New FPGA design solution using quantum computation concepts

Alin Gheorghiță Mazăre, Laurențiu Mihai Ionescu, Ioan Liță, Gheorghe Șerban
Department of Electronics, Computers and Electrical Engineering, University of Pitesti
Pitesti, Romania
laurentiu.ionescu@upit.ro , l_ionesco@yahoo.com, ioan.lita@upit.ro

Nadia Belu
Department of Manufacturing and Industrial Management
University of Pitesti
Pitesti, Romania

2021 IEEE 27th International Symposium for Design and Technology in Electronic Packaging (SIITME) | 978-1-6654-2110-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/SIITME53254.2021.9663653

***Abstract*—Quantum computer emulators implemented on FPGA are generally designed to illustrate possible applications in which these concepts can be used - as a method of functional testing of possible applications involving quantum computing. In this paper we try to present another possible use of a quantum computation emulator, proposed by the authors, for the synthesis of circuits in FPGA with superior performance to conventional or parallel design methods. The paper presents a synthesis solution of parallel structures based on concepts from quantum computation emulated on FPGA. As will be seen in the paper, using the solution improves the computation time compared to the conventional method and reduces the number of resources allocated compared to the parallel solution. So, the solution which we propose is somewhere between conventional sequential logic circuit design methods - it has a shorter response time than these and parallel circuit design methods - it provides a smaller number of resources than these. In order to show the advantages of the proposed solution, we presented the implementation of a circuit for determining the parity of a binary function. The solution was implemented using the conventional solution, the parallel solution and the proposed solution using quantum computing concepts. It will show the performance of the proposed solution compared to the other design models.**

Keywords*— *Quantum computing, Quantum computer emulator, Field Programmable Gates Array (FPGA), Circuits design

I. INTRODUCTION

In the context of current technological developments, FPGA circuits have become available for the wide consumer market and therefore there are more and more applications that have migrated from conventional technologies (microprocessors, microcontrollers) to FPGA. However, there are some restrictions such as global buses or clock signals. Their number is small compared to the number of cells in the FPGA [1].

Quantum computation is a relatively new concept starting from the behavior of microparticles modeled by quantum mechanics [2]. The basic idea is that the processing is done for several states at the same time, so we have a parallelism of states. This parallelism is possible due to the superposition of the microparticle states: for example, the polarization of photons can be both horizontal and vertical, the spin of an electron can be both positive and negative, etc. At least theoretically, this leads to very fast execution of tasks. Algorithms could be used mainly as search solutions in an unordered data set [3] searches that by conventional methods take a long time. For example, while a search by conventional methods in a set of N data takes on average N / 2 steps, the search using the Grover quantum computation algorithm is approximately SQRT(N) steps. The implementation of quantum calculus is done on dedicated structures based on photons [4] or electrons [5]. The field is very dynamic and with a huge potential to be the future direction of computer development. In general, emulators are software, which will run on ordinary computers possibly using different media: for example, an emulator for Mathematica was developed and presented in the paper [6]. In order to get closer to the parallelism of states, emulators are implemented on hardware structures: their operational parallelism can better emulate the parallelism of states in a quantum system. The solution presented in the paper [7] describes a quantum emulator with the calculation module (quantum transformations), the state coding module (quantum state), the quantum measurement and the communication module implemented in the FPGA. A comprehensive study of modern emulators implemented in digital structures was also done in the paper [8]. Here are several approaches using arithmetic circuits and a kernel of operations constructed as scalable structures. A library that allows the generation of Verilog files for the implementation of a parameterizable and portable quantum emulator interface on several CMOS logic circuit architectures is presented in the paper [9]. The solution presents a digital analog approach of the emulator by using digital calculation modules and analog signals.

Originally developed as a method of emulating in FPGA the structures from a quantum computer, our solution has also demonstrated its efficiency in allocating cellular structures from FPGA.

The paper is structured as follows: the next section is a brief introduction to the concepts of quantum computing and a presentation of the quantum search algorithm (Grover algorithm). Section 3 presents the emulator model implemented and the experimental results section show how

it was tested and what the performances are - in terms of search time and in terms of resources occupied in the FPGA.

II. Quantum Computation Concept

Quantum computing is based on the use of the idea of qubit (quantum bit). A qubit can have state 0, state 1 (just like a bit) but can also have a superposition between the two states. Each state is represented by a complex number, so that the qubit is represented as a pair of two complex numbers:

Q = (z,o), where 'z' represents the complex number associated with state 0 and 'o' represents the complex number associated with state 1.

The modulus of the complex number squared represents the probability that the respective state will be found in a measurement. So:

$|z|^2 = P_0$, the probability that in a measurement the system will be in state 0

$|o|^2 = P_1$, the probability that in a measurement the system will be in state 1.

