Field Programmable Gates Array implementation of quantum computation structures

Laurentiu-Mihai Ionescu [1)], Alin Mazare [1)], Daniel Visan [1)], Nadia Belu [2)], Adrian Lita[3)]

Department of Electronics, Computers and Electrical Engineering
[2)] Department of Manufacturing and Industrial Management,
[3)] Polytechnic of Bucharest, Bucharest, Romania
University of Pitesti, Pitesti, Romania
laurentiu.ionescu@upit.ro

***Abstract**: The use of quantum computation, according to theory, can bring substantial improvements to future computing systems. The problem is the still experimental and hardly accessible technology of their implementation. The article presents an original solution for implementing a quantum computing structure emulator. Unlike other types of emulators of this kind, the proposed solution is not only a qualitative assessment of the performance of the quantum algorithms, but also a method of their effective implementation using existing technologies. The paper will present the solution of representation of the qubit, quantum gates implementation as well as the quantum interferometer for a system with two qubits. To illustrate the proposed solution, we will also present the method of implementing a search algorithm: determining the parity of a Boolean function.*

1. INTRODUCTION

The quantum calculus idea does not necessarily imply advanced quantum mechanics knowledge in the same way the use of artificial neural networks does not necessarily imply the knowledge of a neurologist or the use of genetic algorithms does not involve the experience of a genetic engineer or a microbiologist. It is a model that is used to increase the performance of a system. In the following lines we shall give a brief introduction to the concept and how it can improve performance.

The information unit of the quantum calculus is the qubit (quantum bit - the equivalent of the conventional digital system bit). A qubit can have state 0, state 1 (like a bit), or a super-position between the two states. A super-position means that the qubit can have both state 0 and state 1 with a certain probability. Super-position is something that, at first sight, does not have an equivalent in digital systems.

At this point, I would like to highlight the difference between the three types of computing systems: the deterministic conventional system, the statistical system and the quantum system. The conventional deterministic always has a fixed determined result: for example, a compare module of two numbers always has a predictable result: if the two numbers are equal, the result will be 1 while if the two numbers are different, the result is 0. The statistical system is one whose output may be different under the same input conditions. For example, a binary output can be 0 with a probability of 0.3 and it can be 1 with a probability of 0.7. This means that under the same input conditions, if we repeat the operation 10 times, we will have, in theory, in 3 cases 0 and in 7 cases 1 on the output. The quantum system must have two operation modes: one in which it evolves in a quantum way and one in which the measurement takes place, and in which the system becomes deterministic or statistical. During quantum evolution, the system is in all possible states at the same time, in each state with a certain probability. When we make the measurement, the system becomes statistical or even deterministic with the behavior that we have shown above.

For example, when the system evolves in a quantum way, the system can be both in 0 and 1, less in 0 (0.3 probability) but more in 1 (0.7 probability) – but as we have mentioned before, it is in both states. When we make a measurement (thus, we want to find the value of output), the system will turn into a statistical one - the output will be 0 with the

probability of 0.3 or 1 with the probability 0.7. What does quantum computing mean? It means that by determining a proper quantum evolution sequence so that, ultimately, when we measure, we identify a deterministic system rather than a statistical system. And the answer, of course, is the right one. Quantum evolution is based on the concept of interference. A block diagram of an interferometer (can be optical – with photons or with electrons) is shown in the figure below.

