

# A. Artículos

## A.1. Aplicaciones

### A.1.1. En Hardware Evolutivo

Las plataformas desarrolladas en este trabajo fueron utilizadas como una forma novedosa y económica de implementar algoritmos de hardware evolutivo, la posibilidad de tener una FPGA y un procesador en la misma placa permitió la implementación de algoritmos genéticos de forma intrínseca, es decir, verificando el nivel de ajuste de cada individuo a la solución requerida dentro del mismo dispositivo. En esta área se generaron dos artículos:

**Intrinsic Evolvable Hardware for Combinatorial Synthesis Based on SoC+FPGA and GPU Platforms** Será publicado en la conferencia *Genetic and Evolutionary Computation Conference GECCO 2011*; clasificada 7ma entre 711 conferencias en inteligencia artificial (AI conference ranking) y es publicado por ACM (Association for Computing Machinery<sup>1</sup>)

---

<sup>1</sup>ACM, es la sociedad educativa y científica informática más grande del mundo.

# Intrinsic Evolvable Hardware for Combinatorial Synthesis Based on SoC+FPGA and GPU Platforms

Carlos Camargo Bareño

Universidad Nacional

cicamargoba@unal.edu.co

Luis Fernando Niño

Universidad Nacional

lfninov@unal.edu.co

Cesar Pedraza Bonilla

Universidad Santo Tomás

cesarpedraza@usantotomas.edu.co

Jose Martinez Torre

Universidad Rey Juan Carlos

joseignacio.martinez@urjc.es

## ABSTRACT

This paper presents a novel a parallel genetic programming (PGP) boolean synthesis implementation on a low cost cluster of an embedded open platform called SIE. Some tasks of the PGP have been accelerated through a hardware coprocessor called FCU, that allows to evaluate individuals onchip as intrinsic evolution. Results have been compared with GPU and HPC implementations, resulting in speedup values up to approximately 2 and 180 respectively.

## Categories and Subject Descriptors

B.5.2 [Hardware]: R-T-L implementation—*optimization*

## General Terms

Design, Performance

## Keywords

Hardware copyleft, Evolutionary, boolean synthesis

## 1. INTRODUCTION.

One of the main goals in combinatorial synthesis consists of finding compact boolean expressions in the sum of products (SOP) form with the smallest possible number of variables and terms. Boolean algebra offers a way to find compact expressions, but this process depends on the designer's experience, resulting in non-optimal or inadequate expressions. There are other techniques available for combinatorial synthesis such as the Karnaugh maps, the Quine-McCluskey algorithm, the Reed-Muller algorithm. In general terms, these algorithms have disadvantages such as exponential complexity, lack of restrictions management, and multiple solutions.

As an alternative to the traditional design of combinatorial circuits, some authors have proposed bio-inspired tech-

niques to create combinatorial circuits that can not be obtained with the traditional methods and to add some restrictions to the design such as delay, area, etc. Nicholson [14] has used Simple Genetic Algorithms (SGAs) with a fixed length representation for small problems, Kajitani [11] has worked with Variable-length Genetic Algorithms (VGAs) evolving up to 6-bit problems. Aguirre *et al* [1] used tree-based genetic programming (GP) for evolving small circuits, Xu [23] has worked with adaptive immune GA obtaining only 4-variable circuits, as well as Coello with ant colony algorithms [3]. Others authors have proposed a platforms to use reconfiguration techniques with hard-time restrictions due to the high reconfiguration latency [20, 13, 19].

One of the main problems to create circuits by using these techniques is the response time, due the high computational requirements for implementing any bio-inspired algorithm. As a result, only small circuits can be created in reasonable time [21, 12]. To solve the scalability restriction of creating digital circuits by using parallel genetic programming (PGP), we have developed a low cost cluster-based SoC + FPGA architecture called SIE. A fitness coprocessor unit (FCU) was developed on each FPGA in order to accelerate the convergence of a Intrinsic PGP (individuals are tested in the same platform), as well as provides an appropriate support for 8 variables synthesis problems.

These strategies allow to create combinatorial blocks that cannot be obtained with traditional methods, and to add some restrictions to the design such as delay and area. These designs have a very low limited number of variables [4] and they are mainly oriented to obtain a few basic structures.

In order to use parallel genetic programming (PGP), an FPGA cluster-based architecture to solve the combinatorial synthesis problem on-chip has been developed. The fitness coprocessor unit (FCU) on each FPGA helps to accelerate the convergence of the algorithm, as well as provide an appropriate support for 12-variable synthesis problems. The success of the system is mainly due to the capability of evaluating a chromosome in the FPGA through a virtual LUT-oriented architecture without using high-latency partial reconfiguration techniques, and determining the fitness value for an individual faster than other related works.

Due to that the capability of GP to run on massive parallel architectures, the proposed synthesis algorithm was compared to a High Performance Cluster (HPC) and a graphics processing unit (GPU) implementations.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO'11, July 12–16, 2011, Dublin, Ireland.

Copyright 2011 ACM 978-1-4503-0690-4/11/07 ...\$10.00.

## 2. GENETIC PROGRAMMING IN COMBINATORIAL SYNTHESIS

This section describes some of the most important aspects of the proposed evolutionary algorithm for the combinatorial synthesis problem, such as chromosome representation, the fitness problem and the genetic operators.

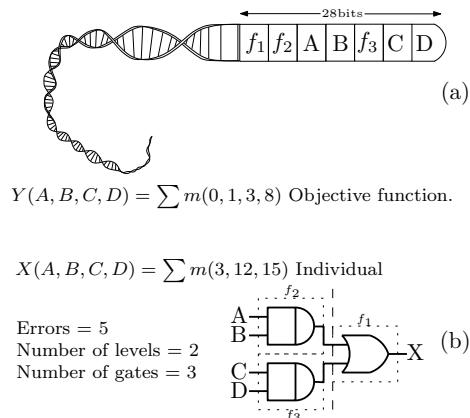
### 2.1 Chromosome representation.

The codification is the way a logic circuit is represented using a bit array in order to be used in the evolution process [5]. This representation must be able to represent all the different solutions to the problem, also the crossover and mutation operators should not generate infeasible individuals, and it must cover all the solution space. There are different ways of representing combinatorial hardware for a genetic algorithm: tree-based representation in 2-D, PLD-like structures and cartesian representation are some of them [8] [18] [9].

The 2-D tree representation is appropriate for implementing parallel systems because it allows splitting the chromosomes for balancing the computational load [15]. Figure 1 shows the selected cell-based structure representation. Each basic cell has 3 functions  $f$  and 4 input variables  $v$  coded in binary. This representation allows the addition of more cells to represent larger circuits. It must be mentioned that the chromosome length has to be variable because the length of the solution to the synthesis problem is unknown a priori.

### 2.2 Fitness function.

Finding the appropriate fitness function is important because it is responsible for quantifying the way a chromosome or individual meets or does not meet the requirements.



**Figure 1: Cell-based structure representation.**

$$fitness = \omega_1 \left[ \sum_{j=1}^m \sum_{i=1}^n Y(j, i) - X(j, i) \right] + \omega_2 \cdot P(x) + \omega_3 \cdot L(x) \quad (1)$$

In equation 1 the fitness function for the proposed GP is shown. Constants  $\omega_1$ ,  $\omega_2$  and  $\omega_3$  are used for establishing the weights of each of the parameters that will determine the fitness function. The double-summation term calculates the number of matches of the individual  $X$  for all the pos-

sible combinations at the output with the target functions  $Y$ ; the  $P(X)$  function is used for calculating the number of logic gates of a chromosome taking into account some of the *introns* or segments of the genotype string that will not have any associated function and that do not contribute to the result of the logic circuit that they represent. The function  $L(X)$  is used for determining the number of levels the circuit has, or in other words the number of gates that the critical path crosses. The  $m$  constant is the number of outputs in the circuit and  $n$  the number of possible combinations of inputs in the circuit.

### 2.3 Genetic operators.

The selection operator is responsible for identifying the best individuals of the population taking into account the exploitation and the exploration [15]. The first one allows the individuals with better fitness to survive and reproduce more often, and the second one means searching in more areas and making it possible to find better results. In the other hand, the mutation operator modifies the chromosome randomly in order to increase the search space. It can change: 1) an operator or variable and 2) a segment in the chromosome. Both are executed randomly with a certain probability. A variable mutating probability during the execution of the algorithm (evolvable mutation) [16] is more effective for evolvable systems. Finally, the crossover operator combines two selected individuals for obtaining two additional individuals to add to the population. A crossover system with one or two crossover points randomly selected has been implemented because it is more efficient for evolvable systems [10].

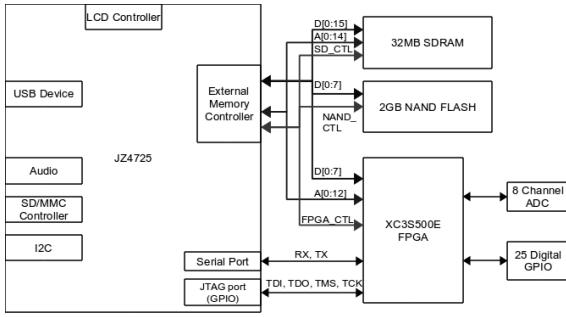
## 3. COPYLEFT SIE PLATFORM.

The hardware project copyleft SIE [?] [?] created by our team work, is composed of a custom embedded platform and a software development kit based on Linux operating system; allowing the generation of commercial applications under the Creative Commons BY-SA licenses, which allows the distribution and modification of the design (including commercial applications), with only the requirements that the derived products be under the same license and that the proper credit be given to the author of the original work. This is what defines the base of the *hardware copyleft* products.

### 3.1 The SIE Platform.

The SIE platform is composed by a System-On-a-Chip processor and a FPGA (Figure 2). The SoC is a MIPS processor (Ingenic JZ4725) running at 400MHz with peripherals that allows to control: a 2GB NAND memory for file storage, a 32MB SDRAM, a serial communication channel (UART), micro-SD memories, I2C port, 3-inch color LCD display, 2 input and 1 output audio stereo interfaces and 2 analog inputs. The XC3S500E Xilinx FPGA gives 25 GPIOs and controls an 8 channel analog-to-digital converter. There are two communication channels between the FPGA and the processor, the first one is a JTAG that allows the FPGA configuration and the execution of tests to the implemented circuits. The data, address and control buses are part of the second channel, which is used to exchange information between the processor and the peripherals implemented inside the FPGA.

SIE provides power connection through a USB port, which



**Figure 2: SIE development platform structure**

is also configured as a network interface. This allows the file transfer and the execution of a remote console using the *ssh* protocol. This communication channel can also be used to configure the non-volatile NAND memory, which simplifies the whole configuration of the platform through the USB port.

SIE provides an economical alternative (70 USD per node) for implementing evolutionary algorithms; Vasicek et al. [17], Higuchi et al. [7] and Sekanina[24] use architectures based on Xilinx Virtex 2 Pro family for implementing similar applications, closed to 1000 USD y 3000 USD per node. By separating the FPGA and the processor and providing a communication channel between them, SIE can use cheaper devices (4 USD for the processor, 10 USD for the FPGA) reducing significantly the cluster's costs. On the other hand, selected GPU for the present work is about 200 USD.

## 4. GP IMPLEMENTATION

The GP was implemented in two stages: the first one is about software development and its parallelization on HPC and GPUs, and the second one refers to a hardware implementation to speed it up on the SIE platform.