Obviously, the sum of the squares of the modules of complex numbers must be 1. The complex number phase represents the state phase. Its significance is less obvious, it does not influence the result when a measurement is made but it has an essential contribution when performing operations in the superposition state.

Quantum registers are used in quantum computing algorithms: several qubits. In this case, complex numbers are associated with each state. For example, a system composed of three qubits will be represented by 8 complex numbers associated with each state resulting from the binary combination of the three qubits. A quantum computing algorithm involves bringing the system into the superposition state, maintaining it in this state while the algorithm is running and then, finally, performing a measurement that leads to the expected result. The most important element is finding the algorithm that leads to the evolution of the system to the desired result. For example, the quantum search algorithm (Grover algorithm) involves the application of quantum transformations: the Hadamard transform, the F(0) transform - the PI phase shift for state 0 and the F(?) transform - the PI phase shift for a given state '?' which is to be identified. All these transformations are applied on a set of N data. After applying about SQRT (N) steps (more precisely PI / 4 SQRT (N) steps) the system converts to the searched state - more precisely with a probability of 99.99% to a measurement will find the location that has the searched state. Each step of the search leads to an increase in the probability that the solution will be found - so the system gradually evolves from a state of superposition of all states in which all possible solutions coexist with equal probability to a state in which only the solution sought is measured.

The basic operators in quantum computing are called quantum gates. 2 main types of gates are identified: those that determine transformations of the module of complex numbers that represent the states and those that determine transformations at the level of the phases of the complex numbers that represent the states. There is also a basic operator, called the Hadamard transform, which also influences the modulus and phase of the states (introduces phase shifts with PI at some states). The interconnection of several quantum gates forms a quantum circuit. Quantum calculation algorithms can be implemented on these circuits.

III. Quantum Circuit FPGA Emulator Model

The proposed model uses a register to store states and phases for one qubit or more qubits (quantum register with several qubits). A scheme of a qubit, as it is implemented in our model is presented in the figure below.

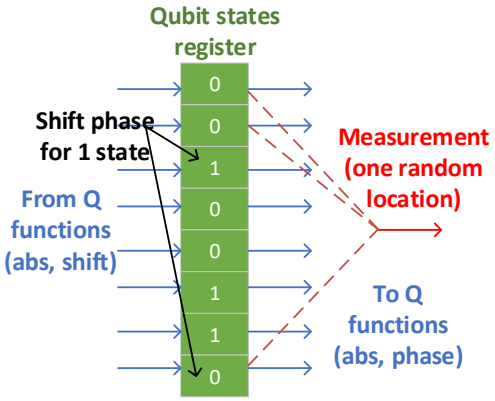

Fig. 1. Modeling a qubit

The qubit is a serial register. All operations that take place with it take place with the entire data set in qubit. The measurement means the serial reading of a bit - depending on the number of bits of 0 or 1 that exist in the register (randomly distributed) it is performed with a probability of reading state 0 or 1. After reading, the extracted value can be considered the state of qubit . The phase of each state is encoded in a memory addressable by the contents of each location of the register. When implementing the search algorithm we have only two phase values (0 and PI) reason why we did not use the phase memory but we coded the two states by the position of the bit in the register. Thus, as can be seen in the figure, the High zone in the register is assigned to state 0 and the low zone to state 1. A state 0 that is not in its zone is considered out of phase with PI.When implementing the algorithm, a scheme of the type in figure 2 will be used.

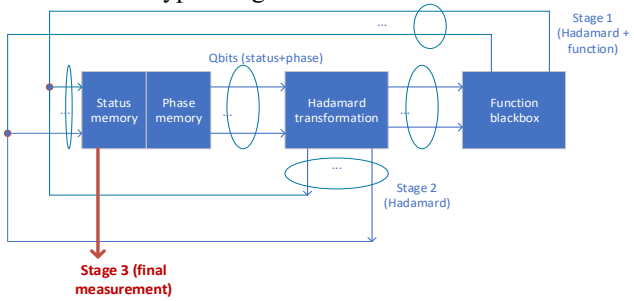

Fig. 2. Quantum circuit for search

As components we have a module that performs the Hadamard transform and one that performs the transforms F (0) and F (?). The Hadamard transform involves reversing the state of some bits in the register and placing them in the area with PI phase shift. Transforms F (0) and F (?) assume the

displacement in the PI zone for state 0 respectively for the unknown state to be identified.

The search problem is as follows: a set of N data (in our case the tests were done for 8 or 16 data) contain only one date with the value 1 and the rest with 0. We want to determine the location where the date is 1.

