EVOLVABLE HARDWARE IN XILINX XCR3064 CPLD

Rustem Popa

"Dunarea de Jos" University of Galati,
Department of Electronics and Telecommunications,
Domneasca Street – 111, Galati, Romania
E-mail: Rustem.Popa@ugal.ro

Abstract: Evolvable Hardware (EHW) is a hardware which modifies its own structure in order to adapt to the environment in which it is embedded. This reconfigurable hardware is implemented on a programmable circuit, whose architecture can be altered by downloading a binary bit string. These bits are adaptively acquired by evolutionary algorithms. In this paper we have used an evolutionary algorithm to design some combinational and sequential logic circuits. These designs have been implemented in a real Xilinx XCR3064 CPLD and have been compared with other designs created by manual methods or other heuristic techniques. A better fitting of circuit resources have been observed in almost all evolutionary designs.

Keywords: Boolean functions, Genetic algorithms, Programmable integrated circuits, Sequential machines, State assignment.

1. INTRODUCTION

EHW is a hardware built on a software reconfigurable logic device, such as Programmable Logic Device (PLD) or Field Programmable Gate Array (FPGA). In these circuits the logic design is compiled into a binary bit string. By changing the bits, arbitrary hardware structures can be implemented instantly. The key idea is to regard such a bit string as a chromosome of a genetical algorithm. Through genetic learning, EHW finds the best bit string and reconfigures itself according to rewards received from the environment. In this way, the hardware structure is adaptively searched by genetic algorithm (GA). This basic idea of EHW is illustrated in the figure 1.

The traditional design process is top-down and begins with a precise specification. EHW is applicable even when no hardware specification is known before. Its implementation is determined through a genetic learning in a bottom-up way. GA is meant to mimic Darwinian evolution. A population of candidates is maintained, and goes through a series of generations. For each new generation, some of the existing candidates survive, while others are created by a type of reproduction and mutation from a set of parents. EHW combine knowledge of both GA and logic design to evolve circuits.

Research in EHW can be divided into *intrinsic evolution*, which refers to an evolutionary process in which each circuit is built in electronic hardware and tested, and *extrinsic evolution*, that uses a model of the hardware and evaluates it by simulation in software.

In this paper we have shown that evolutionary design is favourably against the traditional design in Complex PLD (CPLD) implementation. We have used only extrinsic evolution, but the circuits generated in this way have been tested in a real Xilinx XCR3064 CoolRunner CPLD by using the Xilinx ISE 6.1i software. The remaining sections of the paper are organised as follows: Section 2 describes in more detail the genetic learning component of the EHW. Section 3 shows some examples of combinational and sequential circuits and their implementation by using of traditional and then evolutionary design techniques. All these

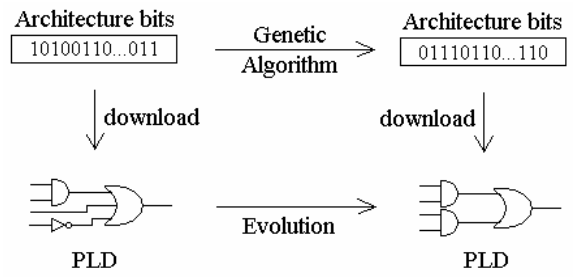

Fig. 1. The basic idea of EHW. A binary bit string, called "architecture bits", is modified by evolution. Each new string of "architecture bits" implements a new electronic circuit in PLD.

circuits have been implemented in a 64 macrocell Xilinx CPLD. These experimental results are given in Section 4. Finally, Section 5 provides the conclusions and future work.

2. GENETIC LEARNING IN EHW

The genotype of an evolved structure on PLD basis is given by the bits for fuse array and bits for logic cells. However, this genotype representation has inherent limitations, since the fuse array bits are fully included in the genotype, even in the case that only a few bits are effective. Iba, *et al.* (1996), has introduced the variable length chromosome, a short chromosome that can increase the maximum size of the evolved circuit. A tiny different method for chromosome representation, by using only the essential number of minterms, was proposed by Popa and Iliev (1999). In this way, this method can reduce the chromosome length and establish an efficient adaptive search.

