer architecture \rightarrow micro and nanoarchitecture \rightarrow circuits \rightarrow devices \rightarrow materials. It is important to distinguish between the two ways in which the term "architecture" can be used. The term "computer architecture" includes all the systems elements, including software, needed to meet the needs of a given information processing application. "Micro- and nanoarchitecture" refers to the implementation details of how the various computing functions can be organized for high information throughput, minimum expense and energy cost. Currently, most of the relevant publications in this area are concerned with aspects which are best described as nanoarchitectures [16].

3.2 Proposed PhD Thesis Layout

Abstract

1.Introduction The problem statement shall be presented here, and what is its relevance to the studied domain. The structure and the state contributions (3-5 pages) are also described.

2.Background Demonstrate wider appreciation. Motivation. The problem statement and the motivation state, as of how the PhD is to be judged.

- 3.Related Work** Through survey and assessment, analysis, with respect to them work.
 - 4.Analysis** What is the nature of our research problem. Which is his relevance for the scientific and more-important for the industrial communities.
 - 5.Design** How the problem is solved. What are the necessary steps to solve it, in terms of algorithms and methodologies.
 - 6.Implementation and experimental results interpretation** What are the qualitative attributes of our proposed solutions.
 - 7.Critical assessment** State hypothesis, and demonstrate precision, thoroughness, contribution, and comparison with closest rival.
 - 8.Further Work**
 - 9.Summary. Conclusions.** Restate contribution.
- Appendix**
Bibliography

4 Realization

4.1 Implementation steps

There are three major research directions (Fig.13), for further development:

- related work:** we must know all the relevant realizations made in the approached field, so that appropriate result assessment of the scientific results is possible.
- optimization techniques:** we must know the methods that are available for optimization purposes.
- quantum computation:** we must know how the computation process flows in the quantum environment.

There are also some steps needed by the software implementation (Fig.14):

- definition of the HDL language that is used: we shall use a programming language or at least a common accepted formalism for defining quantum computation.
- parser: we must implement a parser for the high level formalism defined in the previous item. This allows for having the same internal representation of the program inputs.