### 4.1 Parallel software design.

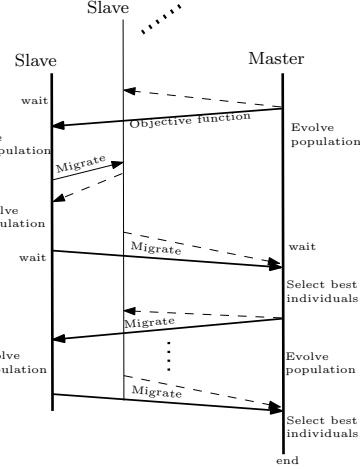
#### 4.1.1 HPC

Natural evolution works with a whole population not with a single individual (except for selection and reproduction) some operations can be done separately, therefore almost all operations in a GP are implicitly parallel. Using the island approach, the population is divided into sub-populations that will evolve in each processor of the cluster or parallel architecture. Figure 3 shows the GP communications scheme in the parallel system. When the system starts, each processor receives the objective function (i.e. a true table), creates its sub-population and starts the evolution process, consisting of fitness evaluation, selection, crossover, mutation and reproduction. Once the system reaches a number of generations, some individuals are selected to be transferred from one processor to another one. A master processor is in charge of collecting the in-transfer individuals and moving them to the rest of the nodes (slaves).

Increasing the probability of convergence of the algorithm [2]. The ratio of data exchange (the number of the best individuals to exchange increases the probability of finding

a better solution) and migration frequency are important parameters to improve the performance of the algorithm.

To implement communications in SIE, a custom message passing library was created to program the distributed system. Standard libraries such Open MPI executes on SIE with a very low performance, due they were designed for robust platforms.



**Figure 3: GP communications scheme in parallel architecture.**

#### 4.1.2 GPU.

Two main parts can be identified to implement the system on GPUs. The first part involves the random number generation and the second one the GP as such. The GP requires random numbers for generating an initial population and then to mutate and cross their individuals. Due that a GPU can not generate random numbers by using C classical libraries, it is necessary generate them in a different way. Generate these numbers on CPU and then transport them to the GPU is not viable because it takes a lot of system clocks. To solve this problem a Mersenne-twister algorithm is executed on the GPU before the *kernel - GP* to make a buffer of random numbers on its own global memory.

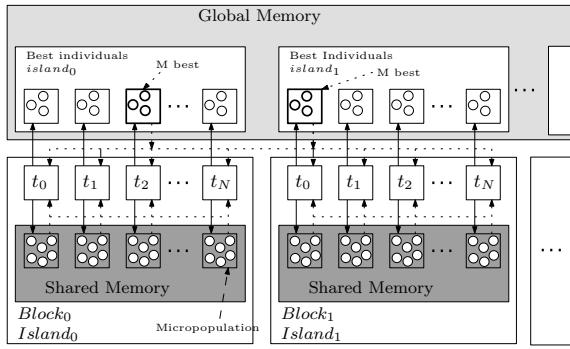
#### Kernel structure.

Figure 4 shows the way the GP has been implemented on the graphics device. A thread  $t$  executes a *kernel - GP* (a common segment of code that process different data in each instance), and generates a  $\mu$ -population, performs the operators of selection, mutation and crossover a number of generations required. After  $P$  generations,  $M$  individuals will be transferred to the global memory and then to the host device (CPU system).  $P$  is known as the frequency of migration and the  $M$  number is called the migration factor.

Also, it can be observed that each thread can cooperate with other threads inside the same block through shared memory, sharing the best individuals and improving the efficiency of the GP.

## 4.2 Hardware design.

In the second stage a profiling of the software implementation with gprof tool, determined that the most costs were

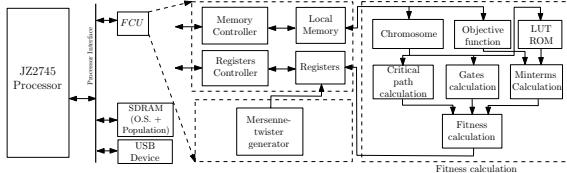


**Figure 4: GPU implementation of the evolvable algorithm.**

the fitness function calculation and the new individual generation (25% and 35% of the execution time, respectively).

Therefore, these two steps have been accelerated with a dedicated hardware in the FPGA. The Fitness Calculation Unit (FCU, see Figure 5) is the hardware element designed in order to accelerate this processes. This coprocessor is connected to the JZ4725 processor through its custom interface (using data, address, control buses).

Inside the FCU, the chromosome passes from the DRAM memory to local memory through the custom interface, and each of its basic cells are converted to an equivalent Virtual Look Up Table (VLUT) with a ROM based translation. Then the number of wrong minterms is calculated using the information from the objective function and a counter as input. Finally, the FCU calculates the final fitness value including the number of gates and the critical path, and will be sent back to the JZ4725 processor. In order to speed up pseudo random number generation, a Mersenne-Twister-based coprocessor was inserted through the same custom interface.



**Figure 5: FCU structure.**

## 5. PERFORMANCE EVALUATION.

To obtain the performance and prove the scalability of the algorithm in SIE, it was compared to two parallel architectures. The first one is a High Performance Computer (HPC) called ALTAMIRA. The HPC setup is made up of 18 eServer BladeCenters with 256 JS20 nodes (512 processors), using a Myrinet network with 1 Gbps bandwidth. The second platform is a GPU architecture based on a NVIDIA GTS450, made by 192 CUDA cores, each one working at 1,56 GHz and a DDR5-1GB global memory.

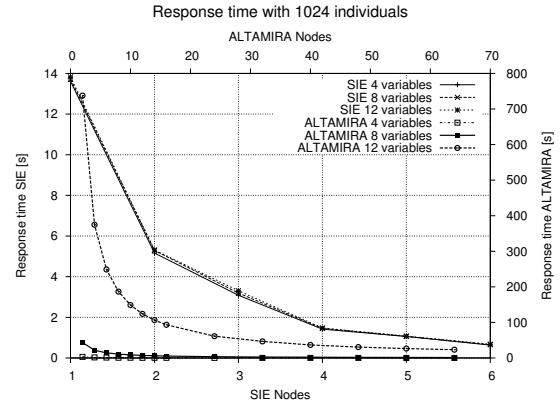
On the other hand, the SIE configuration is made up of 6 JZ4725-FPGA nodes with the architecture described above.

The response time is measured for the parallelized versions of the GP in both ALTAMIRA and SIE. Several scenarios have been tested with different input parameter configurations: 1) number of input variables (4, 8 or 12, corresponding to a comparator problem of 2, 4 and 6 bits); 2) population size (512, 1024 or 2048) and 3) number of nodes running the experiment, ranged from 1 to 6 in SIE, and 2 to 16 or 64 in ALTAMIRA. The first and second parameters determine the size of the problem. The last one gives an idea about the scalability of the system.

### 5.1 Response Time.

Figure 6 shows the response time for both platforms with different numbers of nodes and variables with 1024 individuals during 100 generations. These results show the high performance of SIE cluster for the algorithm. This experiment demonstrates that the response time for SIE does not depend on the size of the problem. In contrast, response time in ALTAMIRA has a high dependency of the size of the problem, because individuals had to be evaluated by software.

Figure 7 shows the response time in both architectures when varying the number of individuals of the population. It is observed that both architectures have a strong dependency of the number of individuals in the algorithm. This is because increasing this number causes an increment of the software computational load for both clusters. Even in this scenario, SIE shows an excellent performance compared to ALTAMIRA.

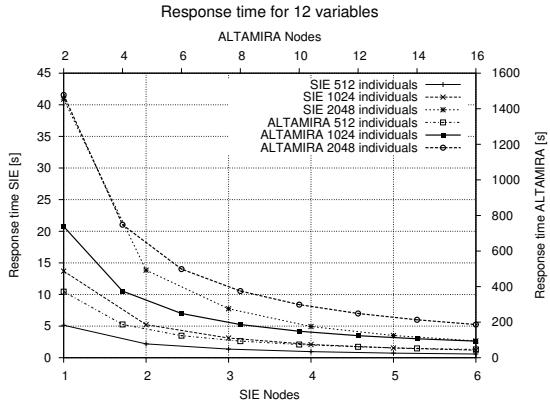


**Figure 6: Response time in SIE and ALTAMIRA for different number of variables.**

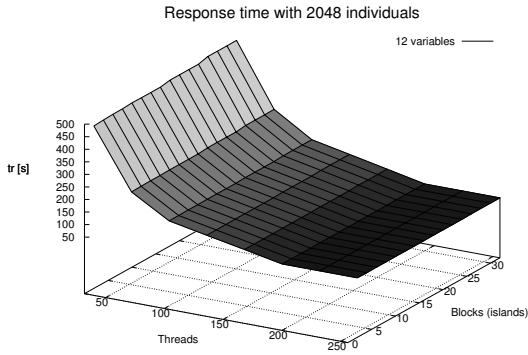
Figures 8 shows the response time of executing the GP in the graphics hardware with problems of 4, 8 and 12 variables are fixed. Varying the number of islands and threads, can be observed that the best scenario is obtained when the number of threads is increased independently the number of islands. This is because when more threads are launched more parallelism is performed in the system, until the maximum of threads permitted by the GPUs is reached.

### 5.2 Speedup.

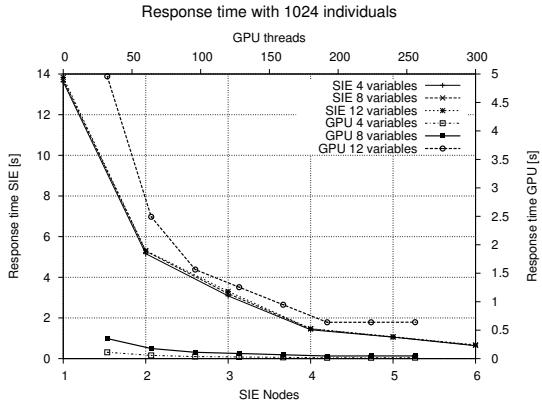
The speedup of the SIE vs ALTAMIRA for different number of variables is presented in figure 10 with the number of variables fixed to 12. The excellent performance of SIE can



**Figure 7: Response time in SIE and ALTAMIRA for different number of individuals.**



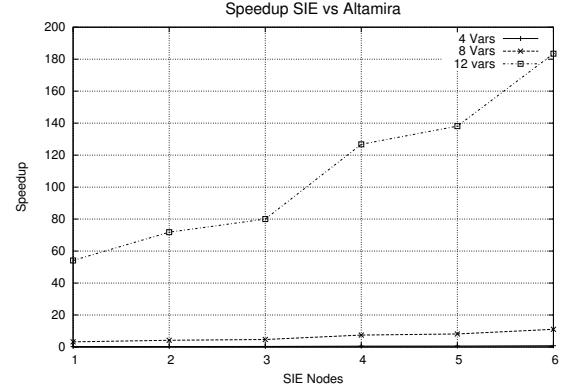
**Figure 8: Response time of the algorithm in NVIDIA GTS450 GPU.**



**Figure 9: Response time in SIE and GPU for different number of individuals.**

be explained because individuals have been directly tested in hardware (FPGA), obtaining a combination of their true ta-

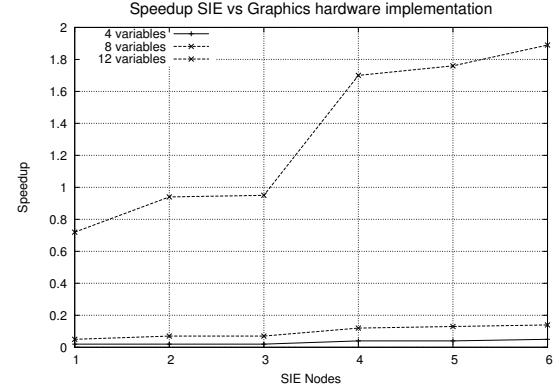
ble on each cycle of the system clock. On the other hand, individuals evaluated in software by ALTAMIRA require a lot of system clocks, causing response times hundreds of times higher than SIE.