All the developed algorithms in this paper are based on the fundamental structure of a GA. The initial population of chromosomes (bit strings) is generated randomly. All these potential solutions are evaluated using a fitness function. In our case, for a single boolean function, fitness is the ratio between the number of the correct values of the function and the number of all possible values (which is 2^n, if the boolean function has n input variables). A well-designed circuit will be obtained only when the value of fitness is 100%. An approximately value of the fitness is unacceptable here.

The next step is selection and reproduction. For each individual, a number of copies are made, proportional to its fitness, while keeping the population size constant. The least fit individuals are deleted. This is the survival of the fittest part of the GA.

The next step is crossover, where individuals are chosen two at a time, as parents. They are converted into two new individuals, called offsprings, by exchanging parts of their structure. Thus, each offspring inherits a combination of features from both parents. We have obtained the best results with one point crossover, with a probability of 80%. This operator may be used more times on different selected pairs of chromosomes in a generation.

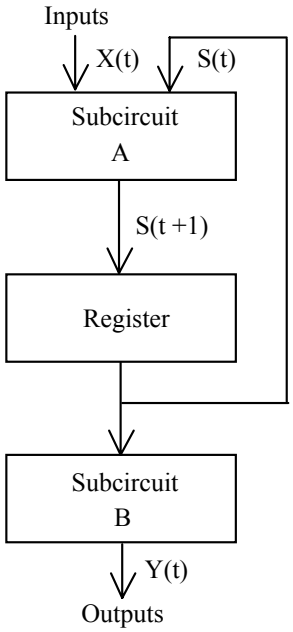

Fig. 2. A finite state machine may be divided into purely combinational blocks (subcircuits A and B) and register.

The next step is mutation. A small change is made to each resultant offspring, with a small probability. After mutation is performed on an individual, it no longer has just the combination of features inherited from its two parents, but also incorporates the additional change caused by mutation. This ensures that the algorithm can explore new features that may not yet be in the population. It makes the entire search space reachable despite the finite population size. The whole process is repeated for several generations, and, if the best chromosome in population will have the fitness of 100%, then this bit string represents a good solution for our function.

3. SOME EVOLUTIONARY DESIGNS

The first successful evolved circuits have been the digital combinational logic circuits. The evolution of sequential logic circuits is considerably less mature. The complexity of circuit connections and encoding chromosomes to evolve the sequential logic circuit may be one of the reasons that not much work has been done in this area (Ali, *et al*., 2004).

As we can see from the figure 2, the sequential circuits may be divided into purely combinational blocks and registers. Large sequential circuits are typically modelled by smaller interacting finite state machines (FSMs). A FSM is defined as a mathematical model of a system with discrete inputs, discrete outputs and a finite number of internal configurations or states. The states of a system completely summarise the information concerning past inputs to the system that is needed to determine its behaviour on subsequent inputs.

In the first subsection we have shown the design of a boolean function of 3 variables. In the next two, we have used different ways to design the combinational blocks of the two distinct FSMs.

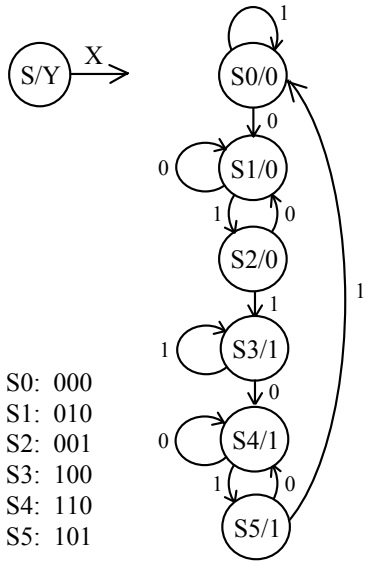

Fig. 3. A sequence detector described as state transition graph and GA state assignment.