To create the search framework we built a content addressable memory (CAM). Memory allows you to write one location at a time (or multiple locations at the same time). It also allows searching for a date: the date is entered on an entry and at each location there are comparators that identify whether the date was found in that location or not. So, from each location we will have a match type output that shows whether the date was found (1) or not (0) on that date.

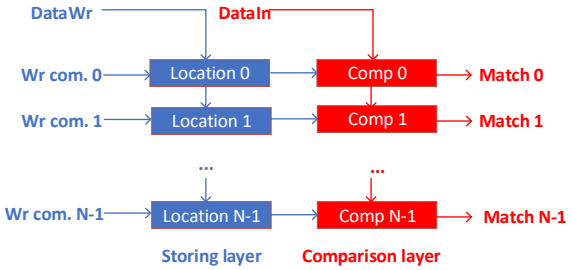

Fig. 3. Content addressable memory (CAM) used in experiments

The problem now arises in finding which location has match 1 - so at which location is the date sought. In the next section we will present the techniques used to solve this problem: one by sequential search, another by parallel implementation and finally the last one by using the proposed quantum model.

IV. Experimental Results

The figure below shows how to test the system.

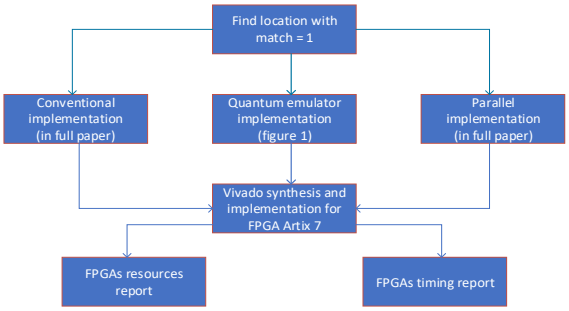

Fig. 4. The design flow taken to obtain the results, after all three implementations

As you can see in the figure 4, we have the 3 implementation methods. It starts from the memory accessible through the content. As a problem to be solved is the determination of the location that generated match = 1, so that found the date placed on the date entry of the memory accessible through the content.

a. Sequential query solution

Match entries from all CAM locations are taken over by a MUX (sized for this purpose). The MUX selections are controlled by a counter as can be seen in the figure 5.

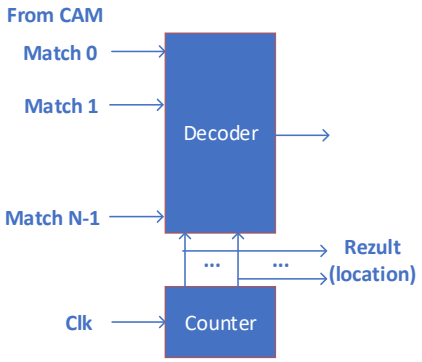

Fig. 5. Sequential search algorithm implementation

When we have a match on one of the MUX inputs the counter will be stopped and will indicate the number of the input that generated match = 1.

b. Parallel solution

On each match is placed a module that generates the location number and match status. The module has a vertical input that comes from the locations above. If the location has match = 1 then the match value next to the location number propagates down.

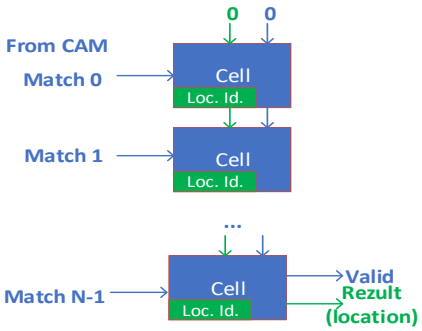

Fig. 6. Parallel search implementation

c. Quantum search solution

The solution uses an implementation of the Grover search algorithm. The Hadamard, F (0) and F (?) transforms practically determine a "diffusion" of the solution with match = 1 through the network of selection circuits as can be seen in the diagram in the figure.

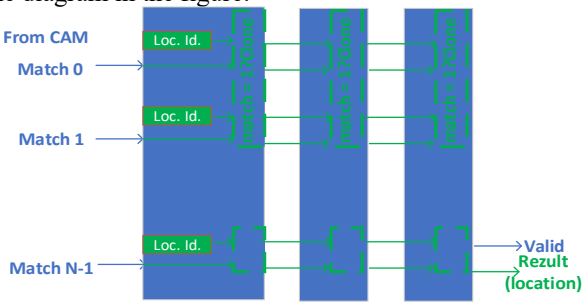

Fig. 7. Quantum search implementation

d. Comparative analyzing

All three solutions were implemented on a Basys 3 board from Digilent with FPGA Artix 7 (figure 8).

