Implementation of Small Evolvable Combinational Logic Circuit by FPGA

Petr Burian
University of West Bohemia in Pilsen, Department of Applied Electronics and Telecommunications
Univerzitni 26, 30614 Pilsen, Czech Republic, E-mail: burianp@kae.zcu.cz

***Abstract* - The main subject of this paper is the design of an evolvable combinational circuit. It deals with an interconnection of the genetic algorithm and the reconfigurable digital circuit. This paper examines the dependence of particular aspects of the genetic algorithm for the evolvable hardware domain usage. The practical implementation of this algorithm by the FPGA circuit is researched as well.**

I. INTRODUCTION

The usage of the evolutional computational techniques is a new and modern way for the design of a digital circuit. New and innovational solution of the circuit can be found by these techniques. The evolutional techniques and the suitable reconfigurable structure are the fundamental elements for an evolvable hardware. They are the circuits, that are able to be self development. They can change their function dynamically in time.

The basic representative of the computational techniques is the genetic algorithm. This algorithm is inspired with the Darwin's theory of the evolution of species. The genetic algorithm is able to solve the optimization problems very effectively. Just this feature can be exploited for the evolvable circuits. The finding of the suitable configuration information for the reconfigurable structure is the optimization problem in the evolvable hardware domain.

The reconfigurable structures are most often constructed from the functional cells and the multiplexers. The function of these elements depends on its configuration information. The function cells can work in various levels. The cells can work on a gate level, they implement the basic logic functions, or on a higher level, then they implement higher functions (an adder, a multiplier, max, min). The lower level makes it possible to find a more innovational and unconventional solution.

The evolutional algorithm and reconfigurable structure should be implemented in one circuit. For this domain, the FPGA circuits are very suitable. Indeed, we could use a processor for the implementation of the evolutional algorithm. However, the FPGA circuits provide better possibilities for the optimization of the speed.

II. RECONFIGURABLE STRUCTURE

I used the reconfigurable structure from the fig.1 [1]. It is a reconfingurable combinational logic circuit. It disposes of 4 inputs and 2 outputs. The circuit is controlled by 32-bits configuration bitstream. And it can implement up to 178764 unique logic functions. Every function cell has 2 inputs and 1 output and implements 8 logic functions: NOR, x AND not y, not x AND y, AND, OR, not x OR y, x OR not y, NAND.

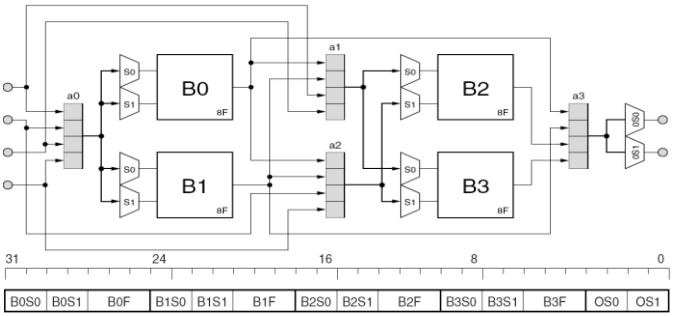

Fig. 1. The reconfigurable structure and its configuration bistream [1]

The groups of the multiplexers control the connection between the cells, between the inputs and the cells, and between the cells and the outputs.

III. GENETIC ALGORITHM

The genetic algorithm has many parameters and modifications. I decided that the standard modification (the SGA – The Standard Genetic Algorithm) of this algorithm will be used.

The individuals of the algorithm are represented by mean of the binary 32-bits stream. It correspondes with the configuration stream of the reconfigurable structure. The one-point crossovering is used. It means that two parents (the individuals) enter into the crossovering and two offsprings are the output. The selection of the parents depends on a probability of the crossovering (the Pc parameter). Two types of the mutation are used. The first way depends on a probability of the mutation (the Pm parameter). This way of the mutation can change several bits in the individual. The second way always changes only the one bit (single-bit mutation) of the individual. And this type of the mutation doesn´t depend on the Pm parameter. The key element of every evolutional algorithm is a fitness (valuation) function. This function tells us, how quality solution the individual produces. I tested the individuals with all input combinations. The fitness expresses the number of the lines in the truth table, where the output matches required output. I used the reconfigurable structure with 4 inputs and 2 outputs. For that reason, the maximum of fitness is 32 (16 x 2). The tournament type of the selection is used in the SGA. New individuals are selected from the offsprings, the mutants and from the individuals of the former generation. The selection is complemented by an elitism. It means that the best individual (the leader) gets into the next generation.

IV. HARDWARE IMPLEMENTATION

Both the reconfigurable structure (the circuit) and the standard genetic algorithm are implemented in the FPGA circuit. The circuit is created by means of the VHDL language. I used the FPGA Altera Cyclone II circuit.

The whole evolvable combinational logic circuit is composed of the Avalon periphery for a softcore NIOS II processor. This conception is very profitable for the debuging and the testing. We can very easily observe and control the process of the algorithm by the NIOS II processor and its J-TAG console. However, the algorithm itself runs very quickly in the created logic.