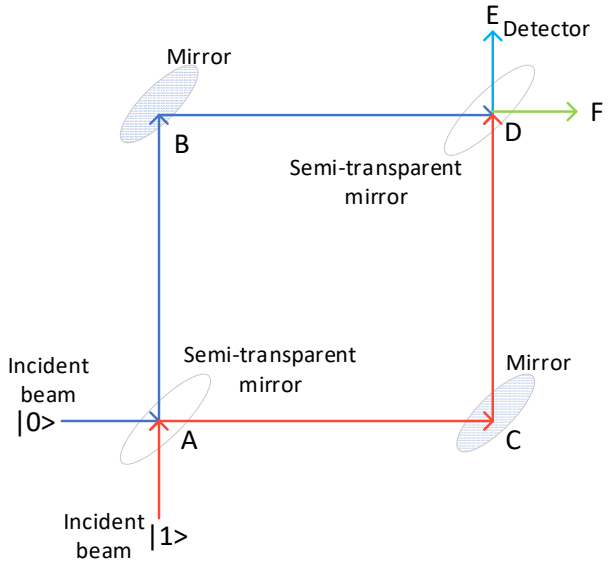

Fig. 1. Optical interferometer block-diagram

Point A is a semi-transparent mirror. An incident beam is divided into two beams: one reflected and one refracted. Further on, on each way, the two resulting beams arrive in normal mirrors (points B and C). They will recompose at point D, where there is another semi-transparent mirror. In points E and F there are two measuring instruments (e.g. detectors). The probability of detecting points E or F is given by a module in which the two beams are composed (their correlated sum). If the beams are phase, then their amplitudes gather, if they are in anti-phase, then they are subtracted. So along with the amplitude of the two beams, their phase is also very important. If on the two interferometer paths: A, B, D or A, C, D, other device interferes then produce supplementary phase-shifting and the way in which the beans will be detected in E or F will be different.

The conclusion is that for each state (interferometer arm) we have a value that must express both amplitude and phase of bean. The most appropriate form we can represent this is by using complex numbers. So, a quantum system with a qubit is represented in the form (a, b) - the Heisenberg notation or a <0 | + b <1 - Schrodinger's notation. Both notations show the same thing: the complex number "a" is associated with with the state 0 of the qubit, while the qubit state 1 is associated with the complex number "b". Any complex number can be represented as $a = A\, e^{i\phi}$ Here, A = | a | represents the complex number module or the amplitude while "phi" represents the complex number phase. When making a measurement, $|a|^2$ is the probability of finding the system in state 0 while $|b|^2$ is the probability of finding the system in state 1.

The advantage of using quantum computing systems is being analyzed by specialists from three points of view: that of the number of problems that can be solved; that represented by the complexity of the circuit networks involved in the implementation; and that of the response time.

Number of problem solved:

The heuristic or pseudo-heuristic algorithms implemented on statistical systems can solve a higher class of problems than deterministic systems. Thus, they can solve the class of problems that can be solved by the deterministic systems plus the class of problems that cannot be solved by deterministic systems.

That is why we can say that the set of problems solved by statistical systems (BPP – bounded error probabilistic polynomial) includes the problem solved only by the deterministic systems (P - polynomial).

On the other hand quantum computing systems (BQP – bounded error quantum probabilistic polynomial) convert in statistical systems when the measuring takes place [1]. So, as it is shown in [2]:

$$P \subseteq BPP \subseteq BQP \quad (1)$$

Complexity of circuits:

An analysis of the complexity of the systems involved in problem solving is a research topic. For deterministic networks, there appears to be a variation of complexity of type $O(n^d)$ - where "n" is the number of operands' bits and "d" is the number of operands. For example, a problem of searching for prime numbers in a string can be solved in a minimum number of $O(n^d \log n)$. There is a heuristic algorithm that can solve the same problem in $O(n^3 \log(1/e))$ where "e" is the probability of an error occurrence in determining the result. On the other hand, when using Shor's quantum search algorithm, it would appear that

only $O(n^2 \log n \log (1/e))$ quantum gates [3] are required. These ratings determine the number of operations and the number of virtual computing units that are involved in these operations.

Response time

Statistical systems have a problem concerning the response time: it can be much higher than in the case of deterministic systems. Moreover, the response time cannot be determined - it is characterized by variations which are sometimes significant.

The Grover quantum search algorithm [4] returns such a result after an approximate number $\pi/4$ SQR(n) of searches with a probability of at least $1 - (1/n)$. For example, in a memory with 1k locations, we would have an average of 500 deterministic searches (between 1 and 1000) and only 25 quantum searches, after which we obtain the result with the accurate probability of 0.999!

Implementation solutions

The extraordinary performance that a quantum calculus system (quantum computer) would bring is no longer a secret. The major advantage is reducing the number of queries and thus the time of search. For example, a quantum search for any record in a collection of unordered data will reduce the number of necessary accessing requests by at least 2 times. We say "at least" because, somewhat paradoxically, the higher the number of data in the collection, the higher the total queries / total data drops [5]! The main issue that remains is related to the implementation methods of quantum computing modules. Two main technologies are used: the photons one and the electrons one. While photon technology allows the interferometer to be implemented and the propagation of the correlated photons [6] encounters difficulties in implementing the quantum gateways. On the other hand, the electron technology allows for the easy implementation of the quantum gateways [7], but there are difficulties in correlating electrons and propagating them.

There are also other technologies that attempt to implement quantum computing structures, listed in several synthesis papers, for example in [8]. Up to date are also the trials of implementing the quantum computing structures, using molecular structures and their chemical activity, as it is shown in [9].