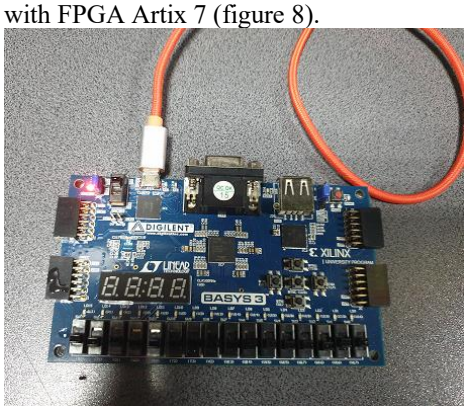

Fig. 8. Picture with Basys 3 board used in experiments – communication between PC and board is performed using same USB cable for configuration and power

Two variants were implemented: N = 8 and N = 16 for all three solutions. The resources consumed in the FPGA for N = 8 and for N = 16 are presented in the table below. Some equivalence can be observed at N = 8 in terms of resources but a more significant variance at N = 16. As can be seen, the sequential solution seems more complex for N = 8 but is kept as dimensions for N = 16 On the other hand, in the parallel solution, an increase in consumed resources can be observed. The quantum solution registers a slight increase but is kept below the parallel solution at N = 16.

TABLE I. RESOURCES FOR IMPLEMENTATION

Algorithm	**Resources**
N = 8	
Serial	Number of Slices: 41 out of 3584 Number of Slice Flip Flops: 67 out of 7168 Number of 4 input LUTs: 47 out of 7168
Parallel	Number of Slices: 38 out of 3584 Number of Slice Flip Flops: 64 out of 7168 Number of 4 input LUTs: 52 out of 7168
Quantum	Number of Slices: 40 out of 3584 Number of Slice Flip Flops: 64 out of 7168 Number of 4 input LUTs: 48 out of 7168
N = 16	
Serial	Number of Slices: 80 out of 3584 Number of Slice Flip Flops: 132 out of 7168 Number of 4 input LUTs: 92 out of 7168
Parallel	Number of Slices: 78 out of 3584 Number of Slice Flip Flops: 128 out of 7168 Number of 4 input LUTs: 107 out of 7168
Quantum	Number of Slices: 74 out of 3584 Number of Slice Flip Flops: 128 out of 7168 Number of 4 input LUTs: 97 out of 7168

The estimated time in the synthesis phase of the circuits is presented below for N = 8 and N = 16. Here too we can observe a variation of time with increasing size at the serial solution but keeping it constant at the parallel solution. Interesting is the evolution of time to the quantum solution.

TABLE II. RESPONSE TIMING

Algorithm	**Timing**
N = 8	
Serial	13.035ns Estimated for N/2 steps number: 4 * 13.035ns = 52.14ns
Parallel	14.648ns
Quantum	14.644ns
N = 16	
Serial	13.531ns Estimated for N/2 steps number: 8 * 13.531ns = 108.248ns
Parallel	24.226ns
Quantum	19.501ns

V. CONCLUSIONS

From the presented results, it can be seen that the quantum search solution obtains an optimum in terms of resources consumed and search time. From our point of view, it could be a design solution that can have significant results for other algorithms as well. In terms of content-addressable memories, we seem to have interesting results in terms of allocated resources and response time. A future direction of research will be to investigate for higher values of N the effectiveness of our solution.

We can experiment with other types of quantum algorithms to see the results obtained after implementation with the solution proposed here. The solution can be seen as a future approach in FPGA circuit design - another possible direction of research in the field of FPGA application development.

ACKNOWLEDGMENT

The research that led to the results shown here has received funding from the project "Increasing the institutional capacity of bioeconomic research for the innovative exploitation of the indigenous vegetal resources in order to obtain horticultural products with high added value (BIOHORTINOV)", ID: PN-III-P1-1.2-PCCDI-2017-0332/Pc1,Pc2, UEFISCDI.

REFERENCES

[1] Zhengjie Li, Yufan Zhang, Jian Wang and Jinmei Lai, "A survey of FPGA design for AI era", Journal of Semiconductors, 2020

[2] 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

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

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

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

[6] T Jones, SC Benjamin, QuESTlink–Mathematica embiggened by a hardware-optimised quantum emulator, Quantum Sci. Technol. 5 (2020) 034012

[7] Jakub Pilch, Jacek Długopolski, An FPGA-based real quantum computer emulator, Journal of Computational Electronics volume 18, pages329–342(2019)

[8] Naveed Mahmud Esam El-Araby David Caliga, Scaling reconfigurable emulation of quantum algorithms at high precision and high throughput, Quantum Engineering. 2019

[9] Jeroen van Dijk ; Andrei Vladimirescu ; Masoud Babaie Edoardo Charbon Fabio Sebastiano, SPINE (SPIN Emulator) - A Quantum-Electronics Interface Simulator, Proceedings - 2019 8th International Workshop on Advances in Sensors and Interfaces, IWASI 2019