**Figure 10: Speedup of SIE vs Altamira comparing 1 SIE node = 2 Altamira nodes.**

On the other hand, figure 11 shows the speedup number when SIE and GPU are compared. It can be observed that SIE performance is higher than GPU only when 12-variables problems are evolved. This can be explained because the whole population have been tested in hardware, obtaining a combination of their output on each cycle of the system clock. But, when an individual is tested in software, each combination requires a set of instructions, that requires a lot of cycles of the system clock.

The obtained speedup number is significant (about 50 and 180) when SIE is compared against ALTAMIRA. In [22, 6] the obtained speedup is about 30-40 when 1 accelerator processor was used and compared against a single PC.



**Figure 11: Speedup of SIE vs GPU comparing 1 SIE node = 32 threads.**

## 6. CONCLUSIONS AND FUTURE WORK.

This paper presented a novel way to evaluate individuals in an intrinsic evolvable algorithm on an open embedded

platform called SIE, and results were compared to an HPC called ALTAMIRA and a high performance NVIDIA GPU. To accelerate the evolution process, a coprocessor was implemented to calculate the fitness function and to generate random numbers, improving the performance for problems with more than 6 bits.

Results showed a speedup of 2 when SIE was compared to GPU, even when 192 processing cores were used in the last one. When compared with ALTAMIRA, SIE showed a speedup up to 180, due the high computational load in software when 16 nodes were used in ALTAMIRA. Significant speedup numbers were obtained due individuals have been directly tested in hardware, getting a combination of their true table on each cycle of the system clock. In the other hand, when an individual is tested in software, each combination requires a set of instructions, that requires a lot of cycles of the system clock.

Tests proved that the algorithm is more effective for 4-bit and 8-bit problems. 12-bit problems in SIE had excellent performance, but because the search space is too long, converging to a suitable solution was difficult for the algorithm. This problem could be solved as future work with some improvements in terms of multiple FCUs inside an FPGA, more nodes, and other hardware-accelerated genetic operators.

## 7. REFERENCES

- [1] A Aguirre, C Coello, and B Buckles. A genetic programming approach to logic function synthesis by means of multiplexers. *Proc. of the I NASA/DoD Workshop on Evolvable hardware.*, pages 46 – 53, 1999.
- [2] Erick Cantú-Paz. Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of Heuristics*, 7(4):311–334, 2001.
- [3] CAC Coello, RL Zavala, and BM Garcia. Ant colony system for the design of combinational logic circuits. *Lecture Notes in Computer Science*, pages 21–30, 2000.
- [4] D Goldberg and J Holland. Genetic algorithms and machine learning. *Machine Learning*, January 1998.
- [5] F. Rothlauf. *Representations for genetic and evolutionary algorithms*. Springer, January 2006.
- [6] Zbysek Gajda and Lukas Sekanina. An efficient selection strategy for digital circuit evolution. In *Evolvable Systems: From Biology to Hardware*, LNCS 6274, pages 13–24. Springer Verlag, 2010.
- [7] J. Jeon and P. Rhee. An Evolvable Hardware System Under Varying Illumination Environment. *Advances in Natural Computation*.
- [8] J Koza, F Bennett, D Andre, and M Keane. Genetic programming iii: darwinian invention and problem solving. *Evolutionary Computation*, January 1999.
- [9] J Miller and P Thomson. Aspects of digital evolution: Evolvability and architecture. *Lecture Notes in Computer Science*, January 1998.
- [10] J Miller and P Thomson. Aspects of digital evolution: Evolvability and architecture. *Lecture Notes in Computer Science*, January 1998.
- [11] I Kajitani, T Hoshino, M Iwata, and T Higuchi. Variable length chromosome ga for evolvable hardware. *Evolutionary Computation*, pages 443 – 447, Jan 1996.
- [12] JR Koza, MA Keane, MJ Streeter, W Mydlowec, and J Yu. Genetic programming iv: Routine human-competitive machine intelligence. *Kluwer Academic Publishers*, 2005.
- [13] Juan Manuel Moreno, Yann Thoma, and Eduardo Sanchez. Poetic: A prototyping platform for bio-inspired hardware. In *ICES*, pages 177–187, 2005.
- [14] A Nicholson. Evolution and learning for digital circuit design. *Proc. Genetic and Evolutionary Computation Conf*, pages 519 – 524, 2000.
- [15] Q. Yu, C. Chen, and C. Pan. Parallel genetic algorithms on programmable graphics hardware. *Lecture notes in computer scienc*, January 2006.
- [16] R Krohling, Y Zhou, and A Tyrrell. Evolving fpga-based robot controllers using an evolutionary algorithm. *Proc. I Int. Conf. on Articial Immune Syst*, January 2002.
- [17] R. Oreifej, R. Al-haddad, H. Tan, and R. Demara. Layered Approach to Intrinsic Evolvable Hardware Using Direct Bitstream Manipulation of Virtex II PRO Devices. *International Conference on Field Programmable Logic and Applications*, 2007.
- [18] T Higuchi, T Niwa, T Tanaka, H Iba, and H de Garis. Evolving hardware with genetic learning: a rst step towards building a darwin machine. *Proc. of the second Int. Conf. on from animals to animats*, January 1992.
- [19] Andres Upegui and Eduardo Sanchez. Evolving hardware by dynamically reconfiguring xilinx fpgas. In *ICES*, pages 56–65, 2005.
- [20] Andres Upegui and Eduardo Sanchez. Evolving hardware with self-reconfigurable connectivity in xilinx fpgas. In *AHS*, pages 153–162, 2006.
- [21] Z Vascek and L Sekanina. Hardware accelerators for cartesian genetic programming. *Lecture Notes in Computer Science*, pages 230–241, 2008.
- [22] Zdenek Vasicek and Lukas Sekanina. Hardware accelerators for cartesian genetic programming. In *Eleventh European Conference on Genetic Programming*, LNCS 4971, pages 230–241. Springer Verlag, 2008.
- [23] H Xu, Y Ding, and Z Hu. Adaptive immune genetic algorithm for logic circuit design. *Proc. of the first ACM/SIGEVO*, pages 639–644, 2009.
- [24] Z. Vasicek and L. Sekanina. An evolvable hardware system in Xilinx Virtex II Pro FPGA. *Int. J. Innovative Computing and Applications*, 2007.

**Low Cost Platform for Evolvable-Based Boolean Synthesis** Publicado en el 2nd IEEE *Latin American Symposium on Circuits and Systems* LASCAS 2011.

# Low Cost Platform for Evolvable-Based Boolean Synthesis

Carlos Iván Camargo

Facultad de Ingeniería

Universidad Nacional de Colombia - Bogotá.

cicamargoba@unal.edu.co

César Pedraza Bonilla

Facultad de Ingeniería de Telecomunicaciones.

Universidad Santo Tomás - Bogotá.

cesarpedraza@usantotomas.edu.co

**Abstract**—Evolutionary algorithms are another option to the combinational synthesis because they allow to create hardware structures that can not be obtained with other techniques. This paper shows a parallel genetic programming (PGP) boolean synthesis implementation based on a low cost cluster of a embedded platform called SIE, based on a 32-bit processor and Spartan-3 FPGAs. Some task of the PGP has been accelerated in hardware and results has been compared with an HPC implementation, resulting speed-up values up to 40 approximately.

**Index Terms**—Embedded systems, Evolutionary algorithms, boolean synthesis, cluster.

## I. INTRODUCTION.

One of the main aims in combinational synthesis consists of finding compact boolean expressions in the sum of products (SOP) form with the smallest possible number of variables and terms. Boolean algebra offers a way to find compact expressions, but this process depends on the designer's experience, resulting in non-optimal or inadequate expressions. There are other techniques available for combinational synthesis such as the Karnaugh maps, the Quine-McCluskey algorithm, the Reed-Muller algorithm, etc. In general terms, these algorithms have disadvantages like exponential complexity, lack of restrictions management, and multiple solutions. As an alternative to the traditional design of combinational circuits, some authors have proposed bio-inspired techniques based on simple genetic algorithms (SGAs), variable-length genetic algorithms (VGAs), tree-based genetic programming (GP), simulated annealing, ant colony algorithms etc. These strategies allow to create combinational blocks that cannot be obtained with the traditional methods, and to add some restrictions to the design such as delay, area, etc. These designs have a very low limited number of variables [1] and they are mainly oriented to obtain a few basic structures.

In order to use parallel genetic programming (PGP) we have developed an FPGA cluster-based architecture to solve combinational synthesis problem on-chip. The fitness coprocessor unit (FCU) on each FPGA helps to accelerate the convergence of the algorithm, as well as provides an appropriate support for 12 variables synthesis problems. The success of the system is mainly due to the capability of evaluating a chromosome in the FPGA through a virtual LUT-oriented architecture without using high-latency partial reconfiguration techniques, and determining the fitness value for an individual faster than

other references.

## II. GENETIC PROGRAMMING TO COMBINATIONAL SYNTHESIS

This section describes some of the most important aspects of the evolutionary algorithm for the combinational synthesis problem, such as chromosome representation, the fitness problem and the genetic operators: Chromosome representation. The codification is the way a logic circuit is represented using a bit array in order to be managed in the evolution process [2]. This representation must be able to represent all the different solutions of the problem, also the crossing and muting operators should not generate unreal individuals, and it must cover all the solution space so the search is really random. There are different ways of representing combinational hardware for a genetic algorithm: tree-based representation in 2-D, PLD-like structures and cartesian representation are some of them [3] [4] [5]

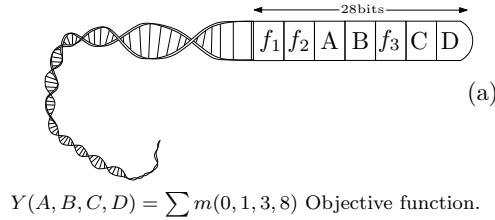
The 2-D tree representation is appropriated for implementing parallel systems because allows splitting the chromosomes for balancing the computational load [6]. Figure 1 shows the selected cell-based structure representation. Each cell has 3 functions  $f$  and 4 input variables  $v$  coded in binary. This representation allows adding more cells to represent larger circuits. It must be mentioned that the chromosome length has to be variable because the length of the solution to the synthesis problem is a priori unknown.

