An Evolvable Driver for a Non-Linear Damped Pendulum

A. Delai[1], I. Bravo[2], J. R. de Oliveira[1], J. C. Garcia[2], J. Olivares[3]
[1]*Department of Computer Engineering and Industrial Automation, University of Campinas, C.P. 6101 13081 970 Campinas SP - Brazil*
[2]*Department of Electronics, University of Alcala, 28871 – Alcala de Henares, Madrid - Spain*
[3]*Department of Computer Architecture, Electronics, and Electronic Technology, University of Cordoba, 14071 – Cordoba - Spain*
ignacio.bravo@uah.es

***Abstract*—In this work, an evolvable hardware technique has been applied to a real control environment composed by a damped pendulum. A combinatorial 4 bit circuit input/output driver design by genetic algorithms will be implemented in an Altera Cyclone II FPGA, aiming to prove the efficiency of the evolvable design. A proportional driver tuned by a genetic algorithm (Kp-AG) was also implemented and subjected to the same tests to compare their performance against the Evolvable Hardware design. Experiments use a square reference signal to force overshoot errors, and, a sinusoidal signal to sample the response for both drivers. The results obtained by the experiments show that the evolutionary approach to design drivers can be competitive and improve considerably the control in this particular case of nonlinear control using low resolution feedback information.**

***Index Terms*—FPGA, evolvable hardware, genetic algorithm, nonlinear pendulum.**

I. INTRODUCTION

The use of evolutionary techniques for control systems purposes used to be restricted on parametric optimization, such as the use of genetic algorithms to adjust the parameters of PID controllers [1]. Afterwards, the use of an Evolvable Hardware (EHW) technique for building a damped pendulum digital driver was proposed in [2], in which the information to design the driver is only the feedback of their behaviour through testing. This technique got beyond parametric optimization, using heuristics for designing a logic circuit dedicated to control.

EHW consists in integration of evolutionary computation and programmable hardware devices [3]. This is an active research area whose the first researches started in the beginning of the 1990's [4]–[7], starting to deal with problems involving combinational logic circuits design. Usually evolvable techniques require parallel programming and are suitable for Field Programmable Gate Arrays (FPGA) design [8], [9], being the main objective is to provide an autonomous hardware design with minimal or no human intervention.

In this context the design of electronic circuits often require the ability to develop circuits of increasing complexity, which requires novel techniques to solve effectively and time and combinatorial complexity problems. Intelligent techniques like simulated annealing [10] and artificial neural networks [11] are frequently used to search for alternative solutions to classic engineering methods for optimization electronics systems.

This paper proposes a hardware implementation of a 4 bits digital control circuit for a real nonlinear damped pendulum, based on the simulation work produced in [12], extending that proposal to a real world application using Field Programmable Gate Arrays (FPGAs), and collecting significant data for the validation of the technique. The generation of prototype tests based on such technology aims to propose an alternative solution to the problem of controlling nonlinear systems, such as control of robot manipulators [13], which motivates a new application of EHW in the control area that is not commonly used.

This work consists in integrating mechanical and electrical parts and designing the digital systems to program the FPGA. The FPGA contains a pendulum driver and other necessary functions, such a state machine and a PWM generator. The driver basically works controlling the energy flux to a DC motor that moves the gearbox attached to the pendulum, switching between higher and lower energy levels and changing the motor polarity. The feedback is performed by an incremental encoder coupled with the axis of the pendulum and inserted into the digital driver as a closed-loop control. The use of FPGAs in EHW projects is appropriate because such devices allow rapid prototyping and reconfiguration.

This work is organized as follows. In Section II we introduce the preliminary concepts and the general proposal of the EWH technique. Section III refers to the preliminary work, the fundamentals and the characteristics of the problem, and also the circuit representation using GA chromosomes. The general scheme of the work, including the used hardware, graphics and tables is described in Section IV, followed by the conclusions in Section V.

Manuscript received February 26, 2013; accepted December 19, 2013.
This research was supported by University of Alcala through the project IPRIM (ref. CCG2013/EXP-064) and Spanish and Brazilian Program through EINTA project (ref. (PHB2006-0077-PC)).

II. THE EVOLVABLE HARDWARE TECHNIQUE

Since the last decade, many researches have been applied to study of the design of electronic circuits, using principles based on the neo-Darwinian theory: this process establishes the survival of the most adapted species in a changing environment. EHW proposes an automatic design of electronic devices based on a behavioural description of the system, searching for more general solutions. The process underlying the evolution can be called EHW engine, which generates the evolution steps necessary for the automated design of the circuit (Fig. 1). The EHW engine can be provided by algorithms inspired by biological evolution, characterized by the research line called Evolutionary Computing. EHW can find new design methodologies that may result in circuits and architectures with different behaviours due to the exploitation of the search regions often ignored by traditional methods [14].