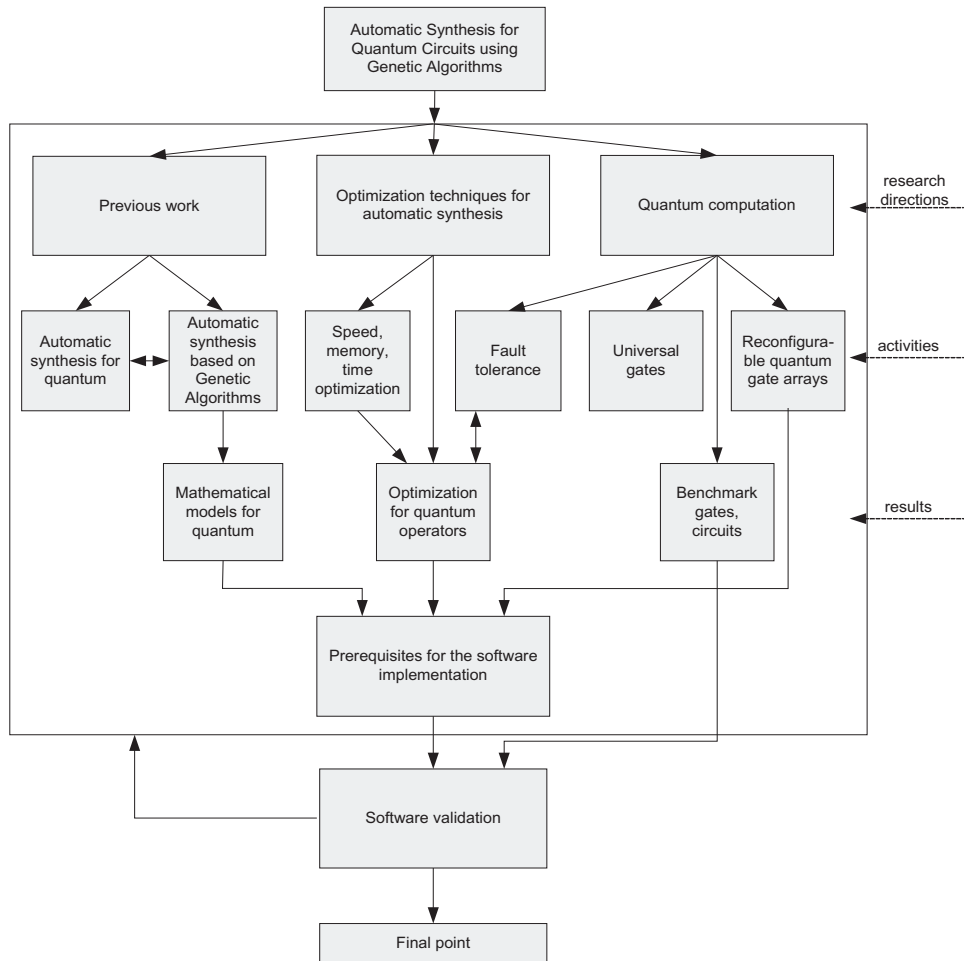


Fig. 13. Proposed research process flowchart.

- data structure: we shall use a proper data structure, because we must perform internal optimization for the automatic synthesis.
- simulation: simulation is necessary for observing quantum circuit output function (we will use an available solution).
- genetic algorithm: is necessary to perform the search process required by the quantum circuit synthesis.

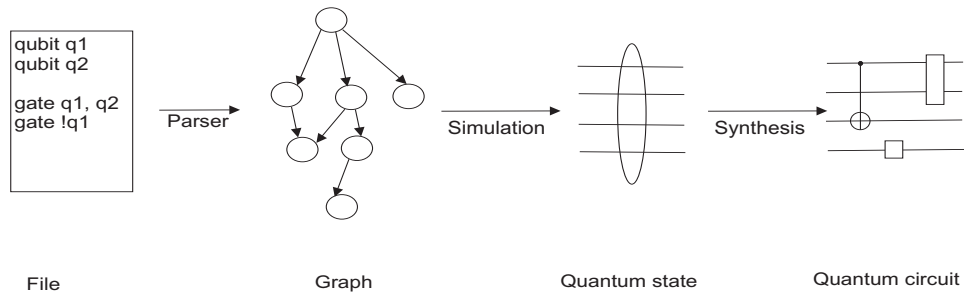


Fig. 14. Software implementation steps.

4.2 Comparing results

Throughout this research process we must compare our results with other reported results. At this moment we identified two major solutions in the simulation field:

- QCSIM is a quantum computation simulator written in C++. It does a discrete-time simulation of 2-level quantum entities (qubits). QCSim reads a file written in a subset of QHDL which includes declaration of qubits, the initial state of the quantum system, and a series of quantum gates and commands [11],[19].
- QuIDDP:High-Performance Quantum Circuit Simulation is a fast, scalable, and easy-to-use computational interface for generic quantum circuit simulation. It supports state vectors, density matrices, and related operations using the Quantum Information Decision Diagram (QuIDD) datastructure [12].

For example, the comparison of the results between the above specified simulators are freely available (Fig.15).

In the issue of quantum synthesis, we still do not have available results, in order to relate.

Benchmark	No. of Qubits	No. of Gates	QuIDDPro		QCSim	
			Runtime (s)	Peak Memory (MB)	Runtime (s)	Peak Memory (MB)
rc_adder1	16	24	0.44	0.0625	N/A	> 2GB
rc_adder2	16	24	0.44	0.0625	N/A	> 2GB
rc_adder3	16	24	0.44	0.0625	N/A	> 2GB
rc_adder4	16	24	0.44	0.0625	N/A	> 2GB
9sym1	12	29	0.2	0.0586	8.01	128.1
9sym2	12	29	0.2	0.0586	8.02	128.1
9sym3	12	29	0.2	0.0586	8.04	128.1
9sym4	12	29	0.2	0.0586	8	128.1
9sym5	12	29	0.2	0.0586	7.95	128.1
ham15_1	15	148	1.99	0.121	N/A	> 2GB
ham15_2	15	148	2.01	0.121	N/A	> 2GB
ham15_3	15	148	1.99	0.121	N/A	> 2GB

Fig. 15. Comparison result between QuIDDPro and QCSim.

