Fully Integrated Artificial Intelligence Solution for Real Time Route Tracking

Laurentiu–Mihai Ionescu[1)], Alin Mazare[1)], Adrian–Ioan Lita[2)], Gheorghe Serban[1)]

[1)] Faculty of Electronics, Communications and Computers, University of Pitesti, Romania
[2)] Politehnica of Bucharest, Romania
ioan.lita@upit.ro

Abstract: *In this paper the authors propose a solution in which an intelligent algorithm – genetic algorithm in our case – is used to generate commands for a robot in real time, so that the robot can determine the optimal moves considering several aspects: route tracking and low power consumption. Genetic algorithms are intelligent solutions for multi–criteria optimization and using them to find solutions to the optimization problems containing constrictions. However, they were designed as algorithms running on computer and therefore cannot ensure rapid generation of the solutions. On the other hand, the problem of determining the optimal response to command a robot requires real time response. The paper presents a method for hardware implementation and integration in a FPGA circuit of a genetic algorithm, in order to accelerate the convergence and to generate solutions in real time.*

1. INTRODUCTION

The FPGA circuits occurred in the mid-90s. They were made available for general purpose applications in 2000. One research area consists of partially or fully implementation of intelligent algorithms in FPGA reconfigurable circuits. Genetic algorithms (GA) – a class of intelligent algorithms based on artificial evolution – are bio-inspired solutions for finding the optimal response to a problem that has multiple criteria.

The concept of bio-inspired solution has been described first in 1970 [1] and the implementation solution was on computer. As a whole class of bio-inspired intelligent algorithms, the genetic algorithms [2] were implemented in software applications as sequential work algorithms. There are numerous applications and research, even commercial software tools using genetic algorithms. They are used to solving multi-objective problems [3].

In fact, all problems of optimizations, in any area can be solved using artificial intelligence solutions. With evolutionary algorithms there are more AI solutions like neural networks and fuzzy systems used to solve optimization problems [4].

Providing an optimal solution using software implemented algorithms occurs within a relatively large time and would not be realistically used as an application with real time response.

On the other hand, FPGA circuits allow the implementation of parallel processing solutions. Implementation of a genetic algorithm on FPGA modules means converting sequential blocks to parallel modules.

The problem of using software implemented genetic algorithms is the great convergence time. Also, being pseudo-random solutions, the convergence time varies and cannot guarantee a response within a certain amount of time. Thus, they cannot be used as applications with a real-time response. Instead, GA can be used as application optimization solutions to work in real time, as was shown in [5] or [6].

A new-appeared area which is continuously evolving is the one related to the implementation of the software algorithms on hardware structures, in order to parallelize some of their components and increase the response speed [7]. Especially this is possible on reconfigurable circuits, a flexible hardware solution for implementing various applications in many areas like data mining [8] or signal processing [9].

In the genetic algorithms, the hardware implementation solutions have allowed reducing the convergence time so that, even if this is variable, it can fit a maximum permissible limit for the applications with real-time response. There are several papers dealing with the implementation of genetic algorithms on different hardware structures. Name of HGA was introduced in [10] for presenting a FPGA partial integrated GA (some modules were running on computer). Later, there have been developed several solutions for full integration of GA in FPGA [11], [12] or ASIC [13] with outstanding results in accelerating the convergence.

The solution presented in this paper offers a convergence time less than 2 ms – regardless of the input values which are taken into account. This means providing solutions within a very short time and making them suitable for real-time applications.

The next section is dedicated for presenting genetic algorithms. In section 3 the system is described, the leaving section 4 for presenting results.

2. GENETIC ALGORITHMS

The genetic algorithms are search methods using pseudo-random algorithms. The search procedure is carried out in several stages [2]:

1. The solutions are encoded in chromosomes and individuals. Obviously for this step we do not know the solutions but we do know what they will represent. For example, if they represent real numbers then we will have chromosomes of real number type (floating or fixed-point representation).

2. A first set of random solutions is generated. This set is called the generation 0.

3. The population goes through the evaluation process. To make an assessment we need a purpose. The solutions are judged on how close or far away are from the intended purpose. For example, the purpose may be an expression of an equation.

4. Population goes through the selection process. Now each individual has got an associated note received at 3 that reflects the proximity of the sought solution. At this stage individuals are sorted according to the marks they received. Here the individuals for the next steps are selected. The method of selection may vary, the most used is the roulette one.

5. The crossover operator applies to the parents possibly with a certain probability. One or more crossing points are established – at the gene's level – and then genetic information from parents swap in the crossing points.

6. The mutation operator is a selection of an individual of the population – with a certain probability. One or more genes are randomly modified to this.

7. The individuals' elimination is optional – if it is to keep a population with a certain number of individuals. After stages 5, 6 and possibly 7 we have a new generation.

8. 3–7 is repeated until the solution is decided in 3. The evolutionary loop can also be stopped if the exceedance of a maximum permitted number of generations is observed.

3. HARDWARE GENETIC ALGORITHM

In figure 1 we have a block diagram of the system.