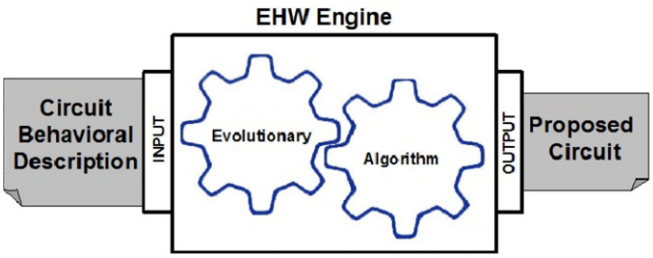

Fig. 1. EHW process.

Evolutionary Algorithms are typically applied to solve search problems in the form of $f : S \to \mathbb{R}$, where S is the search space which consists of all possible solutions for a particular problem [14]. Depending on the particularities of the problem, the solutions can be represented by n-dimensional vectors composed by binary, integer or real numbers. A real number, called Fitness Value, is associated to all existing solutions. This number measures the quality of a particular individual to solve that specified problem. The process that scores the individuals is called Fitness Function. Basically, the task of an evolutionary algorithm is sampling the search space S to finding the best solutions in accordance with the problem, using the Fitness Value as a guide.

In this case, the EHW engine uses a collective learning process on a limited population of individuals (circuits), which implies in a parallelism in the search for solutions. Usually each individual represents (or encodes) a point in the search space of potential solutions to the problem under study. An evolutionary process is applied to the population, allowing after several generations the emergence of better individuals than its predecessors.

Inside the EHW engine, an evolutionary algorithm works in a loop until a limited number of generations is reached or a satisfactory circuit is found, as observed in Fig. 2. There are basically 4 cyclic steps: selection, crossover, mutation and evaluation. First, a random circuit population is created. After that, the population need to be evaluated; in the evaluation step, every individual (circuit) receives a Fitness Number that corresponds to their adaptability (how well it does the job). Using the fitness value, the selection chooses individuals to be part of the reproduction process by crossover. The crossover step mixes the chromosome genetic material of the parents chosen by selection, generating a new population. In the mutation step some circuits' parts are changed to produce more diversity and insert new individual characteristics. There are no guarantees of finding the optimal circuit solution, because of the size and complexity of the search space, but it is possible to find very good and satisfactory circuits for some applications.

This heuristic technique works exploring the search space where we can also find circuits obtained through classical design techniques (Karnaugh Map [15], Quine-McCluskey Algorithm and Boolean algebra), but also explore regions of space where there are unconventional circuits, which may be more suitable for some cases. This is the main EHW advantage in addition to design automation. The way that the Evolutionary Algorithm (EA) explores the search space depends on the circuit encoding and others parameters like crossover probability and mutation rate. According to [14], it can be summarized that the most notable features of circuits observed in both, intrinsic and extrinsic evolution:

– Potential to find novel circuits;

– The possibility to find new design rules from the novel circuits obtained;

– Evolutionary methods can manage a larger set of design specifications compared to human design techniques;

– Evolutionary systems have been able to achieve competitive circuits when compared with the state of the art in electronics;

– Evolutionary tools are more appropriate to analog design, this fact could produce a new trend in the research on analog circuits.

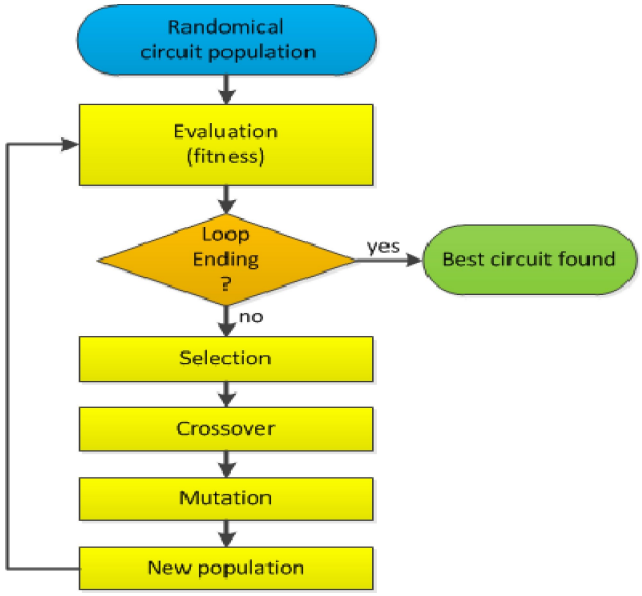

Fig. 2. EHW steps diagram.