### A. Fitness function.

Finding the appropriate fitness function is important because it is responsible of quantifying the way a chromosome or individual meets or not the requirements.

$$fitness = \omega_1 \cdot [\sum_{j=1}^m \sum_{i=1}^n Y(j, i) - X(j, i)] + \omega_2 \cdot P(x) + \omega_3 \cdot L(x) \quad (1)$$

In equation 1 the fitness function for our GA is shown. Constants  $\omega_1$ ,  $\omega_2$  and  $\omega_3$  are used for establishing the weights of each of the parameters that will determine the fitness function. The double-summation term calculates the number of coincidences of the individual  $X$  for all the possible combinations at the output with the target functions  $Y$ . The  $P(X)$



$$X(A, B, C, D) = \sum m(3, 12, 15) \text{ Individual}$$

Errors = 5

Number of levels = 2

Number of gates = 3

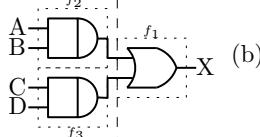


Fig. 1. Cell-based structure representation.

function is used for calculating the number of logic gates of a chromosome taking into account some of the *introns* or segments of the genotype string that will not have any associated function and that do not contribute to the result of the logic circuit they represent. The function  $L(X)$  is used for determining the number of levels the circuit has, or in other words the number of gates that the critical path crosses. The  $m$  constant means the number of outputs in the circuit and  $n$  the number of possible combinations of inputs in the circuit.

#### B. Genetic operators.

The selection operator is responsible of identifying the best individuals of the population taking into account the exploitation and the exploration [6]. The first one allows the individuals with better fitness to survive and reproduce more often, and the second one means searching in more areas and making possible to find better results. In the other hand, the mutation operator modifies the chromosome randomly in order to increase the search space. It can change: 1) an operator or variable and 2) a segment in the chromosome. Both are executed randomly and with a certain probability. A variable mutating probability during the execution of the algorithm (evolvable mutation) [7] is more effective for evolvable systems. Finally, the crossing operator combines two selected individuals for obtaining two additional individuals to add to the population. A crossing system with one or two crossing points randomly selected has been implemented because it is more efficient for evolvable systems [8].

### III. PLATAFORMA HARDWARE COPYLEFT SIE

El proyecto hardware copyleft SIE [9] permite la creación de aplicaciones comerciales bajo la licencia Creative Commons BY - SA [10] la que permite la distribución y modificación del diseño (incluso para aplicaciones comerciales), con el único requisito de que los productos derivados deben tener la misma licencia y deben dar crédito al autor del trabajo original. Lo que constituye la base de los productos *hardware copyleft*.

#### A. Hardware copyleft

Al ser inspirado en el movimiento FOSS, los dispositivos *hardware copyleft* comparten la misma filosofía [11], y son su complemento perfecto. Los requisitos para que un dispositivo HW sea reproducible y modificable son: Disponibilidad de los esquemáticos y los archivos de la placa de circuito impreso en un formato que permita el uso de herramientas abiertas; la cadena de herramientas de compilación y depuración para desarrollo de aplicaciones; el código fuente de: el programa que inicializa la plataforma, la herramienta que carga dicho programa en la memoria no volátil, el sistema de archivos y aplicaciones; documentación completa que indique como fué diseñada, construida, como utilizarla, como desarrollar aplicaciones y tutoriales que expliquen el funcionamiento de los diferentes componentes.<sup>1</sup>. Adicionalmente, se debe contar con la posibilidad de fabricación y montaje, lo que constituye la principal diferencia entre el software y el hardware libre. Esto contrasta fuertemente con el movimiento de software libre, en donde no se requiere inversión de capital para modificar un proyecto existente. Por esta razón, pueden existir varios niveles de libertad, un proyecto que utilice componentes costosos y de difícil consecución limitará su alcance a un sector determinado.

#### B. Arquitectura de la plataforma SIE

La plataforma SIE está compuesta por (Figura 2) un System on Chip MIPS (Ingenic JZ4725) de 400 MHz con periféricos que permiten controlar: una memoria NAND de 2GB para almacenamiento de datos y programas, una memoria SDRAM de 32 MB un canal de comunicación serial (UART), memorias micro-SD, un puerto I2C, un LCD a color de 3 pulgadas, 2 entradas y salidas de audio stereo, 2 entradas analógicas; una FPGA XC3S500E de Xilinx que proporciona 25 señales de entrada/salida digitales de propósito general (GPIOs) y controla un conversor analógico digital de 8 canales. Existen dos canales de comunicación entre la FPGA y el procesador: uno para controlar el puerto JTAG, lo que permite la configuración de la FPGA y ejecución de pruebas a los circuitos implementados en ella; y otro que proporciona el bus de datos, dirección y control para comunicarse con las tareas HW o periféricos implementados en la FPGA.

SIE proporciona un canal de comunicación y alimentación a través del puerto USB, y es configurado para ser utilizado como una interfaz de red, permitiendo la transferencia de archivos y ejecución de una consola remota utilizando el protocolo ssh; este canal de comunicación también se utiliza para programar la memoria NAND no volátil, por lo que para realizar la programación completa de los componentes de la plataforma solo es necesario un cable USB.

### IV. EA IMPLEMENTATION.

The algorithm was implemented in two stages: the first one is about software development and the second one refers to a hardware implementation to speed it up on SIE platform.

<sup>1</sup>Todo esto se encuentra disponible en: <http://projects.qi-hardware.com/index.php/p/nn-usb-fpga/>

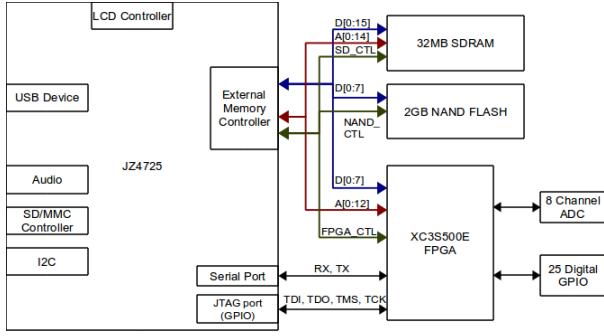


Fig. 2. Estructura de la plataforma de desarrollo SIE

#### A. Software design.

Natural evolution works with a whole population not with a single individual (except for selection and reproduction) some operations can be done separately, therefore almost all operations in a GP are implicitly parallel. Using the island approach, the population is divided into sub-populations that will evolve in each processor of the cluster or parallel architecture. When the system starts, each processor creates its sub-population and starts the evolution process, made up of fitness evaluation, selection, crossing, mutation and reproduction. Once the system reaches a number of generations, some individuals are selected to be transfer from one processor to another. A master processor is in charge of collecting the in-transfer individuals and to move them to the rest of the nodes (slaves), increasing the probability of convergence of the algorithm. The ratio of data exchange (the number of the best individuals to exchange increases the probability of finding a better solution) and migration frequency are important parameters to improve the performance of the algorithm. To implement communications in SIE, a custom message passing library was created program the distributed system.

#### B. Hardware design.

In the second stage a profiling of the software implementation determined that the most consuming part were the fitness function calculation and the new individual generation (25% and 35% of the execution time, respectively). Therefore, these two steps have been accelerated with a coprocessor in the FPGA. The Fitness Calculation Unit (FCU, see Figure 3) is the hardware element designed in order to accelerate this processes. This coprocessor is connected to the JZ4725 processor through its custom interface.

The chromosome pass from the DRAM memory to the FCU through the custom interface, and each of its cells are converted to a equivalent Look Up Table in ROM based translation. Then the number of wrong minterms is calculated using the information from the objective function and a counter as inputs. Finally the FCU calculates the final fitness value including the number of gates and the critical path, and will be send back to the JZ4735 processor. In order to speed up pseudo random number generation, a Mersenne-Twister-based was inserted through the same custom interface.

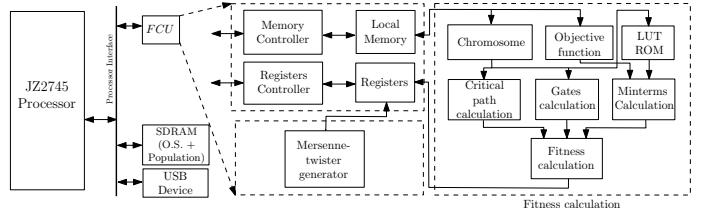


Fig. 3. FCU structure.

## V. EVALUATION.

To obtain the performance and prove the scalability of the algorithm in SIE, it was compared to a High Performance cluster called ALTAMIRA. The cluster setup is made up of 18 eServer BladeCenter, with 256 JS20 nodes (512 processors), using a Myrinet network with 1 Gbps bandwidth. On the other hand, the SIE configuration is made up of 6 JZ4725-FPGA nodes with the architecture described before. The tests response time measurements for the parallelized versions of the GP both in ALTAMIRA and SIE. Several scenarios have been checked with different input parameter configurations: 1) number of input variables (4, 8 or 12, corresponding to a comparator problem of 2, 4 and 6 bits); 2) population size (512, 1024 or 2048) and 3) number of nodes running the experiment, ranged from 1 to 6 in SIE, and 2 to 16 or 64 in ALTAMIRA. The first and second parameter determine the size of the problem. The last one, gives an idea about the scalability of the system.

#### A. Response Time.

Figures 4 shows the response time for both platforms with different number of nodes and variables with 1024 individuals during 100 generations. These results the high performance of SIE cluster for the algorithm. This experiment demonstrate that response time for SIE do not depends of the size of the problem. Contrarily, response time in ALTAMIRA has a high dependency of the size of the problem, because individuals had to be evaluated in software.

Figure 5 shows the response time in both architectures when varying the number of individuals of the population. It is observed that both architectures has a strong dependency of the number of individuals in the algorithm. This is because increasing this number causes an increment of the software computational load for both clusters. Even in this scenario, SIE shows an excellent performance compared to ALTAMIRA.

#### B. Speedup.

The speedup of the SIE vs ALTAMIRA for different number of variables is presented in figure 6 with a number of variables fixed to 12. The excellent performance of SIE can be explained because individuals have been directly tested in hardware (FPGA), obtaining a combination of their true table on each cycle of the system clock. In the other hand, individuals evaluated in software by ALTAMIRA requires a lot of system clocks, causing response times tens of times higher than SIE.

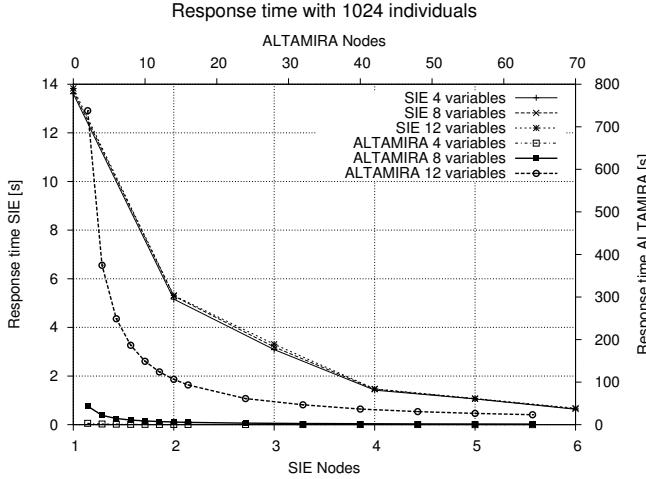


Fig. 4. Response time in SIE and ALTAMIRA for different number of variables.

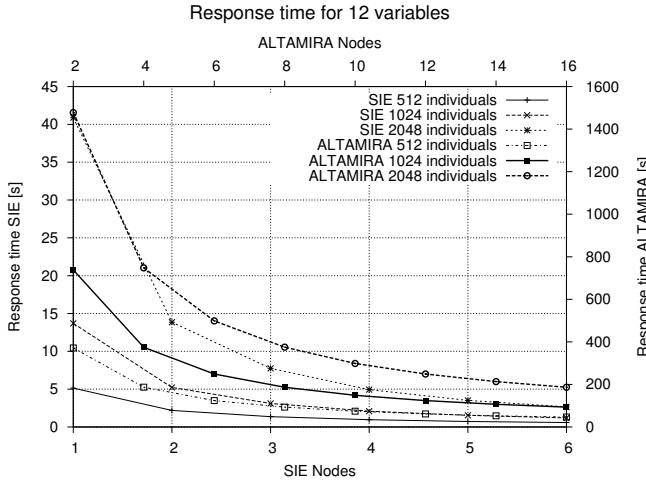


Fig. 5. Response time in SIE and ALTAMIRA for different number of individuals.