The emulation of quantum computing structures through software technologies – tools for scientific world or even quantum programming languages [10] or the use of hardware for full quantum emulator as is presented in [11] or for a hardware accelerator for a quantum framework emulator [12] is a very recent field of research but whose results mainly concern the quantification of quantum algorithm performance rather than its effective implementation. This is because there is a limitation on implementations with existing hardware and software technologies: the impossibility of describing the quantum superposition. The solution, proposed by us within this article, is a possible implementation of quantum structures using existing technologies - digital FPGA modules. This proposed solution seems to solve the problems of superposition representation in digital systems. As it shall be seen in the paper, the implementation of the quantum emulator does not only provide a quantitative evaluation of the quantum algorithms, but even an optimal implementation of them. As later presented, to demonstrate the functionality of the solution, we shall implement simple quantum search algorithms - determining by one query the type of a binary function with two inputs and an output.

2. DESCRIPTION OF THE PROPOSED SIMULATION METHOD

The solution suggested by us is based on the qubit emulation by two digital periodic signals - each signal corresponds to a state of the qubit. As it can be seen in Figure 2, the signal fill factor is the complex number module associated with the state while the phase of the signal is the phase of the complex number. The advantage of this representation is that it allows easy implementation of the correlated amount between two or more such signals.

Figure 3b illustrates this. If we have a prime psi bean and a second psi beam (shifted from the first one) then the psi result is the correlated amount between the first two using a simple logic gate. As we have seen, the correlated amount is an important component of the interferometer. Any quantum algorithm involves the use of an interferometer.

As it is also shown in [5] quantum computing operations are performed by the so-called quantum gates. Any quantum gate is part of one of the two large classes of gates: U gates (which involve state-level transformations) and shift gates, which involve phase-

shift signals. Both classes of gates can be easily implemented using our representation.

Using the Heisenger representation, a quantum gate can be mathematically modeled by a square matrix NxN where N represents the dimension of the qubit vector in the Heisenberg representation.

For example, the matrix U = (0,1; 1,0) represents the U modeling transformation of the NOT type for a qubit. Thus, the qubit (1,0) is converted to (0,1) if we multiply it with the U matrix. This is interpreted as it follows: if the qubit has the state 0, then it will have the state 1 after transformation.

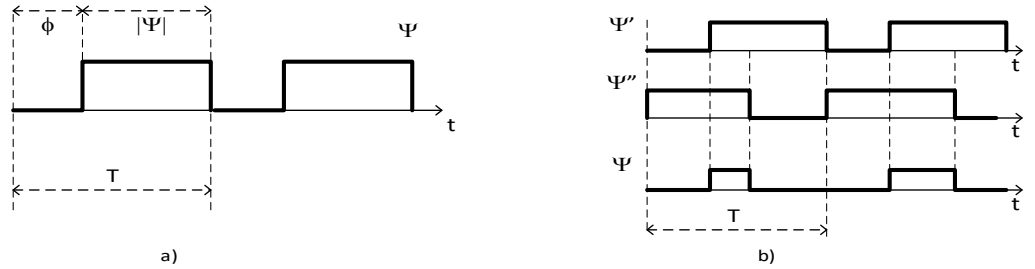

Fig. 2. Codification of qubit (equivalent of bit in quantum computation) using a digital signal

Figure 3 illustrates how the two classes of gates can be implemented with our solution. The U transformations are simple signal routings - for example, a quantum NOT gate will simply route input 0 to output 1 'and input 1 to output 0'. Phase shift circuits are controlled delay lines of the input signal: as it can be seen in the figure, we can have a phase shift for each state - 0 or 1 logic.

Fig. 3. Codification of qubit (equivalent of bit in quantum computation) using a digital signal

Figure 4 shows the implementation mode for a function with two inputs of the type:

$$(x,y) \rightarrow (x,|f(x) + y|_{mod\ 2}) \quad (2)$$

Where f (x) represents a function of the form: f(x) = not(x).

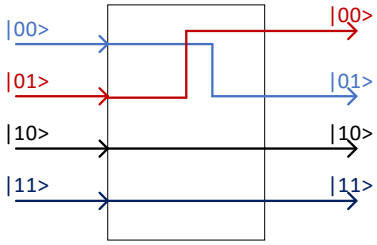

Fig. 4. Diagram of 2 inputs function quantum gate used in our paper given in the expression (2)

The interesting part is that only a signal routing of 00, 01, 10 and 11 is sufficient to implement this function. If two or more inputs arrive on an output then the output will be represented by the correlated sum of the inputs - for more than two inputs, we shall apply the majority type where signals are present. For example, if we have 3 inputs that have no signal and a 4th one where we have a signal, then we do not apply the majority principle but only we will have the 4th signal output. On the other hand, if we have 4 signals present, out of which 3 are phase-shifted with PI and the 4th one with phase 0, then the output will be a phase-shifted signal with PI.