EHW was first proposed in the early 90s for hardware design. It falls into two categories: original design and adaptive systems. Original design uses EA to get a system that meets a predefined specification. Adaptive systems reconfigure an existing design to fault tolerance or to adapt it to a changing operational environment. EHW can be used as an alternative to conventional hardware design methodology [16].

The digital bottom-up circuit design that uses heuristics

like evolutionary algorithms have a problem of scalability, currently limited to 16 input bits [12]. This limit is due to the fact that the search space increases considerably when adding new input bits in the evolving circuits, increasing chromosomes sizes that representing them. Many minutes, hours or even days of computational execution are often spent on a EHW process, depending on the approach used and the target circuit complexity.

EHW has three evolution methods. The first one is the extrinsic evolution where the system is entirely simulated and the circuit is tested inside the simulation environment. The second one is in the intrinsic evolution that evolves the circuits in software environment and tests it in a hardware environment, and finally, the mixtrinsic evolution which mixes intrinsic and extrinsic changes in order to achieve greater circuit portability [14]. The most popular devices used to evolve digital hardware (intrinsic/mixtrinsic) are the FPGAs due to their characteristics of reconfiguration capability, design flexibility and very high logic capacity. The class of evolutionary algorithm used in this case is the Genetic Algorithm, chromosomes can be implemented as circuits, and it is possible to do a parallelized search to find the best design.

III. The Case Study Description

The study realized in [2], [12] proposes the automatic design of a digital controller with nonlinear response characteristics applied to the pendulum control using a genetic algorithm (GA) [17] as the EHW engine. For comparison purposes, the evolution (also by GA) of the classic Proportional Derivative controller (PD) was done. Thus, the results obtained for both classes of controllers: classic PD, which was tuned by GA, and the EHW driver.

A. Nonlinear Pendulum Characteristics

The control target is the damped nonlinear pendulum. The equation that governs the pendulum movement is:

$$ml^{2}\ddot{\theta} + kl^{2}\dot{\theta} + mg\sin\theta = 0, \qquad (1)$$

where m = 0.210 kg is the pendulum rod mass, g = 9.82 m/s^2 is the gravity acceleration, k = 0.3 Nms/rad is the viscosity of the environment, l = 0.285 m the rod size and θ the rod angle. These parameters were obtained based on the physical model of the pendulum used in laboratory tests. The choice of the PD controller over the PID was made because the first one has scalability acceptable to the designing technique applied in the EHW controller. The pendulum model was simulated to evaluate the control system performance [2], [12]; this model is illustrated in Fig. 3.

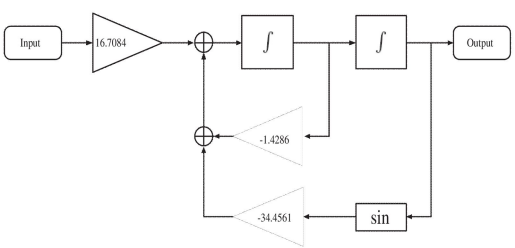

Fig. 3. Simulink pendulum model. The gains were adjusted based on the physical pendulum structure.

B. PID Controller tuned by GA

One of the main points in the evolutionary process of a genetic Proportional Integrative Derivative Controller (PID) is the definition of a chromosomal representation, containing the gains like genetic information. The chromosome is a binary string divided in three parts to represent the three gains: proportional, derivative and integrative (2)

$$(KP, KD, KI) = \underbrace{010\ldots00}_{KP}\,\underbrace{101\ldots01}_{KD}\,\underbrace{111\ldots10}_{KI}. \qquad (2)$$

In this chromosome, each gain is represented by a digital number coded with floating point or integer representation. Basically, the GA changes the bits by crossover and mutation exploring the search space to find the best gains to the controller. Each chromosome is tested and evaluated. In the selection process, the set of parameters that represents the better control performance will have more probability to be chosen to participate in the reproduction process [12].

C. Driver Evolution

For the electronic circuit design, the problem domain is the derivation or the circuit construction and the solution obtained from this process, which behaves according to the given specifications. The first step after obtaining the behavioural specifications is to define the individual representation (the circuit representation) and a function that calculates their "skills" in solving the problem. After that, it is necessary to consider how the evolutionary operators, such as crossover and mutation, are applied to the representations to provide increased individuals adaptability through the generations.

In the GA utilization the circuits are represented by chromosomes. One example is the codification presented by [14] where the combinational logic circuits can be represented by a sum of products of the system logic variables (Fig. 4).

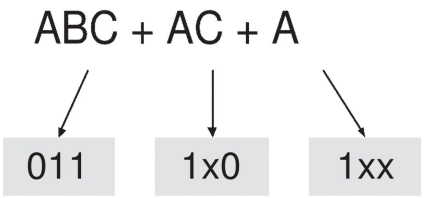

Fig. 4. Sum of products chromosome representation.