4.3 Actual realization

1. Definition of HDL language used: we decided to use the QHDL (Design Language for Quantum Computing). It is taken from report "Final Report-A Design Language for Quantum Computing" ORA TR-03-0001, Odyssey Research Associates, Inc., Ithaca, New York.

This design language was used for the first time by the NIST-USA (National Institute of Standards and Technology), they started to develop a quantum simulator called QCSim [11], between 2002-2004. The same language is cited by the QuIDDPro simulator [12]. The ATC-NY (Architecture Technology Corporation) wants to developed the QHDL offering: a formal, user-friendly, machine-readable language, QHDL, that is capable of describing the full range of quantum algorithms; a suite of tools for working with QHDL; an on-line archive of algorithms expressed in QHDL.

All of this prerequisites, together with the powerful formalism from the QHDL, made us to choose as HDL for our goal the QHDL description. By combining flexibility and precision with features for scalability, QHDL will be a bridge between current research and future industrial applications. QHDL will provide two

new features: a shared language of interchange (communication) and a common substrate on which to build quantum computing tools (automation) [18].

Declarations are as follows:

- $DECL := "variable" NAMELIST : "" qubit";$
- $NAMELIST := VAR_NAME | VAR_NAME , " NAMELIST$
- $VAR_NAME := defnofvariablename$
- $INIT := " = " CONSTKETLIST";$
- $CONSTKETLIST := AMPKET | AMPKET - " CONSTKETLIST | AMPKET + " CONSTKETLIST$
- $AMPKET := [AMPLITUDE]KET$
- $AMPLITUDE := FLOAT | FLOAT " I" | " I"$
- $FLOAT := floatingpointnumber$
- $KET := "|" ("0" | "1") + " > "$

We implemented a parser based on the above knowledge, our result being included in the Qapp software (this is, at least for the moment, the name of our synthesis application) as is shown in the Fig.17. Our parser respects the QHDL formalism, being able to read names, numbers and punctuation. After parsing, we are able to create an internal list that contains the qubit declarations (as variables) and the list with the quantum gates used and them parameters. We are able to parse the following quantum gates:

- (a) cnot
 - (b) copZ
 - (c) Hadamard
 - (d) measure
 - (e) noisyHadamard
 - (f) traceOver
 - (g) Toffoli
 - (h) opX
 - (i) opY
 - (j) opZ
 - (k) probXYZ
 - (l) chadamard
 - (m) genAmpDamp
 - (n) BSWDepolCPhase
 - (o) rotateY
2. File example: the input file has three main sections. The first one describes the qubits used in the application, the second section is the initial given state. The last section specifies the functional description of the algorithm (Fig.16).

```

# A Quantum circuit to demonstrate the BB84 communication protocol with
# just a single bit (in the presence of eavesdropping).

# Alice encodes a polarized photon and transmits it. The basis
# is either rectilinear (+, encoded as |0> or |1>) or diagonal
# (X, encoded as |0>+|1> or |0>-|1>).

variable Photon:qubit;
variable AInfo:qubit;
variable ABasis:qubit;
variable EBasis:qubit;
variable BBasis:qubit;
variable BasesEq:qubit;
variable AInfoNEqBInfo:qubit;
variable Error:qubit;

=|00000000>;

# Alice chooses to send a 0 or a 1
hadamard(AInfo);
measure(AInfo); # to collapse into classical random choice
# Put that information on the photon
cnot(AInfo, Photon);

# Alice randomly picks the rectilinear or diagonal basis
hadamard(ABasis);
measure(ABasis);
# ... and rotates the photon if she picked the diagonal basis
chadamard(ABasis, Photon);

```

Fig. 16. Example of an QHDL input file.