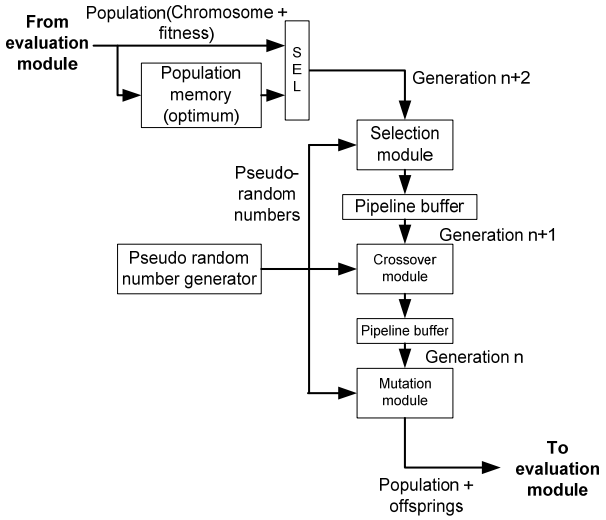

Fig. 1. Our solution of HGA. The pipeline buffers are dual ports BRAMS and ensure processing of more generations in same time.

All GA stages were implemented in hardware modules. First, the entire population was evaluated. Each individual has fitness (evaluation rating). Then the pair individuals-fitness was passed through the selection module. From the selection module, individuals reach the crossover module and the mutation module (modules responsible with genetic operations). Crossover and mutation generates new individuals (offspring) which are evaluated.

GA was implemented in a pipeline flow. Thus, while the generation *n* is to the input of mutation module, generation *n+1* brings parents for crossover operation, and the generation *n+2* are prepared for selection. All this is possible through the use of special SRAM memories that exist inside FPGA: Block RAM memories. They allow bi-port configuration: one port used for writing and another for reading. The evaluation module consists of arithmetic cells, which get the minimum for coordinator function and energy function.

As we mentioned, our system will be used for determining the position of a mobile. It wants to avoid obstacles and reduce energy consumption. If the mobile is at point coordinates (x0, y0), there is a function called T (x_0, y_0, x, y) which determines the possibility of following a new route to the coordinates (x, y). A high value of this function means that at the point coordinates (x, y) either an obstacle is found or an obstacle is near to that point.

It is impossible to write the value of such functions but can be obtained practical by using an ultrasonic sensor located on a 360 degrees rotating device.

Energy consumption is given by a function (function P) that estimates both the power consumption to perform a move to the point (x, y) (power consumption for moving to a flat ground) and the value of the function I – dependent on the slope until to the point (x, y):

$$E(x,y) = P(x_0,y_0,x,y) + I(x_0,y_0,x,y) \quad (1)$$

The amount of power consumption may be estimated by knowing the length of the distance to the x, y, while the slope can be determined by measuring the slope of the terrain (do it by means of a pair of ultrasonic sensors placed at different angles).

By performing a full rotation (360 degree) of the device with the sensors, we have values for the functions T and E. We obtain an expression of the form:

$$\alpha*T(x_0,y_0,x,y) + \beta*E(x_0,y_0,x,y) \rightarrow 0$$

It is therefore reduced to the determination of (x, y) coordinates (the unknown) in a linear equation with two unknowns. From the mathematical point of view it is impossible, but it is possible by using a solution based on heuristic pseudo GA.

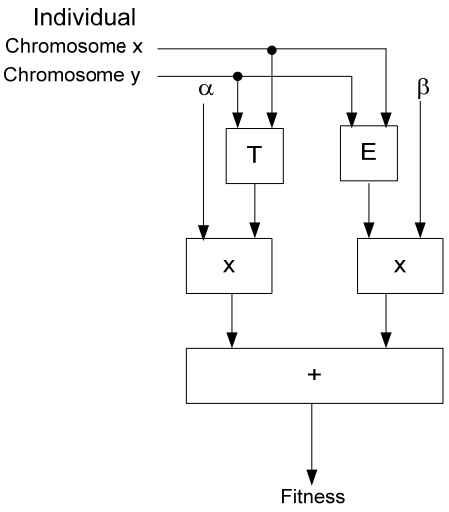

Fig. 2. Evaluation circuit for 1 individual. For each individual there is an evaluation module in genetic pipeline flow.

4. System Testing and Experimental Results

The system is implemented on a Zybo board that contains a FPGA Zynq 7000 manufactured by Xilinx in 22 nm technology.

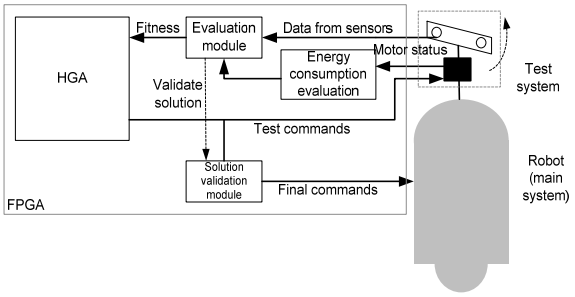

Fig. 3. Experimental scheme.

For sensing we used ultra sound sensors (pair) placed on a mobile fitting. The sensors are not in same horizontal plane and there is a difference between the heights where the sensors are placed. Table 1 represents experimental input data.