The resulting chromosome is a ternary string. Each position in the chromosome can assume the values "0" for the complemented variable, "1" for non-complemented variable and "x" to the absence of variable. That is a simple codification but suitable to be used in circuits containing only one output.

The codification used to evolve the driver is a gate based codification that represents the circuits by netlists (Fig. 5). In the chromosome shown in that figure, N1 represents the first input connection of a logic gate, N2 is the second input connection of the same gate, "-" indicates that there is no third input connection in this case and, at last, the AND2 represents the type of the logic gate and the number of inputs. These types are predefined in a list of gates allowed to the circuit. The notation is repeated along the other parts

of the chromosome. This representation allows $n \geq 1$ where n represents the circuit outputs, and also allows a variable size circuit representation and manipulation by simple genetic operators.

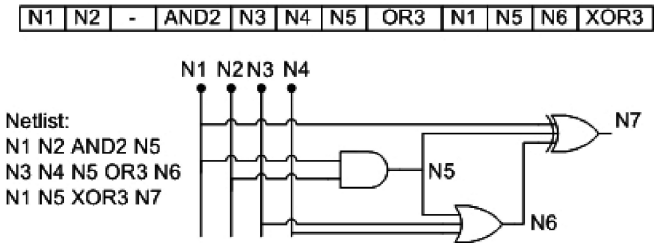

Fig. 5. Netlist chromosome representation.

The driver chromosome was defined as a binary string using 6 bits to each gate input connection N and 3 bits to the gates function (BUF, INV, AND, NAND, OR, NOR, XOR, XNOR).

As seen in Fig. 2, the first step to evolve the driver is the creation of a random initial population of chromosomes. Using extrinsic evolution, the PSPICE simulator is called by Simulink to decode the chromosomes and generate the respective logic circuit (a bitstream becomes a circuit to simulation testing). In this process, the circuit is evaluated using the Simulink pendulum model applying a square function described by (3) as the reference signal

$$r(t) = \frac{®}{2\left[«(t) - «(t-8) - «(t-16) + «(t-24)\right]}. \quad (3)$$

The preparation of a new iteration involves the generation of a new population of individuals based on selected individuals for prior population. The process of selection (through the technique of roulette elitist runs) ensures keeping the best individual of previous generation into the current one. Then the population undergoes crossover and mutation with rates of 90 % and 5 % respectively [12]. The mutation uses a high value due to the large search space of the problem. Both parameter values seek diversity in the solutions obtained through the evolutionary process. The single-point crossover recombines the chromosomes from a randomly chosen position, called the cut-off point. This operation performs the exchange of bits of two strings (chromosomes) generating a new chromosome with paternal inheritance. In the mutation operation a number of points are also chosen randomly according to the predefined mutation rate and these points are then reversed. Finally a new generation was produced and should be evaluated by the Fitness Function, before starting a new iteration of the algorithm. The process will be repeated until a reading a predetermined number of generations, usually a few thousand.

D. Fitness Function

In an evolutionary algorithm, during the selection process, all generated solutions are evaluated by a Fitness Function which reflects the adaptability level of an individual. The function chosen to evaluate the EHW driver and the Kp-AG driver is based on minimizing the index value error Integral of the Square Error (ISE) [12]. The Fitness equation (4) describes the Fitness has an error minimization guide to the GA, where *Jise* is the Integral of the Square Error value.

$$Fitness = f(Jise) = \frac{1}{1 + Jise}, \quad (4)$$

If the *Jise* = 0 (no errors) the Fitness will be 100 % to the circuit analysed, that indicates an optimal driver circuit to the problem in this particular case.

IV. APPLICATION

The implementation objective in building a real model based on a theoretical/simulated system is to demonstrate experimentally the behaviour and functionality of this design approach in real environments. This system was implemented in an Altera FPGA EP2C20F484C7N Cyclone II using schematic and VHDL programing. All the necessary circuitry: the EHW and Proportional drivers, state machines, PWM generators, was implemented in this single chip. The driver's input (EHW and Kp-AG) is a 4 bit error signal generated by comparison between the references values and the pendulum real position, using the 2's complement code. Equation (5) corresponds to the error used as the feedback, where *e(t)* is the error signal, *r(t)* the reference signal and the *y(t)* is the pendulum position

$$e(t) = r(t) - y(t). \quad (5)$$

The output is a 4 bit word which corresponds to the PWM control levels, like shown in Fig. 6.

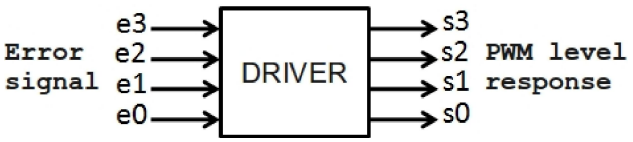

Fig. 6. Diagram of the driver (EHW and Kp-AG).