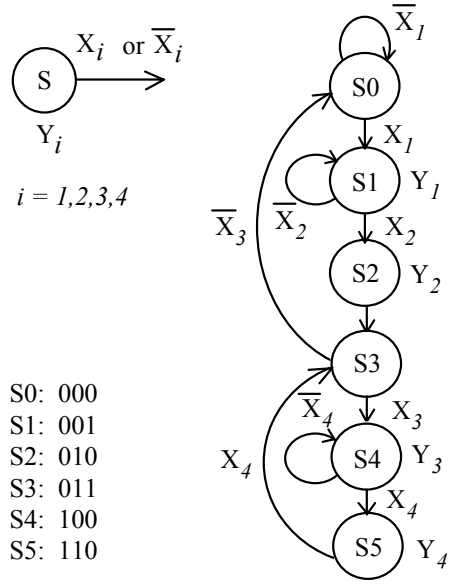

Fig. 4. A computer interface described as state transition graph and manual state assignment.

3.1 The Implementation of a Boolean Function

We have considered a boolean function represented in a minimal disjunctive form by using a Karnaugh map:

$$f = \bar{x}_1 \cdot x_2 \cdot x_3 + x_1 \cdot \bar{x}_3 + \bar{x}_2 \cdot \bar{x}_3 \qquad (1)$$

This representation has a cost of 7 gates and 13 inputs, including inverters. By applying some switching-algebra theorems our function may be written in the next form:

$$f = x_3 \oplus \overline{\bar{x}_1 \cdot x_2} \qquad (2)$$

Now, the cost of implementation is only of 3 gates and 5 inputs. Unfortunately, there is no algorithm to find this convenient form of the function, only the heuristics and experience of the human designer.

We have tried to find another representation of this function by evolutionary design. We have used the idea given in Coello, *et al.* (2000). Each combinational circuit is represented as a rectangular array of logic gates. Each of these gates has two inputs and one output, and the logic operator may be selected from a list. At the beginning of the search, all the gates from the matrix are disposable to implement a functional circuit. Once a functional solution appears, then the fitness function is modified such that any valid designs produced are rewarded for each gate which is replaced by a simple wire. The algorithm tries to find the circuit with the maximum number of gates replaced by wires that performs the function required.

The chromosome defines the connection in the network between the primary inputs and primary outputs. We have used a network of 4 gates, a population of 32 chromosomes, 10 of them beeing changed each generation, a single point 100% crossover and 5% rate mutation.

A feasible solution has been obtained in less than 100 generations. This function may be written as:

$$f = \overline{\overline{\overline{x_1 \oplus x_2} + x_1 \oplus x_3}} \qquad (3)$$

We can see that, in this case, the cost is of 3 inverting gates and 6 inputs, and this solution has the minimum delay time between any input and the output of the circuit, in a gate level implementation.

Finally, the most extended representation of this function is the disjunctive canonical form, with a total cost of 9 gates and 23 inputs. We have implemented all these four different equations of the function in a real programmable circuit and the results are compared in Section 4.

3.2 The Implementation of a Sequence Detector

The FSM represented in the figure 3 is a sequence detector with one-input, one-output and 6-internal states. When the input sequence 011 occurs, the output becomes 1 and remains on this logic value until sequence 011 occur again. In this case, the output returns to 0, and maintain this value, until a new sequence 011 appears. This circuit has been described in Ali, *et al.* (2004).

Firstly a GA has been used to find optimal state assignment. An example of state assignment generated in this way is shown in the figure 3. The chromosome represents the FSM as a list of states. The goal of the GA is to extract the optimum state assignment, which requires the least number of logic gates. A more detailed description of this problem is presented in Ali, *et al.* (2004).

Then, the extrinsic EHW has been used to find the functional design of combinational parts of the sequence detector. The two subcircuits, A and B, are evolved by using the method presented in the subsection 3.1 and in Coello, *et al.* (2000).

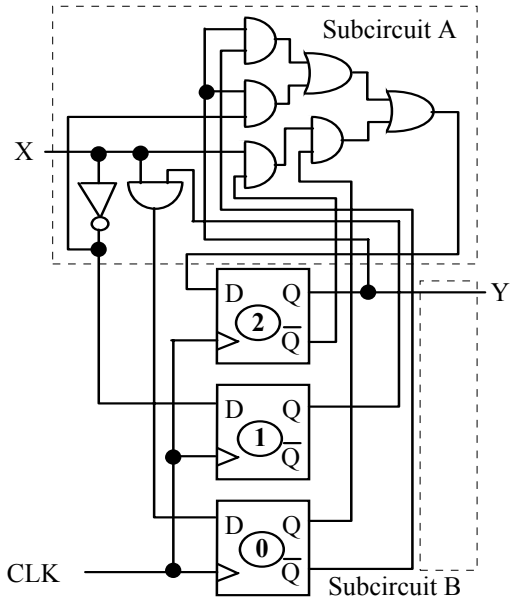

Fig. 5. Evolved optimal circuit solution of the sequence detector (first solution).