3. Initial state: it is computed by normalization of initial quantum qubits. This operation is necessary, in order to have a defined starting point for the following operations. The normalization provide a unique initial state for the system (Fig.17).
4. Quantum gate list: we are able to create a list with the recognized quantum gates, each gate showing the received arguments. The gate list is necessary at simulation, simulation taking all gates from the beginning to the end, in the given order, provided by the list.

5 Potential contributions

Personal contributions:

1. The master dissertation on the same topic as the proposed PhD thesis "Techniques for Automatic Synthesis of Quantum Circuits" presented in 2005 [17] (Fig.18).
2. Our first article submitted to ICANNNGA '07, International Conference on Adaptive and Natural Computing Algorithms, 11-14 April 2007.
3. A software program which performs automatic synthesis for quantum circuits, described in matrix representation with real elements, which is available in a download form [10]

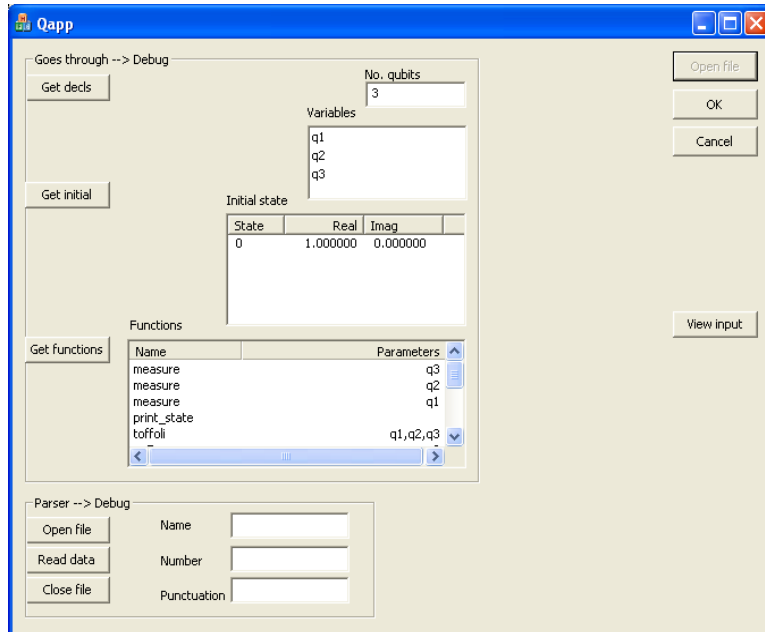


Fig. 17. Software implementation - parser and normalization.

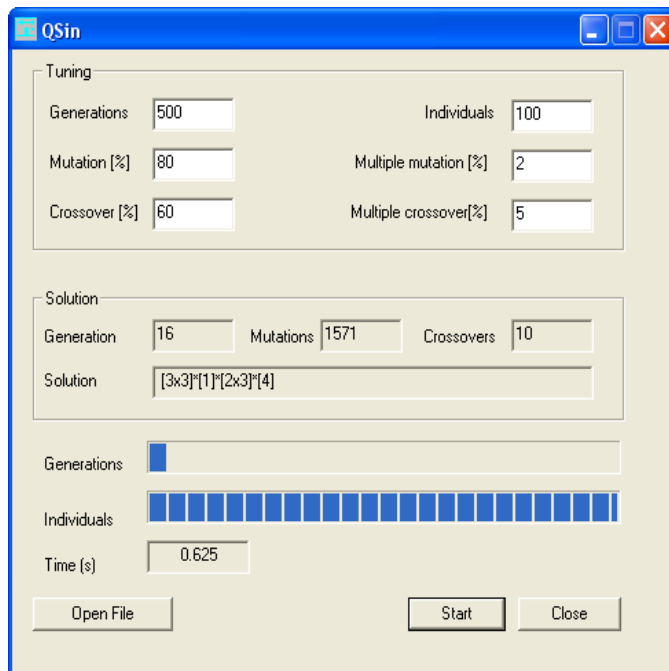


Fig. 18. Quantum Synthesis developed during the Master program.