Figure 7 shows both drivers behavioural. We can observe the EHW driver (green line) has a nonlinear characteristic due to the approaching based on experience, opposing the Kp-AG method (red line) based on parameter tuning of a linear driver.

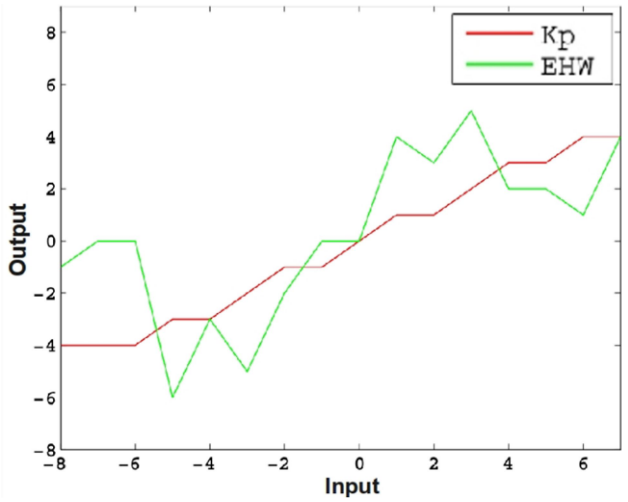

Fig. 7. Kp and EHW behavioural.

A. The control system implementation

The plant used in the benchmark experiments was an

electromechanical pendulum [18]. The system has the same behaviour that the simulation model that was used in the evolvable process. An Electro-Craft E-530 CC motor (24 Vcc/24 W) is coupled to a reduction gearbox system on the stem of the pendulum. An incremental optical encoder that allows 4000 different position points provides the pendulum position feedback.

The reference signal was generated by a computer commanding the pendulum to a specific position. Also, it was used a computer running the MATLAB XPC Target system with a digital acquisition board (NI PCI-6503, 1 kHz Sampling) to acquire the encoder signals. Figure 8 shows a system diagram whose components were physically assembled. Figure 9 shows the pendulum and FPGA board image.

The FPGA works at 50 MHz (maximal frequency) and the Quartus II version 8.1 was used to program the FPGA. Inside the chip there are blocks witch compose the digital modules (Fig. 10). The first block (ADD) calculates the error and generates a saturated 4 bit error signal (5) to be sent to the next block (EHW/Kp(AG) controller) which responds sending a 4 bit control word to the PWM generator. This generator will convert the control word into a PWM level, which is forwarded from the FPGA to the power module responsible to control the pendulum energy through a voltage level converter (buffers interface). The position signal is produced by the incremental encoder and sent back to the FPGA feedback (after a voltage conversion) to be decoded generating a 12 bit position value. The whole system consumes less than 5 % of the total logic elements available in the FPGA chip.

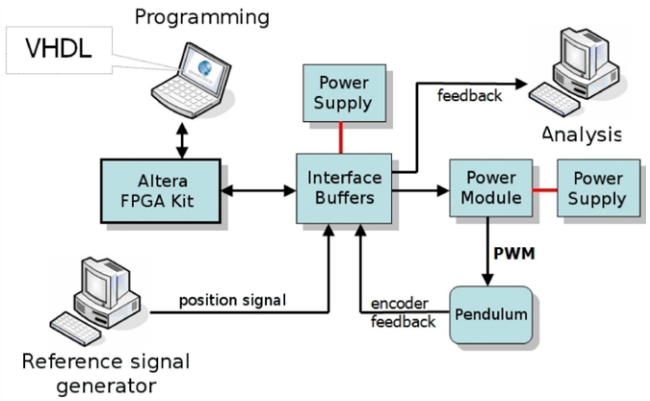

Fig. 8. Benchmark diagram.

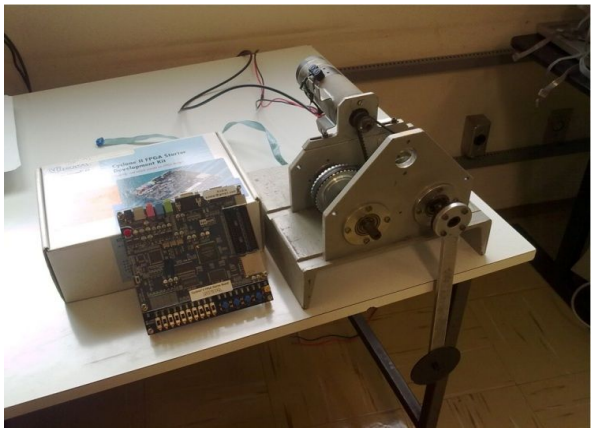

Fig. 9. The pendulum system.