The whole system is devided into several parts (fig. 2). The individuals and their fitness are stored in the embedded memory of the FPGA circuit. This memory is very fast. I used the SGA with 64 individuals. Every individual needs 32 bits for its representation and 8 bits for its fitness. However, the whole algorithm needs 4 times more memory. We have to store also the former generation, the offsprings and the mutants. On the whole, we need about 1280 bytes.

Several modules perform the main operations of the algorithm. They are called: crossovering, mutation, selection, fitness and elitism. These modules are controlled by means of the control register. The modules are connected with the memory by the multiplexer. This multiplexer is controlled by the control register, too. Therefore, this register controls the whole process of the evolution. We can read and change its value by the NIOS II processor via the Avalon bus. The processor can read the value of the leader (the best individual) and its fitness, too. The Avalon slave port is used for communication between the evolvable circuit and the NIOS II processor. The selected way of the implementation enables the very easy usage of this periphery by other systems.

The system also includes the generator of pseudo random numbers. The generator is created by the 32-bits linear feedback shift register with the polynomial $x^{32} + x^7 + x^6 + x^2 + x^0$. This generator is designed so that a random number is generated in one cycle [3].

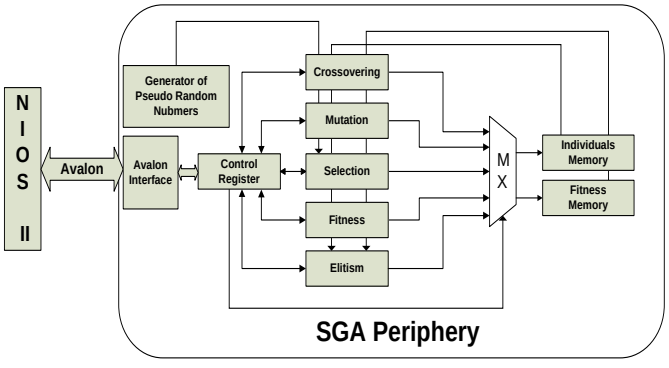

Fig. 2. The diagram of the standart generic algorithm pheriphery

V. RESULTS AND CONSLUSION

The functional prototype of the evolvable combinational logic circuit was implemented. This combinational circuit can change its truth table dynamically in time. That is a very small circuit for now. It has only 4 inputs and 2 outputs. However, larger circuits can be designed by means of the similar ways. The system needs to know only required truth table. The suitable configurational information for the reconfigurable structure is found so that the circuit works correctly.

The speed of the evolution is a very important indicator in the evolvable hardware domain. I tested the system for several test truth tables. The necessary time for the successful evolution doesn´t exceed 150 ms. Nevertheless, for some truth tables, the evolution didn´t find the solution. It is a great negative of this way of the design. We have no confidence that the solution will be found. The setting of the evolutional parameters is very significant too. The experiments show, that the usage of the "single-bit" mutation and a low value of the probability of the crossovering leads to the best results. For that reason, I could suppose that the usage of the crossovering is not essential for this type of the evolution. The following chart shows the results of three various truth tables (in legend in a serial format). The chart describes the dependence of the average number of the generations that are needed for the successful evolution, on the probability of the crossovering.

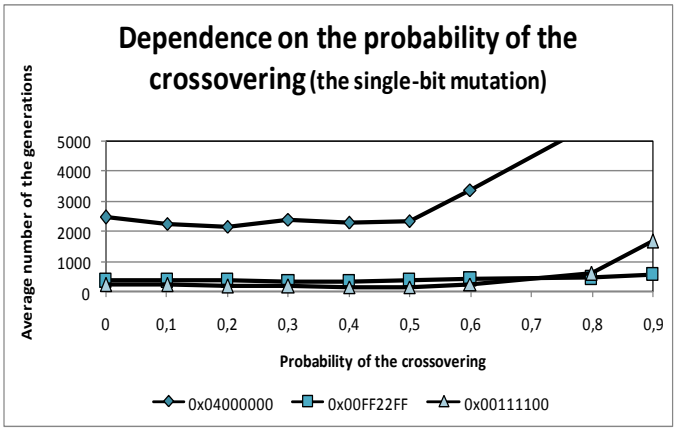

Chart 1. Dependence on the probability of the crossovering

The examination of the evolution detected some deficiences. So I will have to optimize the fitness function to increase the power of the evolution. Now, the power of this implementation is up to 40000 genenerations per second.

REFERENCES

[1] Sekanina Lukas, Mikusek Petr: Analysis of Reconfigurable Logic Blocks for Evolvable Digital Architectures, In: Lecture Notes in Computer Science, 2008, no. 4974, DE, p. 144-153, ISSN 0302-9743.

[2] Sekanina L.: Evolvable components. Springer, Natural Computing series, 2004. ISBN 3540403779.

[3] Martin, P. : A Hardware Implementation of a Genetic Programming System Using FPGAs and Handel-C, In: Genetic Programming and Evolvable Machines, 2008, p. 317 – 343**,** ISSN 1389-2576.