Tab. 1. Experimental inputs for sensors orientation.

Size	Value
Height of mobile system (including sensors fitting)	30 cm.
Difference angle between sensors	30 degree
Angle for highest placed sensor (for obstacle avoidance)	~63 degree
Angle for highest placed sensor (for slope of terrain)	~56 degree
Angle for lowest placed sensor (for slope of terrain)	~27 degree
Horizontal step rotation 1 angle for mobile fitting	18 degree
Advancing step size	60 cm

Genetic algorithm used in tests had the following parameters:

Tab. 2. Parameters of Genetic Algorithm used in experiments.

Parameter	Value
Number of individuals	10
Selection method	Roulette
Crossover ratio	1/generation
Crossover points	1
Mutation ratio	0.25
Mutation points	1

In table 3 we present results after implementation (synthesis and implementation).

Tab. 3. Resources used in Zynq 7000 Z7010.

Resource	Used	From	%
Slice Registers	7480	35200	21.25 %
Slice LUTs	17124	17600	97.3 %
Block RAM/FIFO	5	60	8.33 %
BUFG/BUFGCTRL	5	32	15.63 %
DSP48E1s	40	80	50 %
Slice Registers	7480	35200	21.25 %

For Zynq Z 7010 it is used 97 % from combinational resources. But this circuit is the smallest in this family.

Tab. 4. Response time.

Response time	Value
Convergence time (FPGA response one solution)	11 us
Response time ultrasound emission - reception	2 ms
Rotation / second	60

5. CONCLUSIONS

The proposed solution makes the determination of optimal next coordinate for a mobile system, using a hardware accelerator based on genetic algorithms. The whole system is integrated in a SOC circuit thus ensuring both hardware acceleration and interfacing with the user.

The results show a real time response of the system. A future research direction will be the extension of scope functions to introduce another search criterion.

REFERENCES

[1] J. Holland (1992), "Adaptation in Natural and Artificial Systems" Cambridge, MA: MIT Press. ISBN 978-0262581110.

[2] David E. Goldberg, "Genetic algorithms in search, optimization, and machine learning", Machine learning, Issue 2, Addison-Wesley, Reading, MA, 1989.

[3] Carlos A. Coello Coello, Gary B. Lamont, David A. Van Veldhuizen, "Evolutionary Algorithms for Solving Multi-Objective Problems", Springer Science+Business Media, LLC, 2007.

[4] Belu, Nadia; Anghel, Daniel Constantin; Rachieru, Nicoleta, „Failure Mode and Effects Analysis on control equipment using fuzzy theory", ModTech International Conference - Modern Technologies in Industrial Engineering Volume: 837 Pages: 16-21, 2013.

[5] Vaithiyanathan, D; Seshasayanan, R; Kunaraj, K, „An evolved wavelet library based on genetic algorithm", TheScientificWorldJournal Volume: 2014 Pages: 494319, 2014.

[6] Oklapi, E., Deubzer, M. ; Schmidhuber, S. ; Lalo, E., Mottok, J., "Optimization of real-time multicore systems reached by a Genetic Algorithm approach for runnable sequencing", 2014 International Conference on Applied Electronics (AE), 233 - 238, 9-10 Sept. 2014.

[7] Mihaela Maliţa, Gheorghe Ştefan, Dominique Thiébaut, "Not multi-, but many-core: designing integral parallel architectures for embedded computation", ACM SIGARCH Computer Architecture News - Special issue: ALPS '07--- advanced low power systems archive, Volume 35 Issue 5, December 2007, Pages 32-38.

[8] Iana, G.V.; Anghelescu, P; Serban, G., "RSA encryption algorithm implemented on FPGA", 2011 International Conference On Applied Electronics, SEP 07-08, 2011.

[9] Visan, Daniel Alexandru; Jurian, Mariana; Lita, Ioan, „Reconfigurable Platform for Versatile Generation of Communication Signals", Conference: 32nd International Spring Seminar on Electronics Technology, MAY 13-17, Pages: 233-236, 2009.

[10] S. Scott, A. Samal, S. Seth (1995), "HGA: a hardware-based genetic algorithm", FPGA '95 Proceedings of the 1995 ACM third international symposium on Field-programmable gate array, Pages 53-59, 1995.

[11] R. Faraji, H.R. Naji (2014), "An efficient crossover architecture for hardware parallel implementation of genetic algorithm", Neurocomputing, Volume: 128, Pages: 316-327, ISSN: 0925-2312.

[12] A. Swarnalatha, A.P. Shanthi (2014), "Complete hardware evolution based SoPC for evolvable hardware", Applied Soft Computing, Volume: 18, Pages: 314-322, ISSN: 1568-4946.

[13] Blaschke, J., Sebeke, C., Rosenstiel, W., "Using Genetic Algorithms for Planning of ASIC Chip-Design Project Flows" (2009), 2009 IEEE Congress On Evolutionary Computation, VOLS 1-5 Book Series: IEEE Congress on Evolutionary Computation Pages: 1881-1888, ISBN:978-1-4244-2958-5.