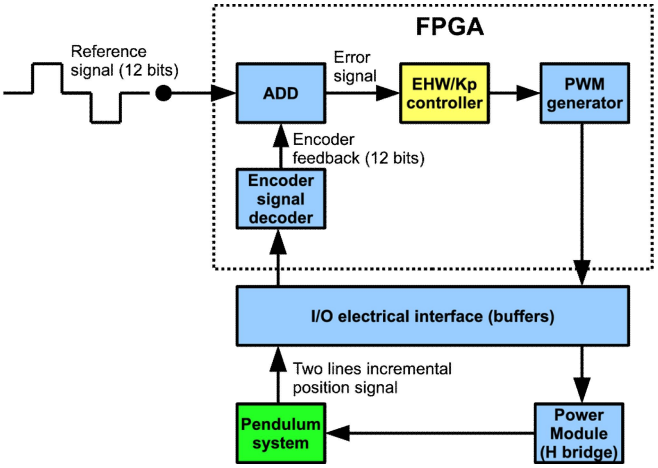

Fig. 10. Simplified FPGA content diagram.

B. *Results*

Figure 11 shows the square reference signal to set the pendulum rod to move 90 degrees clockwise, go back to the origin (perpendicular to the gound), move 90 degrees anticlockwise, go back to the origin where the test stops. In both cases the driver goal is to approximate the second-order system to a first-order system behaviour using only the 4 bit feedback information available.

Analysing both Fig. 11 and Fig. 12, it is possible to see that the EHW driver non-linear approach, in general, reduces the overshoot and approximate the response to the reference better than the Kp-GA controller. This happens because the EHW driver explores control solutions beyond the limitation of the GA parametric optimization of the proportional driver.

The drivers EHW and Kp(AG) were tested separately at a time. Two different reference signals were used in the benchmark test (3). The first one, the square reference, was the signal used to compose the fitness function (Fig. 12). The second signal was a *Sin* function not used previously in the evolution process, totally unknown to the driver before the benchmark test (Fig. 13 and 14).

The Root Mean Square (RMS) error was used to provide a comparison between EHW and the Kp(AG) driver (6),

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^{\,2}}, \tag{6}$$

where n is the total number of error points and e the i^{th} respective error value

The calculated RMS values for square and sin references are exposed in Table I. The table shows the information quantified in encoder counts.

TABLE I. RMS VALUES.

	EHW Driver	**Kp(AG) driver**
Square reference	0.3564	0.3879
Sin Reference	0.0188	0.0189

In general, to the square reference (Fig. 11) the RMS difference between the EHW and the GA proportional driver is 0.0315 (Table I), and to the *Sin* reference is just 0.0001, indicating that the EHW was superior in the first case and equal to the GA proportional in the second case.

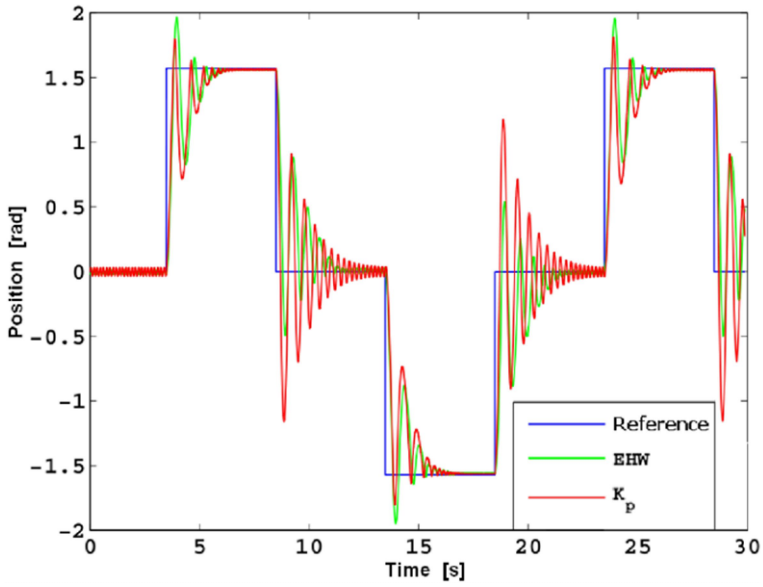

Fig. 11. Temporal response of the position signal.