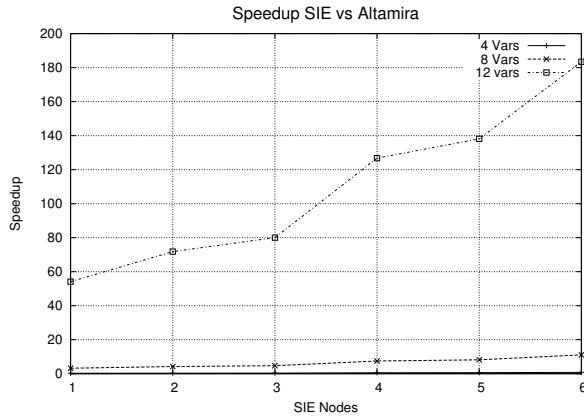


Fig. 6. Speedup of SIE vs Altamira comparing 1 SIE node = 2 Altamira nodes.

### C. Resulting circuits.

Figure 7 shows an example of the resulting circuits for the 2-bit comparator problem. This resulting structures use to be novel, compared to the results obtained with other techniques as Karnaugh maps, and QuineMcCluskey.

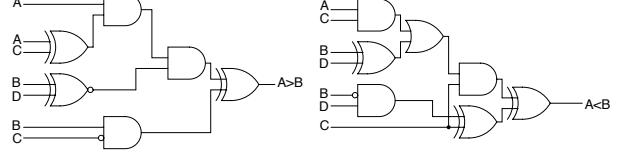


Fig. 7. Example of resulting circuits for the 2-bit comparator problem (A=MSB number1, C=MSB number2).

## VI. CONCLUSIONS AND FUTURE WORK.

This paper showed a novel way to evaluate individuals in an evolvable algorithm on an open embedded platform called SIE, and results where compared to an HPC called ALTAMIRA. To accelerate the evolution process, a coprocessor was implemented to calculate the fitness function and to generate random numbers, improving the performance for problems with more than 6 bits. Test proved that the algorithm is more effective for 4-bit and 8-bit problems. 12-bit problems in SIE had an excellent performance, but because the search space is too long, converging to a suitable solution was difficult for the algorithm. This problem could be solved as future work with some improvements in terms of multiple FCUs inside an FPGA, more nodes, and other hardware-accelerated genetic operators.

## REFERENCES

- [1] D Goldberg and J Holland. Genetic algorithms and machine learning. *Machine Learning*, January 1998.
- [2] F Rothlauf. *Representations for genetic and evolutionary algorithms*. Springer, January 2006.
- [3] J Koza, F Bennett, D Andre, and M Keane. Genetic programming iii: darwinian invention and problem solving. *Evolutionary Computation*, January 1999.
- [4] T Higuchi, T Niwa, T Tanaka, H Iba, and H de Garis. Evolving hardware with genetic learning: a first step towards building a darwin machine. *Proc. of the second Int. Conf. on from animals to animals*, January 1992.
- [5] J Miller and P Thomson. Aspects of digital evolution: Evolvability and architecture. *Lecture Notes in Computer Science*, January 1998.
- [6] Q Yu, C Chen, and C Pan. Parallel genetic algorithms on programmable graphics hardware. *Lecture notes in computer science*, January 2006.
- [7] R Krohling, Y Zhou, and A Tyrrell. Evolving fpga-based robot controllers using an evolutionary algorithm. *Proc. 1 Int. Conf. on Artificial Immune Syst*, January 2002.
- [8] J Miller and P Thomson. Aspects of digital evolution: Evolvability and architecture. *Lecture Notes in Computer Science*, January 1998.
- [9] C. Camargo, W. Spraul, and A. Wang. Proyecto SAKC. URL:<http://en.qi-hardware.com/wiki/SAKC>.
- [10] Creative Commons. Licencias Creative Commons. URL:<http://creativecommons.org/licenses/>, 2004.
- [11] R. Stallman. Philosophy of the GNU project. URL:<http://www.gnu.org/philosophy/>, 2007.

### **A.1.2. Aplicaciones en Robótica**

**CONTROL DE SISTEMAS PARALELOS INSPIRADO EN LA NATURALEZA** 3th  
IEEE Colombian Workshop on Circuits and Systems.

# CONTROL DE SISTEMAS PARALELOS INSPIRADO EN LA NATURALEZA

Carlos Iván Camargo – Universidad Nacional de Colombia  
cicamargoba@unal.edu.co

**Abstract**—En la actualidad los dispositivos electrónicos hacen parte de nuestras actividades, a diario estamos en contacto con decenas de dispositivos digitales, ellos se encuentran integrados a aparatos de uso común como electrodomésticos, sistemas de comunicaciones, sistemas de entretenimiento, etc. Esta “invasión” digital va en aumento gracias al gran desarrollo de la industria de los semiconductores y a la disponibilidad de herramientas de programación. En un futuro no muy lejano, millones de dispositivos serán integrados en calles, puentes, y en nuestras propias casas, estos dispositivos formarán redes intercomunicadas que permitirán supervisar y controlar variables interesantes como tráfico, variables ambientales, identificación de personas. Sin embargo, es necesario dar respuestas a nuevas preguntas y afrontar nuevos retos que se presentan en el momento de diseñar este tipo de sistemas. La naturaleza es una fuente de inspiración para resolver este tipo de problemas ya que podemos encontrar ejemplos de sistemas con millones de componentes que interactúan para cumplir tareas que benefician al sistema como un todo. Un buen ejemplo de sistemas masivamente paralelos lo encontramos en los insectos sociales, vemos como millones de hormigas o termitas trabajan coordinadamente para suplir las necesidades de sus colonias, este tipo de sociedades no poseen elementos de coordinación, ni existen individuos especializados para dicho fin, el comportamiento emerge a partir de interacciones locales. El sistema inmune de los mamíferos es otro buen ejemplo de sistema masivamente distribuido en el que sus componentes actúan de forma coordinada para luchar contra agentes externos. En este trabajo se presenta al sistema inmune artificial como elemento de control de un sistema distribuido, cuyos individuos son robots móviles.

**Index Terms**—Sistemas Bio-Inspirados, Sistemas Multi-Robot, Sistemas Embebidos.

## I. INTRODUCCIÓN

En la actualidad existe un número creciente de investigadores que ven a la naturaleza como fuente de inspiración para realizar sus diseños (por ejemplo: aplicaciones inspiradas en, los sistemas inmunes artificiales, algoritmos genéticos, autómatas celulares). La naturaleza ha creado sistemas muy complejos que están formados por millones de elementos, los cuales ejecutan sus funciones en paralelo, cada elemento realiza una determinada tarea para lograr un objetivo común. Los seres vivos cuentan con mecanismos que supervisan a diferentes niveles, el estado y funcionamiento de sus componentes; si se detecta una anomalía, estos mecanismos son capaces de reparar componentes defectuosos (regeneración) o de iniciar procesos de eliminación de elementos nocivos [1]; ante la presencia de un organismo externo el Sistema Inmune inicia un proceso de eliminación, el cual consiste en la especialización de ciertas células para la detección del agente nocivo, este proceso se realiza de forma paralela

y distribuida, lo cual mejora el tiempo de respuesta. Estas células especializadas finalmente generan mecanismos para la eliminación de dichos elementos. Los diseñadores de sistemas computacionales encuentran especialmente atractivas las propiedades de auto-replicación, auto-reparación y asociación; en la actualidad existen varias líneas de investigación cuyo objetivo es crear metodologías de diseño que buscan *trasladar* estas propiedades al campo de los sistemas digitales y computacionales [2], [3], [4], [5], [6].

Por otro lado, es muy interesante la forma en que comportamientos complejos emergen de la interacción local de millones de elementos sencillos. Ejemplos de este tipo se observan a diario en la naturaleza: los cardúmenes de peces o las bandadas de aves. Existen muchas personas que creen que el comportamiento de las aves o los peces obedecen al seguimiento de líderes, esto debido a la forma de pensar centralizada de los seres humanos, sin embargo, el comportamiento global depende de la reacción individual a acciones de sus vecinos inmediatos. Otro ejemplo se puede observar en el comportamiento de las colonias de insectos, cada individuo realiza una función específica para contribuir a un objetivo general: la supervivencia y la conservación de la especie; a partir de interacciones locales estas colonias crean estructuras complejas, las cuales no podrían ser realizadas por un solo individuo. La respuesta a la pregunta ¿Cómo emerge un determinado comportamiento global a partir de interacciones locales? ha sido la inspiración de un gran número de investigaciones [7], [8], [9], [10]. Sin embargo, esta pregunta no ha sido completamente respondida y existe un largo camino por recorrer hasta que seamos capaces de formular reglas locales que permitan obtener de forma colectiva la solución a cualquier problema complejo.

Existe un considerable número de trabajos en los que se explotan las características distribuidas de los sistemas inmunes artificiales (AIS) para controlar un gran número de robots en tareas de búsqueda y rescate, y detección de minas [11], [12], [7], [13]; otros investigadores comienzan a estudiar la forma de explotar las capacidades de los AIS en el entorno de computación ubicua [14], [15], por otro lado, la utilización de los AIS como medio para obtener tolerancia a fallos en sistemas digitales, ya está siendo estudiada [16], [17], [18]. Esta gama de aplicaciones de los AIS hace pensar en la posibilidad de su utilización para formular un nuevo paradigma computacional o para complementar los ya existentes, por ejemplo, adicionando la capacidad de detección y remoción de fallas. Sin embargo, existen muchas consideraciones que deben tomarse en cuenta en el desarrollo de estos sistemas

[19] antes de obtener un progreso significante en la utilización de los AIS como medio para obtener la tolerancia a fallos.

Ejemplos como los anteriores evidencian el creciente interés en el estudio y desarrollo de sistemas que utilizan metáforas Biológicas en su funcionamiento; tal vez una de las características más importantes de estos sistemas bioinspirados (desde el punto de vista electrónico) es su capacidad intrínseca de adaptación a entornos cambiantes, lo cual les permite exhibir propiedades de tolerancia a fallos. El campo de aplicación de estos sistemas no se limita solo al campo de la electrónica, sino que se extiende a un gran número de actividades humana, como por ejemplo: actividades económicas, políticas comerciales. Por otro lado, proporcionan una fuente de inspiración y una motivación para trabajar con sistemas distribuidos masivos. Sin embargo, para poder ser capaces de crear dispositivos que se comporten de forma similar, primero debemos cambiar la forma de ver el mundo y cambiar nuestra tendencia al centralismo y entrar a una era descentralizada. Sin embargo, el interés en la descentralización no es nuevo; las ideas de la descentralización se expanden hoy más rápidamente y penetran más profundamente en diferentes dominios de las actividades humanas: organizaciones, tecnologías, modelos científicos y teorías del conocimiento. En el caso de las tecnologías actuales como los MEMs (Micro-Electro Mechanical devices) es posible la construcción de millones de dispositivos computacionales equipados con sensores y actuadores y su integración en materiales, estructuras y entornos [20], [21]. Se estima que en el futuro aumenten las aplicaciones de estas tecnologías, por lo que su programación será un reto constante [22]; al igual que entender cómo diseñar y programar grandes sistemas descentralizados.