An important circuit in quantum computation is the Hadamard transformation. A general, recursive expression of the Hadamard transformation (without normalization) is as it follows:

$H_n = (H_{n-1},H_{n-1};H_{n-1},-1\ H_{n-1})$ where "n" represents the number of qubits to which the transformation is applied, with n> 1. If n = 1 then $H_1 = (1,1;1,-1)$.

For example, if n = 2 then $H_2 = (1, 1, 1, 1, 1, -1, 1, -1, 1, 1, -1, -1, -1, -1, 1)$.

This is the modeling of the semitransparent mirror presented at the interferometer. The circuit involves both state transformations and phase transformations (phase-shifting with PI). The figure below (fig.5) shows how the Hadamard transformation was

implemented for 4 signals (corresponding to states 00, 01, 10 and 11).

3. EXPERIMENT

To demonstrate the system performance, we shall illustrate how to implement a search algorithm and compare it to the classic solution.

Search problem:

It is given a function f(x), x is a binary input. The task is to create a system in which the function parity is determined:

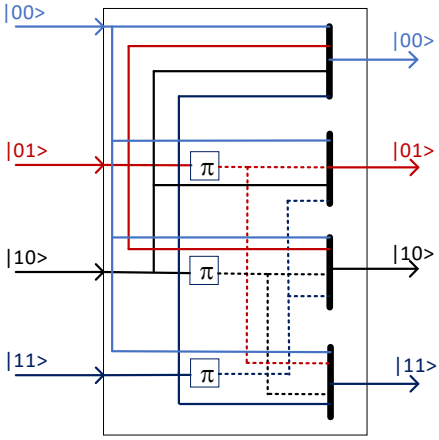

Fig. 5. Diagram of 2 inputs Hadamard gate

If the function is of the type f(x) = 0 or f(x) = 1 then its parity is even. If the function is f(x) = x or f(x) = Not(x) then its parity is odd. To study the efficiency of the calculation method by the model we developed, we had to develop a system that does this in a classic way: we designed a finite state automatic that sends each state to the input (0 or 1) and on the output we get the answer of the function f(x). Finally, we will compare the outcomes obtained and see if they are equal or if they are different. A block diagram of the classic system is shown in the figure below.

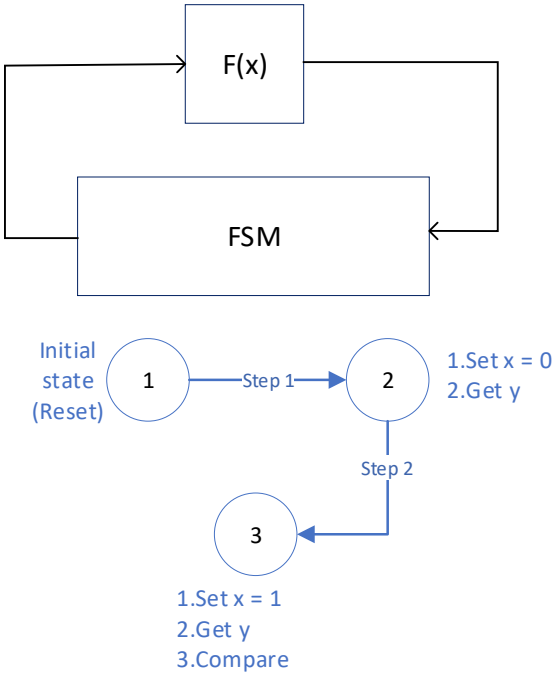

Fig. 6. Diagram of classic implementation of search problem

The quantum emulated system, which gives the answer to the same problem by adding a “y” input with the value f(x) and passing “x” and “y” through two "semi-transparent mirrors" - the Hadamard transformations - is shown in the figure below.