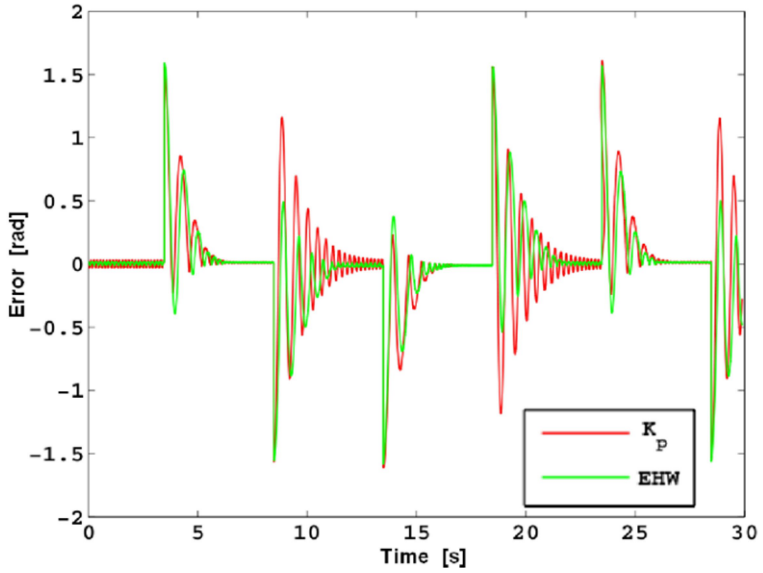

Fig. 12. Temporal response of the error signal. Graphical analysis (square reference).

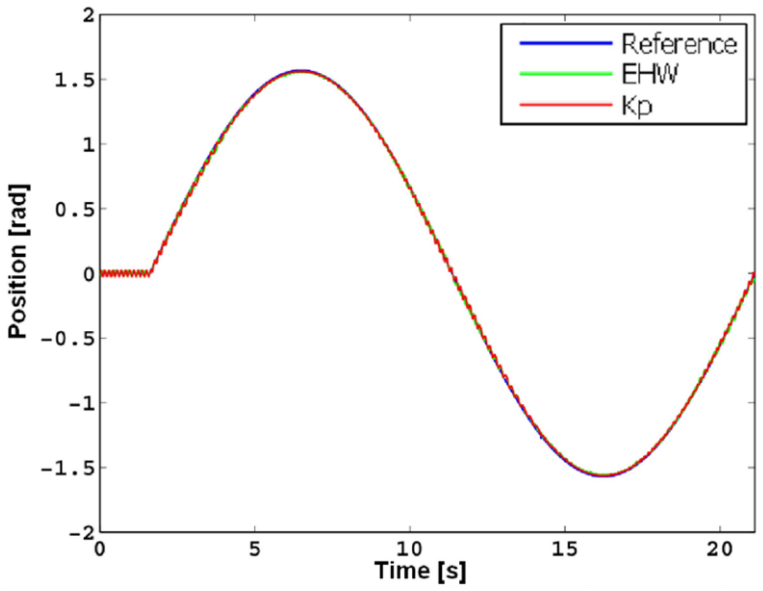

Fig. 13. Temporal response of the position signal.

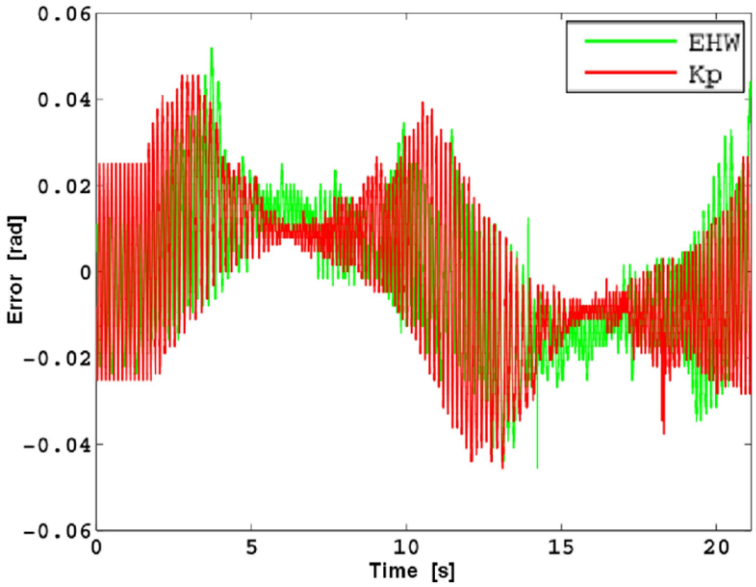

Fig. 14. Temporal response of the error signal. Graphical analysis (*sin* reference).

V. CONCLUSIONS

A practical experiment involving a novel design technique for a nonlinear damped pendulum driver was implemented and described in this paper. Starting with a functional criterion to build a driver (a black box approach), a physical system was built to test and generate data to evaluate the actual performance.