Sin embargo, para estar en condiciones de crear sistemas digitales inspirados en las *sociedades naturales* es necesario dar respuestas a nuevas preguntas y afrontar nuevos retos que se presentan en el momento de diseñar este tipo de sistemas. A continuación se enumeran los principales problemas a los que nos debemos enfrentar:

- **Diseño:** ¿Qué técnicas y métodos permiten el diseño y control de estos sistemas?
- **Organización:** ¿Cómo pueden reorganizarse y coordinar su comportamiento?
- **Uso:** ¿Qué usos se darán a estos sistemas?
- **Adaptación:** ¿Cómo conforman funciones colectivas y como las memorizan?
- **Desarrollo:** ¿Qué tipos de protocolos de interacción y comunicación, que modos de organización podrán crear sistemas seguros, confiables y flexibles?
- **Validación:** ¿Cómo evaluar la validez de estos sistemas?

Para dar respuesta a estas preguntas realizaremos un estudio basado en *Colonias de Robots*, las cuales constituyen un entorno de trabajo adecuado para explorar las diferentes variables que intervienen en el problema; en el presente trabajo no nos limitaremos solo a obtener resultados experimentales sino que estos resultados serán llevados al mundo físico con robots construidos por nosotros, esto se hace con el fin de adoptar nuevas metodologías de diseño de sistemas digitales y evaluar la validez de las soluciones propuestas.

La robótica Móvil representa un campo de investigación que crece rápidamente. En la actualidad existe un gran número

de grupos de investigación que trabajan en el desarrollo de técnicas de auto-organización para sistemas Multi-Robot, entre los que se encuentran: El proyecto Swarm-bots [23], [24], Interaction Labs (USC) [25] [26], Autonomous System Lab (EPFL), MIT Computer Science and Artificial Intelligence Laboratory [27]. De las experiencias obtenidas de estos proyectos, se deduce, que para desarrollar algoritmos de inteligencia artificial, es necesario contar con plataformas Hardware y Software que permitan validar los algoritmos y modelos computacionales propuestos. La característica común de los proyectos ya mencionados es la implementación física de los algoritmos sobre robots reales, lo cual separa estos trabajos de investigaciones similares en el área de los Sistemas Multi-Agente, los cuales son desarrollados en plataformas Software. En la primera parte de este artículo se describirá la plataforma Hardware utilizada para realizar las pruebas, en la segunda se explicarán las bases del modelo utilizado.

## II. ECBOT: PLATAFORMA ABIERTA PARA ROBÓTICA BASADA EN LINUX

ECBOT está formado por la unión de dos grandes componentes: El Componente Hardware y el Componente Software, el primero constituye el dispositivo físico mediante el cual se interactúa con el entorno y sobre el cual el componente Software implementa un determinado algoritmo.

### A. Arquitectura Software

La columna vertebral del componente Software es el proyecto Player/Stage, el cual, es el resultado de un proyecto de investigación del Grupo de Investigación en Robótica de la University of Southern California. Este proyecto esta dividido en tres partes:

- 1) **Player** Es un servidor que proporciona una interfaz flexible a una gran variedad de sensores y actuadores. Como puede verse en la Figura 1 utiliza un modelo cliente/servidor basado en sockets TCP, su principal característica es que permite el manejo de los sensores y actuadores a través de una interfaz de programación de alto nivel, lo cual permite que los programas de control del robot sean escritos en cualquier lenguaje y puedan ser ejecutados en cualquier plataforma<sup>1</sup> (cliente) que posea una conexión de red con el robot (servidor). Además, Player soporta múltiples conexiones concurrentes de clientes, lo cual es muy útil en estudios de control descentralizado.
- 2) **Stage** Es un simulador escalable; que simula robots que se mueven y realizan operaciones de sensado en un entorno de dos dimensiones, estos robots son controlados por Player. Stage proporciona robots virtuales los cuales interactúan con dispositivos simulados. En la Figura 2 se puede observar una simulación multi-robot utilizando las librerías de Stage; una de las características más atractivas del proyecto Player-Stage es que permite la creación de sensores y actuadores que simulan el comportamiento de los dispositivos reales.

<sup>1</sup>También es posible que el cliente y el servidor se ejecuten en el mismo robot.

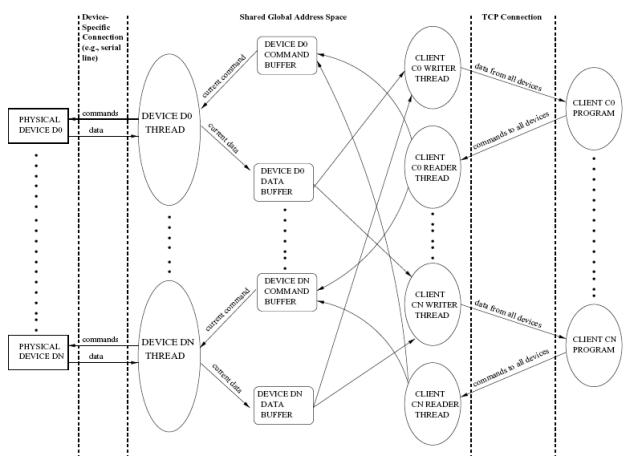


Fig. 1. Arquitectura de Player. [28]

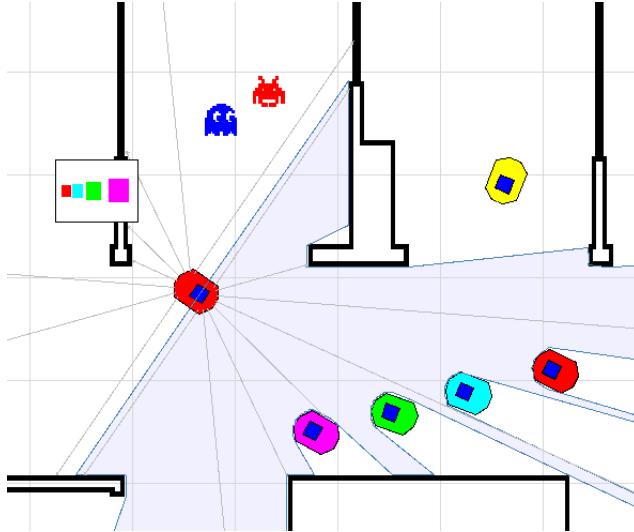


Fig. 2. Captura del simulador Stage.

1) *Arquitectura Hardware*: ECBOT [29] [30] [31] reúne la mayoría de las características de las plataformas comerciales disponibles<sup>2</sup>. Esta plataforma permite la ejecución de aplicaciones Linux [32] [33] y permite realizar de forma fácil la implementación de algoritmos de control. Gracias a su capacidad de comunicación inalámbrica no es necesario detener el robot y llevarlo a un sitio donde sea programado, ECBOT permite cambiar su configuración de forma remota. Este robot es el primero en su tipo diseñado totalmente en Colombia, tanto el HW como el SW fueron adaptados para él por ingenieros Colombianos, esto ayuda a disminuir la brecha tecnológica que existe en nuestro país, y permite que en nuestras instituciones académicas se utilice tecnología de punta, eliminando de esta forma tecnologías obsoletas utilizadas hasta el momento. En la figura 3 se muestra el Diagrama de Bloques de ECBOT,

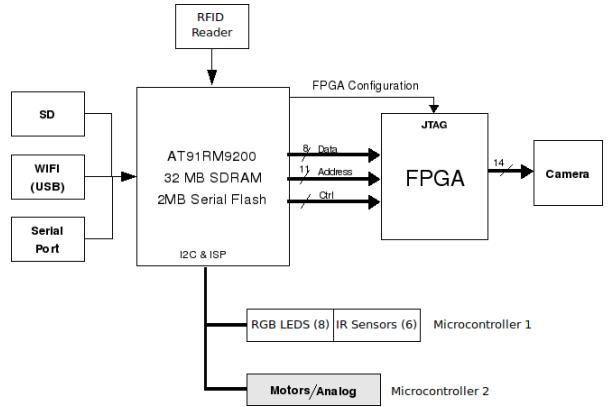


Fig. 3. Diagrama de Bloques del componente Hardware de ECBOT.

### B. Especificaciones de ECBOT

ECBOT fué diseñado pensando en que pueda ser utilizado como una plataforma para el estudio de los sistemas Embebidos, y puede ser utilizado en cualquier campo, es ideal como herramienta didáctica ya que como puede verse en 3 permite implementar tareas HW en una FPGA, lo cual no se enseña en los programas académicos de la Universidades Colombianas. Adicionalmente permite la utilización de Linux como sistema operativo, lo cual facilita el desarrollo de aplicaciones gracias al gran número de librerías y aplicaciones disponibles. El principal aporte de este trabajo es la adopción de estas nuevas tecnologías y metodologías de diseño al entorno académico e industrial Colombiano, proporcionando de esta forma herramientas que permitan a nuestros profesionales competir de forma más eficiente con los productos producidos en el exterior. Por otro lado, este es el primer paso hacia la independencia tecnológica que nos ha convertido en importadores de tecnología, esta plataforma representa un avance de más de 30 años en el diseño de sistemas digitales, y gracias a su carácter libre puede ser utilizado por empresas e instituciones académicas como punto de partida en el estudio de los sistemas Embebidos modernos.

1) *Sensores y Actuadores*: ECBOT está diseñado de tal forma que se le pueda adaptar cualquier sensor que posea la interfaz I2C, y permite controlar hasta 10 servomotores, sin embargo, cuenta con una serie de sensores básicos que le permiten realizar las siguientes tareas:

- Detección de obstáculos: 6 sensores infrarrojos de proximidad.
- Control de posición: Control de 2 motores DC utilizando PWM y BEMF.
- Sensor de Imagen: Permite capturar imágenes en formato subQCIF (129 x 96)
- Leds RGB: Permite cambiar el color del robot.

Cada robot posee un detector de *blobs* de colores, lo cual permite la identificación de ciertos lugares del entorno (Almacenamiento de alimentos, Hogar) y el estado de otros robots.

2) *Comunicaciones*: Cada plataforma robótica puede comunicarse de forma directa con otros robots utilizando la interfaz de red inalámbrica, o de forma indirecta comunicando su estado interno a través de los LEDS RGB, cada color

<sup>2</sup>Roomba: <http://www.irobot.com>, Khepera: <http://www.k-team.com>, Pioneer: <http://www.activrobots.com/>

representa un estado interno, este estado puede ser detectado con el sensor de imagen. En el presente estudio se prefiere esta última forma de comunicación ya que si observamos el comportamiento de los sistemas naturales y biológicos ellos no utilizan comunicación directa. Adicionalmente cada robot cuenta con un sensor RFID que permite de forma indirecta comunicarse con otros robots de la misma forma que lo hacen las feromonas en la naturaleza.

En la Figura 4 se muestra una fotografía de la tarjeta base de la plataforma robótica, en la que podemos observar la localización de los componentes, esta arquitectura está fuertemente influenciada por el robot *e-puck* [34] del EPFL.