Other conferences where we shall submit articles:

- Genetic and Evolutionary Computation Conference (GECCO), July 2007, London, UK (ISI ranked)
- International Conference on Computing Frontiers (CF), 7-9May 2007, Ischia, Italy (DPLB ranked)
- Quantum Information, 10-29 June 2007, Benasque, Spain (ISI ranked)
- EuroGP, 11-13 April 2007, Valencia, Spain (ISI ranked)

Based on the research results, we estimate to write the following reports and articles (Fig.19):

Report 1: Quantum Synthesis Optimization. This report shall reveal how we can optimize the synthesis. For this, we shall present a new approach on quantum data structure optimization using graphs. We propose this report because is important to optimize before going the make the synthesis. We shall use, as relative, the digital circuit synthesis, as is known at this time, Synthesis and Optimization of Digital Circuits, Giovanni De Micheli, Standford University, McGraw-Hill Inc., 1994.

Report 2: Quantum Synthesis with Genetic Algorithms. This report shall present experimental results of the quantum circuit synthesis, enforced with theoretical demonstration. For the quantum computation we shall use, as relative, Quantum Computation and Quantum Information, Michael A.Nielsen and Isaac L. Chuang, Cambridge University Press, 2000.

PhD Thesis: Developing Automatic Synthesis Methodologies for Quantum Circuits using Genetic Algorithms

Articles: about our research, more than three (ongoing 2006-)

Software program: Quantum Circuit Synthesis

6 Conclusion

The process of bridging the gap between quantum computing and genetic algorithms didn't reach its maturity, been an area that needs new techniques for development. As motivation, the following achievements have to be mentioned:

- An automatic synthesis software program developed during Master program.

	Year 2006	Year 2007				Year 2008			
	Quarter 4	Quarter 1	Quarter 2	Quarter 3	Quarter 4	Quarter 1	Quarter 2	Quarter 3	Quarter 4
Project									
Report 1									
Report 2									
PhD Thesis									

Fig. 19. Study planning.

- A first article submitted.
- Parser implementation that recognize the QHDL formalism.
- Normalization implementation in order to compute the initial system state.

Also, following the already mentioned results we aim at attaining the following objectives, throughout the Ph.D. Research Program:

- Definition of an abstract model used to define quantum states.
- Definition of optimization methods with/without genetic algorithms.
- Definition of methodology for automatic synthesis of quantum circuits.

7 References for study

1. A. Narayanan, M. Moore, "Quantum-Inspired Genetic Algorithms". IEEE International Conference on Evolutionary Computation (ICEC-96), Nagoya, Japan pp. 61-66 (May 1996).
2. A. Thompson, "Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution", Springer-Verlag distinguished dissertation series, (1998).
3. B. Rylander, T. Soule, J. Foster, "Computational complexity, genetic programming, and implications", Proc. 4th EuroGP, pp. 348-360 (2001).
4. B. Rylander, T. Soule, J. Foster, J. Alves-Foss, "Quantum Evolutionary Programming", In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), pp. 1005-1011 Morgan Kaufmann (2001).
5. C.P. Williams, A.G. Gray, "Automated design of quantum circuits", In Proc. of 1st NASA International Conference on Quantum Computing and Quantum Communications QCQC '98, LNCS No. 1509 Springer-Verlag, pp. 113-125, Palm Springs, CA (1998).