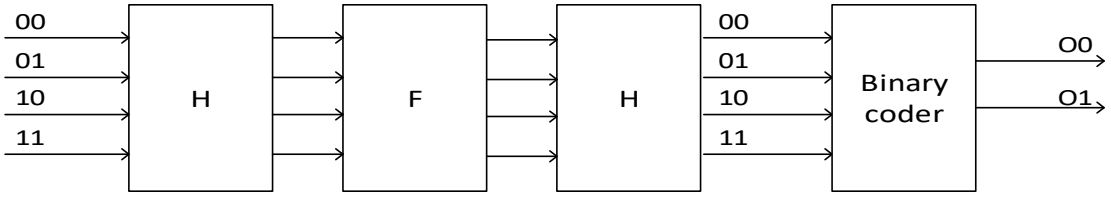

Fig. 7. Diagram of quantum gates implementation of search problem

The explanation of the presence of 4 inputs (instead of 2 inputs) is given by the fact that here we have two inputs: “x” and “y” that are represented by their possible binary combinations: 00, 01, 10, 11. As we have shown in the previous section, for each combination there is a signal. At the input, the presence of 1 on the corresponding input of state "11" while on the other 3 inputs we have 0, which signifies that we have an input that will always be in the "11" state.

As an output we will have: state "11" if the function is even and state "01" if the function is odd.

In other words, the output state "y" is always 1 while the state of output "x" is 1 if the function is even and 0 if it is odd.

The table below represent the resources involved for system implementation and the response time - ActiveHDL was used for its implementation and the Xilinx tools for the synthesis and deployment. The target circuit is a Spartan 3 XC32400. Practically, the classic solution is a sequential one, the response is in three clock signals with the given period.

Table 1. Results of implementation with quantum optimized circuit (implementation on Xilinx Spartan 3 XC3S400).

	Classic	**Quantic**
Slices (logic blocs)	3	2
Flip flops	5	0
LUTs (detailed logic)	4	4
Timing	Sync 3 clk. x 3.92 ns	Delay 9.07 ns

5. CONCLUSIONS

In this paper we have not only targeted to create another emulator for the quantum computational circuits, but also to present a possible implementation solution of the quantum computing structures on classical digital circuits. Basically, we presented a parallel computing solution redesigned to according to the quantum computing structures requirements.

From the experimental results we have presented, our solution - with few optimizations - outperforms as a classic solution in solving a problem of parity determination of a binary function. As a result, our quantum structure implementation solution is more resource efficient (the test was done using the Xilinx synthesis and implementation tools for the Spartan 3 XC3S400 circuit) and as well in terms of response time.

There are several directions to continue research: the first is to improve our model in order to support quantum algorithms in several steps and to implement more complex quantum structures to allow for more sophisticated search operations.

ACKNOWLEDGEMENT

The research that led to the results shown here has received funding from the project "Cost-Efficient Data Collection for Smart Grid and Revenue Assurance (CERA-SG)", ID: 77594, 2016-19, ERA-Net Smart Grids Plus.

REFERENCES

[1] A. Ekert, P. M. Hayden, H. Inamori, Basic Concepts in Quantum Computation, "Coherent atomic matter waves", *Les Houches - Ecole d'Ete de Physique Theorique book series* (LHSUMMER, volume 72), pp 661-701, 2002

[2] A.Steane, "Error Correcting Codes in Quantum Theory", *Physical Rev. Lett*, vol 77, pp.793, 1996

[3] P.W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring*", 35th Annual Symposium on Foundations of Computer Science*, 1994

[4] Samuel J. Lomonaco Jr, "Grover's quantum search algorithm", *Proceedings of Symposia in Applied Mathematics* Volume 58, 2002

[6] Michael A. Nielsen, Isaac L. Chuang, "Quantum Computation and Quantum Information",*Cambridge University Press*, 2010

[7] Jeremy L. O'Brien, "Optical Quantum Computing", *Science*, Vol. 318, Issue 5856, pp. 1567-1570, 2007

[8] S. A. Lyon, "Spin-based quantum computing using electrons on liquid helium", *Phys. Rev.* A 74, 052338, 2006

[9] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe & J. L. O'Brien," Quantum computers", *Nature* volume 464, pages 45–53, 2010

[10] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik & A. G. White, "Towards quantum chemistry on a quantum computer", *Nature Chemistry* volume 2, pages 106–111, 2010

[11] Noson S. Yanofsky and Mirco A. Mannucci, "Quantum Computing for Computer Scientists", *Cambridge University Press*, 2008

[12] M Aminian, M Saeedi, MS Zamani, "FPGA-based circuit model emulation of quantum algorithms", *Symposium on VLSI, 2008. ISVLSI '0*8. IEEE Computer Society Annual, 2008

[13] T Häner, DS Steiger, M Smelyanskiy., Matthias Troyer, "High performance emulation of quantum circuits", *Proceedings of the* International *Conference for High Performance Computing, Networking, Storage and Analysis*, 2016