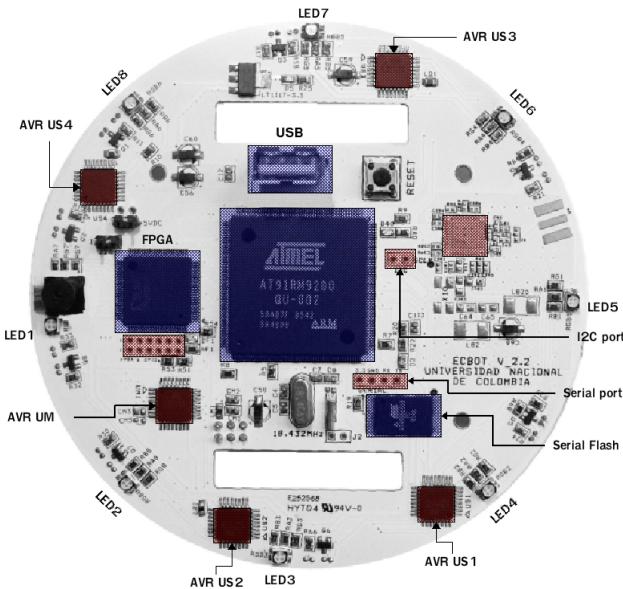


Fig. 4. Lado de Componentes de la tarjeta principal de ECBOT.

### III. ALGORITMO DE CONTROL

El trabajo aquí presentado tiene dos fuentes de inspiración tomadas de la naturaleza: El sistema inmune de los mamíferos y las feromonas utilizadas por las hormigas como medio de comunicación.

#### A. Sistema Inmune Artificial

Especificidad, inducibilidad, diversidad, memoria, distinción entre propio y no propio (non self), autoregulación; son las principales características del sistema inmune humano, y las que permiten que utilizarlo en la solución de problemas en ingeniería.

El marco de control organiza un grupo de agentes en un entorno dinámico, se desarrolla con base en el mecanismo distribuido de la inmunidad biológica. El Sistema Inmune (SI), es un tipo especial de sistema multi-agente donde cada elemento inmune tiene un patrón de comportamiento y funciones específicas para un antígeno particular. El comportamiento de dicho agente, depende del entorno, así como del comportamiento individual; además posee un grupo de capacidades específicas que determinan su inteligencia fundamental. Esta

inteligencia puede ser alcanzada a través de comunicación entre agentes o a través de exploraciones del entorno.

Se adoptó una red de control distribuida basada en el comportamiento, la cual se basa en la transición de estado de comportamiento de los agentes. [35] Propone un nivel de descripción, instanciado en el llamado “comportamiento básico”, bloques constructores para síntesis y análisis de funcionamiento complejo de grupo en un sistema multi-agente. La Biología proporciona evidencia de soporte de unidades de funcionamiento básico en una variedad de niveles. Controlar una pierna de una rana, o un brazo humano, es una tarea compleja, especialmente si se realiza a bajo nivel. Con el fin de reducir la complejidad, la naturaleza impone una abstracción. Musaa - Ivaldi & Giszter [36] muestran que un grupo relativamente pequeño de campos vectoriales básicos, encontrados en la espina de la rana, genera el repertorio completo del comportamiento motriz, aplicando combinaciones apropiadas de los vectores básicos. Tomando esto como idea, se definen funcionamientos como leyes de control que encapsulan grupos de restricciones, de tal forma que se logran metas particulares. Los funcionamientos básicos se definen como un grupo mínimo de tales comportamientos, con propiedades apropiadas, que toman ventaja de la dinámica de el sistema dado, para cumplir de forma efectiva con su repertorio de tareas.

Los comportamientos básicos son una herramienta que pretenden describir, especificar y predecir el funcionamiento del grupo. Si se seleccionan de forma adecuada estos pueden generar comportamientos grupales repetibles y predecibles. La idea de los comportamientos básicos es general: Ellos pretenden ser primitivas para estructurar, sintetizar y analizar el funcionamiento del sistema, así como, bloques constructores para control, planeamiento y aprendizaje. Están relacionados con los atractores dinámicos, estados de equilibrio y otros términos utilizados para describir comportamientos estables, repetibles y primitivos de cualquier sistema [37].

1) *Comportamientos Básicos Para Movimiento en un Plano:* Los comportamientos básicos propuestos en [35] definen grupos de comportamientos sin especificar reglas particulares para implementarlas. El comportamiento de grupo en el dominio espacial puede verse como patrones espacio-temporales de la actividad de los agentes. Ciertas organizaciones de agentes espacialmente fijas son relevantes, así como ciertos patrones espacio-temporales. Las organizaciones de agentes espacialmente fijas corresponden a logros de metas, mientras los patrones espacio-temporales corresponden al mantenimiento de las mismas. La Tabla I muestra una lista de comportamientos que constituyen un grupo básico para un repertorio flexible de interacciones de grupo.

2) *Combinación de Comportamientos Básicos:* Los comportamientos básicos están diseñados para ser un substrato para una variedad de comportamientos de grupo más complejos para un determinado dominio. Generar funcionamientos complejos requiere aplicar algún tipo de operadores de combinación cuyas propiedades sean bien conocidas y produzcan el funcionamiento compuesto deseado. Este es uno de los retos del control *Basado en el Comportamiento*, esto es, coordinar la actividad de multiples comportamientos de entrada para

<b>Safe Wandering</b>	La habilidad de un grupo de agentes para moverse alrededor, mientras evitan colisiones entre ellos y otros obstáculos. La naturaleza homogénea de los agentes puede utilizarse para evitar colisiones entre agentes. Esto es, se pueden divisar dos estrategias diferentes; una para evitar colisiones entre agentes del mismo tipo, y otra para evitar colisiones con todo lo demás.
<b>Following</b>	La habilidad de dos o más agentes para moverse mientras se mantiene uno detrás del otro.
<b>Dispersion</b>	La habilidad de un grupo de agentes para dispersarse sobre un área con el fin de establecer y mantener una distancia mínima predeterminada.
<b>Aggregation</b>	La habilidad de un grupo de agentes de reunirse con el fin de establecer y mantener una distancia máxima predeterminada.
<b>Homing</b>	La habilidad para alcanzar una región o lugar específico.

TABLE I

GRUPO DE COMPORTAMIENTOS BÁSICOS PARA EL DOMINIO ESPACIAL

producir el comportamiento de salida deseado. En la figura 5 se muestran las combinaciones necesarias para obtener los comportamientos *hearding* y *surrounding*

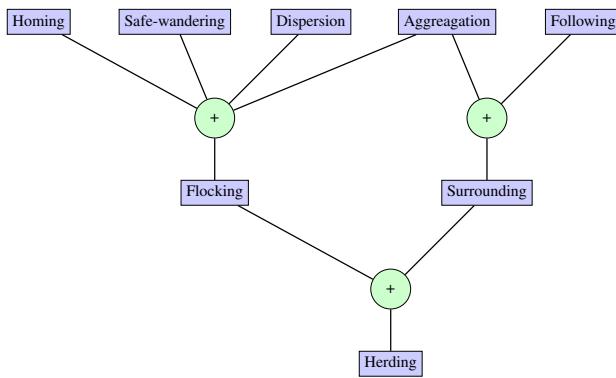


Fig. 5. Ejemplo de combinación de comportamientos básicos

La operación de los agentes no está predeterminada, pero se altera dinámicamente con el fin de adaptarse a un entorno de trabajo dinámico. Este control distribuido y no-determinístico hereda las siguientes funciones del sistema inmune humano:

- 1) **Auto-organización** Los agentes tienen la capacidad única de determinar respuestas para alcanzar objetivos comunes a través de toma de decisiones independientes y utilizando comunicaciones.
- 2) **Reconocimiento de Self/Non-self** Los agentes identifican tareas durante su exploración aleatoria; para evaluar la estimulación de una tarea específica se utiliza la función de afinidad (un índice para identificación non-self). Este reconocimiento depende únicamente de la estimulación proporcionada por la tarea, los agentes no poseen información previa sobre su localización o complejidad. Cada agente posee un set predefinido de capacidades, lo cual los hacen extremadamente flexibles para abordar diferentes tipos de problemas.
- 3) **Adaptabilidad** Los agentes ajustan sus comportamientos al atacar un problema manipulando la mejor respuesta, evaluando la especificidad de sus capacidades.

Se adquieren nuevos conocimientos o funcionalidades cuando se enfrentan a variaciones de tareas y entorno.

- 4) **Robustez** La falla de un agente, mientras ejecuta una operación no paralizará el sistema; ya que el marco de control está encaminado a ser totalmente descentralizado y no se asignan dependencias fijas entre tareas y agentes.
- 5) **Diversidad** El sistema es capaz de aprender de la experiencia, manipulando sus capacidades y realzando su inteligencia fundamental para resolver diferentes problemas.

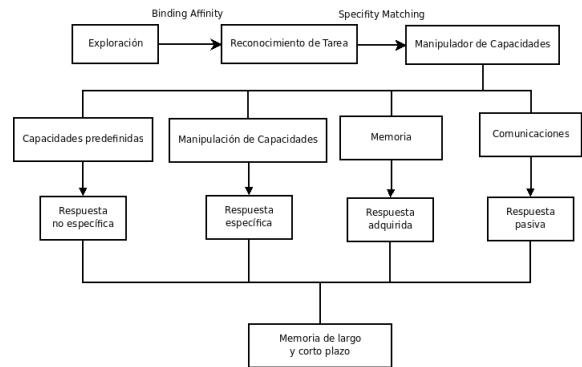


Fig. 6. Flujo de control de un agente del sistema inmune

El marco de control proporciona un grupo de reglas que guían y determinan el comportamiento de un agente en respuesta a un entorno dinámico. A través de la manipulación de las reglas se pueden investigar de forma efectiva eventos desconocidos y variaciones dinámicas del sitio de trabajo. La figura 6 muestra el flujo de control de un agente SI. Primero realizan una exploración del entorno circundante dentro de su rango sensorial ( $SR_S$ ). Basándose en la medición de afinidad, los agentes reconocen y se enfocan en una tarea. Una vez reconocida una tarea, el agente manipula sus capacidades con una función que recibe el nombre de *matching específico* esto permite que el agente realice respuestas apropiadas al abordar diferentes tareas. Se definen cuatro tipos de respuestas inspiradas en el sistema inmune, ellas son: No específicas, específicas, adquiridas y pasivas. La respuesta no específica se encarga de tareas generales, los agentes ejecutan capacidades predefinidas cuando abordan este tipo de tareas; para tareas más complicadas, los agentes requieren la generación de nuevas capacidades específicas a estas tareas, los agentes modifican sus capacidades fundamentales para abordar este tipo de tareas, estas nuevas capacidades son almacenadas en memoria en la forma de respuesta adquirida. La respuesta pasiva se presenta en tareas que requieran cooperación de más de un agente, es este caso, el primer agente que solicita ayuda es el único que puede dar instrucciones a los otros agentes, los cuales exhiben un comportamiento pasivo.

Los cambios de estado son gobernados por un grupo de reglas evento-condición-acción, dichas reglas permiten que los agentes SIA realicen respuestas bajo diferentes situaciones; de acuerdo con estas reglas, los agentes obtienen información a través de comunicación entre ellos, esta información incluye localización de tareas, señales de auxilio, y estados de com-

portamiento de otros agentes; toda esta información permite a los agentes coordinar planes de acción para diferentes tipos de tareas. La Figura 7 muestra el árbol de decisiones que define las acciones de los agentes SIA, este árbol muestra como los agentes se adaptan a cambios en el entorno y se comportan de forma autónoma al abordar las tareas.