The equations of the evolved optimal combinational circuit, as they have obtained by Ali, *et al.* (2004), are the following (see figure 5):

$$D_2 = Q_2 \cdot \overline{Q}_0 + \overline{x} \cdot Q_2 + x \cdot \overline{Q}_2 \cdot Q_0 \quad (4)$$

$$D_1 = \overline{x} \quad (5)$$

$$D_0 = x \cdot Q_1 \quad (6)$$

$$y = Q_2 \quad (7)$$

We have used different notations for the present and the future states of the flip-flops, and for output function, according to Wakerly (2000).

A second evolved solution has been obtained with another state assignment: S0 – 000, S1 – 001, S2 – 011, S3 – 111, S4 – 110 and S5 – 100. The schematic diagram of the circuit is given in the figure 6, and the equations of the combinational circuit are:

$$D_2 = \overline{x} \cdot Q_2 + x \cdot Q_1 \quad (8)$$

$$D_1 = \overline{x} \cdot Q_2 + x \cdot Q_0 \quad (9)$$

$$D_0 = x \cdot Q_1 + \overline{x} \cdot Q_0 \quad (10)$$

$$y = Q_0 \quad (11)$$

A bad state assignment may conduct to much more complex boolean functions for the subcircuits A and B: if S0 – 000, S1 – 001, S2 – 010, S3 – 011, S4 – 100 and S5 – 101, then the equations are:

$$D_2 = \overline{x} \cdot Q_2 + Q_2 \cdot \overline{Q}_0 + \overline{x} \cdot Q_1 \cdot Q_0 \quad (12)$$

$$D_1 = x \cdot Q_1 + x \cdot \overline{Q}_2 \cdot Q_0 \quad (13)$$

$$D_0 = x \cdot Q_1 + Q_1 \cdot \overline{Q}_0 + x \cdot Q_2 \cdot \overline{Q}_0 + \overline{x} \cdot \overline{Q}_2 \cdot \overline{Q}_1 \quad (14)$$

$$y = Q_2 + Q_1 \cdot Q_0 \quad (15)$$

These latest equations have been obtained by manual design, by using Karnaugh maps. All these three solutions have been implemented in a CPLD and the results are discussed in Section 4.

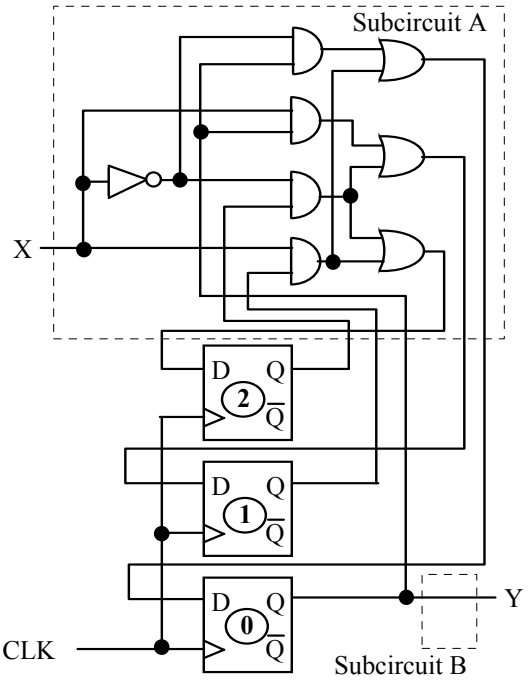

Fig. 6. Evolved optimal circuit solution of the sequence detector (second solution).