When dealing with a nonlinear problem designing nonlinear solutions the evolutionary method shows to be competitive in real environment tests, in this particular case (evolved versus proportional AG control). The empirical process generates a driver with nonlinear behaviour, modelled to fit in actual pendulum system and produce control signals more adapted to the problem.

The collected data in the tests show a better performance to the EHW driver, 8,12 % better than Kp(AG) to the square reference (Table I). Concerning the *Sin* reference the both methods achieved the same performance.

Despite the problems with limited scalability of EHW systems, the results show that this technique is a promising perspective that deserves careful attention. As a perspective, we aim to implement the system using the Proportional Derivative control and the EHW equivalent driver to take data for new analysis.

REFERENCES

[1] M. L. Steinberg, A. B. Page, "Nonlinear adaptive flight control with genetic algorithm design optimization", *Int. J. of Rob. and Nonl. Cont.*, no. 9, pp. 1097–1115, 1999.

[2] T. J. De Campos, J. R. De Oliveira, M. Jungbeck, "*Evolvable hardware a new approach for control design*", in *Proc. of the 7th WSEAS Int. Conf. Evolutionary Computing*, 2006, pp. 28–32.

[3] T. Higuchi, Y. Liu, X. Yao, *Evolvable Hardware (Genetic and Evolutionary Computation).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. [Online]. Available: http://dx.doi.org/10.1007/0-387-31238-2

[4] H. de Garis, "*Genetic programming: Artificial nervous systems, artificial embryos and embryological electronics*", *Proc. of the 1st Workshop on Parallel Problem Solving from Nature*, London, UK: Springer, 1991. [Online]. Available: http://dx.doi.org/10.1007/BFb0029741

[5] S. J. Louis, G. J. E. Rawlins, "*Designer genetic algorithms: Genetic algorithms in structure design*", in *Proc. Fourth Int. Conf. Genetic Algorithms*, 1991, pp. 53–60.

[6] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, T. Furuya, "Evolving hardware with genetic learning: a first step towards building a darwin machine", in *Proc. second Int. Conf. From animals to animats 2: simulation of adaptive behaviour*. Cambridge, MA, USA: MIT Press, 1993, pp. 417–424.

[7] A. Thompson, "*Evolving electronic robot controller that exploit hardware resources", in Proc. of the 3rd European Conf. on Advances in Artificial Life*, London, UK: Springer-Verlag, pp. 640–656, 1995. [Online]. Available: http://dx.doi.org/10.1007/3-540-59496-5_332

[8] R. Popa, D. Aiordachioaie, G. Sirbu, "*Evolvable hardware in Xilinx spartan-3 fpga*", in *Proc. Int. Conf. on Dynamical systems and control (WSEAS)*, 2005, pp. 66–71.

[9] J. Wang, C. Piao, C. Lee, "Fpga implementation of evolvable characters recognizer with self-adaptive mutation rates", *Adaptive and Natural Computing Algorithms*, Springer Berlin / Heidelberg, vol. 4431 of Lecture Notes in Computer Science, pp. 286–295, 2007.

[10] C. Sechen, "Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing", in *Proc. of the 25th ACM/IEEE Design Automation Conf*., 1988, pp. 73–80.

[11] J. S. Yih, P. Mazumder, "A neural network design for circuit partitioning", in *Proc. 26th ACM/IEEE Design Automation Conf.*, 1989, pp. 406–411.

[12] T. J. de Campos, "Hardware evolutivo aplicado a geraçao automatica de controladores para servo-mecanismos", Ph.D. dissertation, University of Campinas, Brazil, 2007.

[13] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, *Robotics: Modelling, Planning and Control.* Advanced Textbooks in Control and Signal Processing. Springer-Verlag, 2nd Ed. 2011.

[14] R. S. Zebulum, M. M. B. Vellasco, M. A. Pacheco, *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*. Boca Raton, FL, USA: CRC Press, Inc. 2001. [Online]. Available: http://dx.doi.org/10.1201/9781420041590

[15] M. Karnaugh. "*The map method for synthesis of combinational logic circuits*", *Trans. American Institute of Electrical Engineers*, vol. 72, no. 9, pp. 593–599, 1953.

[16] Z. Bao, T. Watanabe, "A new approach for circuit design optimization using genetic algorithm", in *Proc. of SoC Design Conference, (ISOCC 2008)*, vol. 01, 2008, pp. 383–386.

[17] J. H. Holland, *Adaptation in natural and artificial systems*, 2nd ed., Cambridge, MA, USA: MIT Press, 1992.

[18] M. A. T. de Souza, "Otimizaçao de Controladores Nebulosos de Takagi-Sugeno utilizando Algoritmos Genéticos", M.S. thesis, School of Electrical and Computer Engineering (FEEC) - University of Campinas (UNICAMP), 2000.