6. D. DiVincenzo, "Two-bit gates are universal for quantum computation", *Phys. Rev. A* (AT5101) (1995).
7. G. De Micheli, "Design and Optimization of Digital Circuits", McGraw-Hill, (1996).
8. G. F. Viamontes, I. L. Markov and J. P. Hayes, "Graph-based Simulation of Quantum Computation in the Density Matrix Representation", *Quantum Information and Computation*, vol.5, no.2 pp. 113-130, February 2005; quant-ph/0403114.
9. G. F. Viamontes, I. L. Markov and J. P. Hayes, "Improving Gate-Level Simulation of Quantum Circuits", *Quantum Information Processing* vol. 2(5), October 2003, pp. 347-380.
10. G. Yang, W.N.N. Hung, X. Song, M. Perkowski, "Exact Synthesis of 3-Qubit Quantum Circuits from Non-Binary Quantum Gates Using Multiple-Valued Logic and Group Theory", *Proceedings Design Automation and Test in Europe (DATE) Conference*, Munich, Germany, Vol.1, pp. 434-435 (2005).
11. G.A. Girdali, R. Portugal, R.N. Thess, "Genetic Algorithms and Quantum Computation", ArXiv:cs.NE/0403003 (March 2004).
12. J. Koza, "Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems", Technical Report STAN-CS-90-1314, Department of Computer Science, Stanford University (1990).
13. J. Koza, "Genetic programming: On the programming of computers by means of natural selection", MIT Press, Cambridge, MA (1992).
14. J. Zhang, J. Vala, S. Sastry, and K. B. Whaley. Optimal quantum circuit synthesis from Controlled-U gates. *Phys. Rev. A* 69, 042309 (2004), E-print quant-ph/0308167
15. K-H. Han, J-H. Kim, "Genetic Quantum Algorithm and its Application to Combinatorial Optimization Problem", In *Proc. of the 2000 Congress on Evolutionary Computation* (2000).
16. K-H. Han, J-H. Kim, "Quantum-Inspired Evolutionary Algorithm for a Class of Combinatorial Optimization", *IEEE Transactions on Evolutionary Computation*, vol. 6, No. 6, pp.580-593 (2002).
17. L. Spector, H. Barnum, K.J. Bernstein, N. Swamy, "Finding a Better-than-Classical Quantum AND/OR Algorithm Using Genetic Programming", In *Proceedings of 1999 Congress of Evolutionary Computation*, Piscataway, NJ, IEEE Press vol 3 pp.2239-2246 (1999).
18. L. Spector, H. Barnum, K.J. Bernstein, N. Swamy, "Genetic Programming For Quantum Computers", *Proceedings of the 3rd Annual Conference on Genetic Programming*, Madison, Wisconsin, USA, Morgan Kaufmann Publishers, pp. 365-373, (1998).
19. L. Spector, H. Barnum, K.J. Bernstein, N. Swamy, "Quantum Computing Applications of Genetic Programming", In *Advances in Genetic Programming*, volume 3, chapter 7, pp. 135-160, MIT Press (1999).
20. L. Spector. *Automatic Quantum Computer Programming. A Genetic Programming Approach*. Kluwer Academic Publishers, (2004)
21. M. Lukac, M. Perkowski, H. Goi, M. Pivtoraiko, C.H. Yu, K. Chung, H. Jeech, B. Kim, and Y. Kim, "Evolutionary Approach to Quantum and Reversible Circuits Synthesis". *Artif. Intell. Rev.* 20, 3-4 (Dec. 2003), 361-417.

22. M. Lukac, M. Perkowski, "Evolving Quantum Circuits Using Genetic Algorithm". In Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware (Eh'02) (July 15 - 18, 2002). IEEE Computer Society, Washington, DC, 177.
23. M. Udrescu, L. Prodan, M. Vladutiu, "A new perspective in simulating quantum circuits", Proc. LBP AAAI GECCO pp.283-290, Chicago IL (2003).
24. M. Udrescu, L. Prodan, M. Vladutiu, "Improving Quantum Circuit Dependability with Reconfigurable Quantum Gate Arrays", 2nd ACM Conference On Computing Frontiers, ACM Press, pp. 133-144 (Ischia, Italy, 2005).
25. R. Cleve, D. Gottesman, "Efficient computations of encoding for quantum error correction," Phys. Rev. A 56, 76-82 (1997).
26. V. Vedral, A. Barenco, A. Ekert, "Quantum Networks for Elementary Arithmetic Operations", Online preprint quant-ph/9511018, (1996).
27. V. V. Shende, S. S. Bullock, I. L. Markov, "A Practical Top-down Approach to Quantum Circuit Synthesis," Proc. Asia and South Pacific Design Automation Conference, pp. 272-275, Shanghai, China, (2005) quant-ph/0406176.