3.3 The Implementation of a Computer Interface

The FSM represented in the figure 4 is a computer interface for serial communication between two computers. A transition from one state to another depends from only one of the 4 inputs $x_i, i = \overline{1,4}$. The circuit has 4 outputs, each of them beeing in 1 logic only in a single state. The FSM has 6 states and has been presented in Popa and Iliev (1999).

With the state assignment given in the figure 4, the traditional design of this circuit gives the following equations for the subcircuit A:

$$D_2 = x_3 \cdot Q_1 \cdot Q_0 + Q_2 \cdot \overline{Q}_1 \quad (16)$$

$$D_1 = x_2 \cdot \overline{Q}_1 \cdot Q_0 + x_4 \cdot Q_2 + Q_1 \cdot \overline{Q}_0 \quad (17)$$

$$D_0 = x_1 \cdot \overline{Q}_2 \cdot \overline{Q}_0 + \overline{x}_2 \cdot \overline{Q}_1 \cdot Q_0 + Q_1 \cdot \overline{Q}_0 \quad (18)$$

The subcircuit B, or the output functions, are given by the following equations:

$$y_1 = \overline{Q}_1 \cdot Q_0 \quad (19)$$

$$y_2 = \overline{Q}_2 \cdot Q_1 \cdot \overline{Q}_0 \quad (20)$$

$$y_3 = Q_2 \cdot \overline{Q}_1 \quad (21)$$

$$y_4 = Q_2 \cdot Q_1 \quad (22)$$

For the evolutionary design of this circuit we have preferred a complete different way than in previous subsection. Each of these boolean functions has a maximum number of 5 inputs and a maximum number of 4 minterms. If we want to implement these functions in a PLD structure (an AND array and logic cells configurable as OR gate), then the number of fuse array links is $2 \cdot 5 \cdot 4 = 40$, and we may to consider this number as the total length of the chromosome.

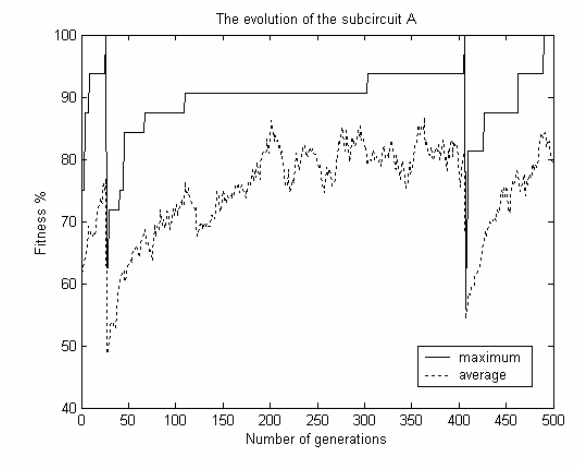

Fig. 7. The evolution of the excitation functions of the computer interface.

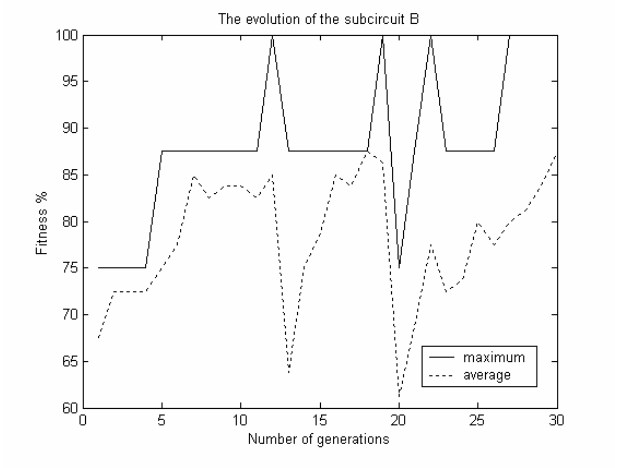

Fig. 8. The evolution of the output functions of the computer interface.

Our GA is a standard one, with the population size of 30 chromosomes. One point crossover is executed with a probability of 80% and the mutation rate is 2%. Six worse chromosomes are replaced each generation. The stop criterion is the number of generations.

Our circuit has 3 excitations functions, $D_i = Q_i^+$, with i = 1, 2, 3, and 4 output functions, y_i, with i = 1, 2, 3, and 4 . Figure 7 reflects the evolution of the circuit for the first 3 functions. However, this circuit is built from 3 independent circuits, each generating one output bit. Therefore, the evolution of a circuit with one output bit is repeated 3 times. The Y axis is the correct answer rate. If it reaches 100%, then the hardware evolution succeeds. All 3 circuits are successfully obtained in less than 500 generations.

In the same way, figure 8 reflects the evolution of the circuit for the output functions. The evolution succeeds after a less number of generations because the total search space is in this case much lower than in previous case (all the output functions have only 3 variables).

Evolution may provide some non-minimal expressions for these boolean functions, but minimization is not necessary for PLD implementations. The length of the chromosomes is greater than the optimal one, and the evolved equations are much more complicated than the given equations from (16) to (22). The complete cost of the whole combinational circuit is consisted of 15 gates and 37 inputs for traditional design, and 30 gates and 102 inputs for evolutionary design. A comparison between these two implementations have been done in Section 4.

4. EXPERIMENTAL RESULTS

All the circuits designed in previous section have been implemented in a real CPLD circuit. This circuit is XCR3064XL, a Xilinx CPLD with 64 macrocells and 1500 usable gates, providing low-power and very high speed, and beeing in-system programmable through JTAG IEEE 1149.1 Interface. Unfortunately, this circuit has only 1000 erase/ programming cycles guaranteed, so it can not be used with intrinsic EHW.

The Xilinx XCR3064XL CPLD is mounted on a development board, called Digilab XCRP, delivered by Digilent, Inc. This low cost platform can be used to implement a wide variety of digital circuits. XCRP board uses a 44-pin PLCC package, with four used for Vcc connections, three for GND, and five for JTAG programming. All remaining 32 I/O pins are routed to the expansion connector, and 31 are also routed to on-board devices (4 for pushbuttons, 8 for slide switches, 8 for LEDs, 10 for the seven-segment display and one for the system clock). The block diagram from the figure 9 shows all connections between the CPLD and the devices on the board.

The XCRP board uses a DB-25 parallel port connector to route JTAG programming signals from a host computer to the CPLD. The programming circuit simply connects the parallel port pins driven by the Xilinx CAD tools directly to the CPLD programming pins. The software we have used is Xilinx Integrated Software Environment (ISE) 6.1i, a complete CAD environment for implementation of complex digital circuits. We have generated the source file of the new project (schematic diagram or VHDL) and have obtained the fitter report and the timing report of the circuit. The bit file may be downloaded in the CPLD by using Xilinx's iMPACT programmer tool from the ISE 6.1i.

We have implemented and analysed all the circuits discussed in Section 3. In the case of boolean function from the subsection 3.1, we have obtained the same results for all different equations done there. The circuit has used a single macrocell from the maximum number of 64 (that is 1/64), only two product terms from the maximum number of 224 (that is 2/224), and only 3 function block inputs from the total number of 160 (that is 3/160). The pad to pad delay is 6 ns, and the total delay of the circuit is not more than this value. We can assume that our software finds an optimal way in connecting the hardware resources of the circuit, even if the function is not done in a minimal form.