References

1. Artur Ekert, Patrick Hayden and Hitoshi Inamori: Basic Concepts in Quantum Computation. Centre for Quantum Computation, University of Oxford (16 January 2000).
2. Andrea Malossini, Enrico Blanzieri and Tommaso Calarco: QGA, A Quantum Genetic Algorithm. Technical Report DIT-04-105 (December 2004).
3. J. H. Holland: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975).
4. K. Svore, A. Cross, A. Aho, I. Chuang, I. Markov: Toward a Software Architecture for Quantum Computing Design Tools. IEEE Computer (Jan. 2006).
5. Prashant Singh: Evolving Quantum Circuits using Genetic Algorithm. Quantum Physics, quant-ph/0511036 (2005).
6. Vivek V. Shende, Stephen S. Bullock, Igor L. Markov: Synthesis of Quantum Logic Circuits. Quantum Physics, quant-ph/0406176 (2004).
7. Michael Nielsen, Isaac Chuang: Quantum Computation and Quantum Information. Cambridge University Press (2000).
8. Lee Spector: Automatic Quantum Computer Programming. A Genetic Programming Approach. Kluwer Academic Publishers (2004).
9. Mihai Udrescu, Lucian Prodan, Mircea Vladutiu: Implementing Quantum Genetic Algorithms: A Solution Based on Grover's Algorithm. Proceedings of the 3rd Conference on Computing Frontiers CF'06, pages 71-82 (2006).
10. QSin a quantum synthesis program developed by Cristian Ruican, download version available on: http://www.cristian.ruican.home.ro/index_files/Site.zip (2006).
11. QCSim web page, <http://hissa.nist.gov/black/Quantum/qcsim.html>
12. QuIDDPro web page, <http://vlsicad.eecs.umich.edu/Quantum/qp/index.html>

13. John Preskill web page, <http://www.theory.caltech.edu/people/preskill/>
14. Centre for Quantum Computation web page www.qubit.org
15. Quantum Circuits Engineering: Efficient Simulation and Reconfigurable Quantum Hardware, PhD Thesis, Mihai UDRESCU-MILOSAN, November 25, 2005
16. Emerging Research Devices with a New Section on Emerging Research Materials, International Technology Roadmap for Semiconductors (ITRS) Report (2004 Update), <http://www.itrs.net/Links/2005ITRS/ERD2005.pdf>, (2005).
17. Tehnici de Implementare a Sintezei Automate de Circuite Cuantice, Master dissertation, UPT, July 2005.
18. QHDL: A Design Language for Quantum Computing <http://www.atcorp.com/systemdevelopment/quantcomp.html>
19. Paul E. Black and Andrew W. Lane, Modeling Quantum Information Systems, Proc. Quantum Information and Computation II, Defense and Security Symposium, Orlando, Florida, (April 2004), SPIE, <http://hissa.nist.gov/black/Papers/modelQuantSimSPIE04.html>
20. M. Udrescu, L. Prodan, M. Vladutiu, A new perspective in simulating quantum circuits, Proc. LBP AAAI GECCO pp.283-290, Chicago IL (2003).
21. QCL, <http://tph.tuwien.ac.at/oemer/qcl.html>
22. List of Quantum Computation Simulators http://www.quantiki.org/wiki/index.php/List_of_QC_simulators
23. G. De Micheli, Design and Optimization of Digital Circuits, McGraw-Hill, (1996).