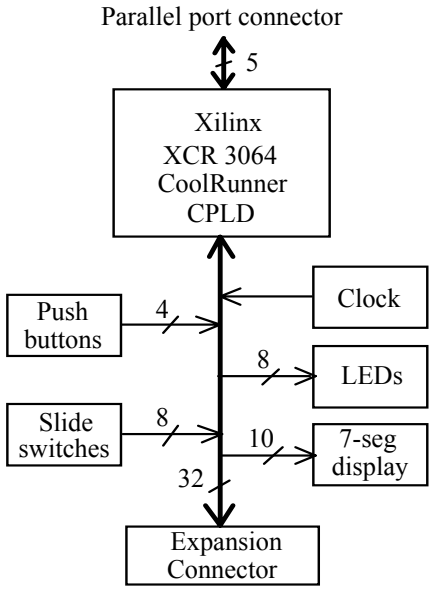

Fig. 9. CPLD connections

The above conclusion is for pure combinational circuits. In sequential circuits, the optimal state assignment is crucial. The sequence detector from the subsection 3.2, implemented with the equations 4,5,6 and 7, has used only 3/64 macrocells, 3/224 product terms, and 3/160 function block inputs. The same circuit, implemented with the equations 8,9,10 and 11, has used 3/64 macrocells, 4/224 product terms, and 4/160 function block inputs. The third circuit, implemented by manual design with the equations 12, 13, 14 and 15, has the worse share of resources: 4/64 macrocells, 9/224 product terms, and 4/160 function block inputs. Even the combinational time delay is different for these circuits (4.7ns, 5.2ns and 7,2ns in that order). All these three circuits have the same number of flip-flops (that is 3/64) and the same number of pins used like inputs/outputs (that is 3/32). It's true that the main differences in the complexity of these three circuits are given by the state assignment. In the best solution, the state assignment has been evolved with a GA.

The computer interface from the subsection 3.3, implemented by manual design with the equations 16 to 22, has used 7/64 macrocells, 11/224 product terms, and 7/160 function block inputs. Evolutionary design, with the same state assignment, provides much more complicated equations. In this case, the complete cost of the whole combinational circuit is consisted of 30 gates and 102 inputs. Surprising is the fact that the implementation of this complex circuit in XCR3064XL CPLD has used only 7/64 macrocells, 10/224 product terms, and 7/160 function block inputs. This is even a better result than in preceding case, because the number of product terms is less with 1. Both implementations have used the same number of flip-flops (that is 3/64) and the same number of pins used like inputs/outputs (that is 9/32). We have preserved the state assignment of the FSM, and the subcircuits are in fact as pure combinational circuits. The interesting fact is that our GA have supplied a better solution than the one given by the minimization tool used for this purpose by the CAD software.

5. CONCLUSIONS

In this paper we have compared two different paradigms in digital design: the traditional digital design and the evolutionary digital design. Our goal was to optimize the digital circuit and to implement it with minimum resources in a CPLD.

We have shown that pure combinational circuits are implemented optimal, even if the boolean functions are faraway of their minimal form, that is software finds the optimal way in connecting the hardware resources of the circuit. Sequential circuits are more sensitive, because of the state assignment, but evolutionary design assures a better fitting of circuit resources in all cases that had been investigated.

Future research must be done in this area. Firstly it is important to find a better representation of the circuit in chromosomes, because complex functions need a great number of architecture bits, which directly influences the GA search space. EHW successfully succeeds only when fitness reaches 100% and in huge search spaces this condition may be not always possible. This is the main reason that for the time being the complexity of evolved circuits is so far small. Unfortunately, our circuit can not be used with intrinsic EHW, but other FPGA circuits may be used in intrinsic EHW experiments (Thompson, 1996).

ACKNOWLEDGMENT

The author would like to thank the Xilinx, Inc. for their academic donation, which consists in Xilinx Integrated Software Environment (ISE) 6.1i software and the Digilab XCRP circuit board provided by Digilent, Inc.

REFERENCES

Ali, B., A.E.A. Almaini and T. Kalganova (2004). Evolutionary Algorithms and Their Use in the Design of Sequential Logic Circuits. *Genetic Programming and Evolvable Machines,* **5**, 11-29

Coello, C.C., A.D. Christiansen and A.H. Aguirre (2000). Use of Evolutionary Techniques to Automate the Design of Combinational Circuits. *Int. Journal of Smart Engineering System Design,* **4**, 299-314.

Iba, H., M. Iwata and T. Higuchi (1996). Machine Learning Approach to Gate-Level Evolvable Hardware. *Proc. of the First Int. Conf. on Evolvable Systems ICES'96, Tsukuba, Japan, October 1996,* 327-343.

Popa, R. and M. Iliev (1999). Evolvable Synthesis of Digital Circuits. *Proc. of the 5-th Int. Workshop on Bases of Electronics SBE'99, Cluj-Napoca, Romania, 10-11 June 1999,* 206-211.

Thompson, A. (1996). An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics. *Proc. of the First Int. Conf. on Evolvable Systems ICES 96, Tsukuba, Japan, October 1996,* 390-405.

Wakerly, J. (2000). *Digital Design: Principles and Practices, Third Edition.* Prentice Hall, Inc., New-Jersey.