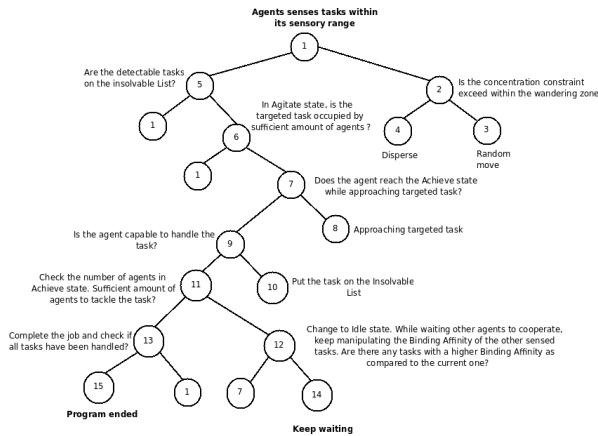


Fig. 7. Árbol de decisiones que define las acciones de un agente SIA [38]

### B. feromonas Artificiales

Uno de los principales problemas al trabajar con sistemas masivamente distribuidos es la comunicación entre los individuos, permitir la comunicación directa entre los miembros del sistema y difundir información del entorno a todos los integrantes hace que el proceso de comunicación requiera mucha capacidad de almacenamiento y procesamiento, lo cual dificulta escalar los resultados obtenidos en una forma directa. Por esta razón, se prefiere la comunicación local [39], en la que cada individuo es capaz de comunicarse con los elementos más cercanos a él. Otro problema que surge al tratar de llevar a la práctica los experimentos de sistemas multi-robot, es que la mayoría de los simuladores asume que la posición del robot es conocida en todo momento y que cada robot conoce la localización de ciertos sitios importantes, esto en la práctica es difícil de implementar. Algunos trabajos reportan algoritmos que permiten la creación de sistemas de coordenadas de forma dinámica [40], sin embargo, solo tienen validez cuando los integrantes del sistema se encuentran estacionarios.

Algunos insectos sociales como las hormigas, muestran un gran desempeño en diversas actividades [41]. No existe un control central en estas comunidades. Son capaces de determinar la localización de comida con la ayuda de sustancias químicas llamadas feromonas, la feromona, es dejada en la superficie de la tierra para una movilización eficiente. Las hormigas construyen caminos utilizando feromonas para formar redes que conectan su nido con las fuentes de comida. El primer camino creado no es el óptimo, sin embargo, los caminos más cortos recibirán mayor concentración de feromonas (debido a que el tiempo requerido para recorrerlo es menor) y en los más largos la concentración de feromonas se reducirá. [42] Esto es muy interesante desde el punto de vista de comunicación multi-agente ya que un agente puede

dejar múltiples mensajes en el entorno. Este tipo de sistema de comunicación indirecta no solo omite la comunicación centralizada, sino que establece una relación entre información y posición en el entorno. [43]. Este fenómeno ha sido estudiado en varios campos de investigación [44] [45] [46] [47]. Debido a que las feromonas son sustancias químicas, son difíciles de utilizar en aplicaciones reales.

Algunos investigadores han utilizado el concepto de feromona virtual para imitar las características de las feromonas, [46] [48] [49]. En [50] y [42] implementan sistemas físicos que se aproximan a las características que se desean extraer de las feromonas utilizando la tecnología RFID. Esta tecnología utiliza variaciones de campo magnético para la comunicación; posee 2 componentes principales, los *tags* y los *readers*. Un *tag* es un dispositivo de identificación que posee un identificador único y una pequeña cantidad de memoria disponible. Un *reader* es un dispositivo que puede reconocer la presencia de un *tag* y hacer operaciones de lectura y escritura sobre él.

Utilizando una red de *tags* distribuidos a lo largo del espacio ocupado por los robots, es posible imitar el comportamiento de las feromonas de las hormigas y de esta forma definir caminos que sean útiles para la ejecución de una determinada tarea. Adicionalmente, los robots pueden utilizarlos para marcar sitios de interés general.

## IV. CONCLUSIONES Y TRABAJO FUTURO

### V.

### REFERENCES

- [1] P. Raven and G. Johnson. *Biology*, chapter 50 The Immune System. McGraw Hill.
- [2] F. Gruau. and P. Malbos. The blob: A basic topological concept for hardware-free distributed computation. In Cristian Calude, Michael J. Dinneen, and Ferdinand Peper, editors, *Unconventional Models of Computation, Third International Conference, UMC 2002, Kobe, Japan, October 15-19, 2002, Proceedings*, volume 2509 of *Lecture Notes in Computer Science*, pages 151–163. Springer, 2002.
- [3] R. Nagpal. Organizing a Global Coordinate System from Local Information on an Amorphous Computer. AI Memo No 1666 MIT, 1999.
- [4] M. Wooldridge and M. Jennings. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 1995.
- [5] R. L. King; A. B. Lambert; S. H. Russ; and D. S. Reese. The Biological Basis of the Immune System as a Model for Intelligent Agents. *IPDPS 1999 Workshop*, 1999.
- [6] Y. Thoma, D. Roggen, E. Sanchez, and J. Moreno. Prototyping with a bio-inspired reconfigurable chip. In *15th IEEE International Workshop on Rapid System Prototyping (RSP 2004)*. IEEE Computer Society, Los Alamitos, California, 2004, 2004.
- [7] S. Singh. S. Thayer. Kilorobot Search and Rescue Using an Immunologically Inspired Approach. In *Distributed Autonomous Robotic Systems*. Springer-Verlag, June 2002.
- [8] E. D'Hondt. Exploring the Amorphous Computing Paradigm. Technical report, VUB, Bruxelles, 2000.
- [9] B. Bonabeau C. Meyer. Swarm Intelligence: A whole new way to think about business. *Harvard Business Review*, 2001.
- [10] M. Resnick. *Turtles, Termites and Traffic Jams Explorations in Massively Parallel Microworlds*. The MIT Press Cambridge, Massachusetts, 1994.
- [11] F. Sahin S. Sathyanath. AISIMAM - An Artificial Immune System Based Intelligent Multi-Agent Model and its Application to a Mine Detection. *ICARIS 2002*, September 2002.
- [12] S. Singh S. Thayer. Development of an Immunology-Based Multi-Robot Coordination Algorithm for exploration and Mapping Domains. *IROS 2002: IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

- [13] S. Thayer S. Singh. Immunology Directed Methods for Distributed Robotics: A Novel, Immunity-Based Architecture for Robust Control & Coordination. *Proceedings of SPIE: Mobile Robots XVI*, November 2001.
- [14] M. Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 1993.
- [15] P Mohr, N Ryan, and J. Timmis. Exploiting Immunological Properties for Ubiquitous Computing Systems. *ICARIS 2004*, September 2004.
- [16] A. Avizienis and R. Avizienis. An Immune system paradigm for the design of fault tolerance systems. *Workshop 3: Evaluating and Architecting Systems for Dependability (EASY)*, 2001.
- [17] A. Avizienis. Toward systematic design of fault tolerant systems. *Computer*, 1997.
- [18] D. Bradley and A. Tyrrell. A Hardware Immune System for Benchmark State Machine Error Detection. In *Congress on Evolutionary Computation*, 2002.
- [19] J. Timmis, M. Ayara, and R. Duncan. Towards Immune Inspired Fault Tolerance In Embedded Systems. In *Proceedings of 9th International Conference on Neural Information Processing*. IEEE, 2002.
- [20] W. Buttera and V. Bove. Literally embedded processors. In *SPIE Media Processors*, 2001.
- [21] A. Berlin. *Towards Intelligent Structures: Active Control of Buckling*. PhD thesis, MIT. Department of Electrical Engineering and Computer Science, May 1994.
- [22] R. Nagpal. *Programmable Self-Assembly: COnstructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics*. PhD thesis, MIT. Department of Electrical Engineering and Computer Science., June 2001.
- [23] F. Mondada, G. C. Pettinari, I. Kwee, A. Guignard, L. Gambardella, D. Floreano, S. Nolfi, J.-L. Deneubourg, and M. Dorigo. SWARM-BOT: A swarm of autonomous mobile robots with self-assembling capabilities. In C.K. Hemelrijk and E. Bonabeau, editors, *Proceedings of the International Workshop on Self-organisation and Evolution of Social Behaviour*, pages 307–312, Monte Verità, Ascona, Switzerland, September 8-13, 2002. University of Zurich.
- [24] G. Baldassarre, S. Nolfi, and D. Parisi. Evolving mobile robots able to display collective behaviours. In C.K. Hemelrijk and E. Bonabeau, editors, *Proceedings of the International Workshop on Self-Organisation and Evolution of Social Behaviour*, pages 11–22, Monte Verità, Ascona, Switzerland, September 8-13, 2002. University of Zurich.
- [25] C. Jones and M. Mataric. Adaptive Division of Labor in Large-Scale Minimalist Multi-Robot Systems. *Proceedings of the IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS)*. Las Vegas, Nevada, 2003.
- [26] Jones and Maja J Mataric. *Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications*, chapter Behavior-Based Coordination in Multi-Robot Systems. Marcel Dekker, Inc, 2005.
- [27] J. McLurkin. Speaking Swarmish. *AAAI Spring Symposium*, 2006.
- [28] B. Gerkey, K. Stoy, and R. T. Vaughan. Player Robot Server. Technical report, USC Robotics Labs, 22 November 2000.
- [29] C. Camargo. Implementación de Sistemas Digitales Complejos Utilizando Sistemas Embebidos. *Memorias del XI Workshop de Iberchip ISBN 959-261-105-X*, 2005.
- [30] C. Camargo. ECBOT y ECBAT91 Plataformas Abiertas para el Diseño de Sistemas Embebidos y Co-Diseño HW/SW. *VIII Jornadas de Computación Reconfigurable y Aplicaciones*, Madrid España, 18 September 2008.
- [31] C. Camargo. ECBOT: Arquitectura Abierta para Robots Móviles. *Séptima Conferencia Iberoamericana en Sistemas, Cibernética e Informática*, Miami, USA, 2 July 2008.
- [32] C. Camargo. Linux como Herramienta de Desarrollo de Sistemas Embebidos. *XII Taller IBERCHIP*, San José, Costa Rica, March 2006.
- [33] C. Camargo. Linux como Plataforma de Desarrollo de Sistemas Embebidos. *Colombian Workshop on Circuits and Systems Bogotá, Colombia*, November 2007.
- [34] EPFL. e-puck EPFL Education robot. <http://www.e-puck.org/>.
- [35] Maja J Mataric. *Interaction and Intelligent Behavior*. PhD thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, May 1994.
- [36] Mussa-Ivaldi F. A. and S. Giszter. Vector field approximation: a computational paradigm for motor control and learning. *Biological Cybernetics* 67, 1992.
- [37] M. Mataric. Integration of Representation Into Goal-Driven Behavior-Based Robots. *IEEE Transactions on Robotics and Automation* 8(3), 1992.
- [38] Henry Y. K. Lau and Vicky W. K. Wong. An Immunity Based Distributed Multiagent-Control Framework. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, January 2006.
- [39] D. Coore H. Abelson, D. Allen. Amorphous Computing. <http://www.ai.mit.edu/publications/pubsDB/pubs.html>, 29 August 1999.
- [40] D. Coore. Establishing a Coordinate System on an Amorphous Computer. <http://swiss.ai.mit.edu/projects/amorphous>, 1997.
- [41] B. Holldobler and E. Wilson. *The Ants*. Springer-Verlag, 1990.
- [42] Herianto, T. Sakakibara, and D. Kurabayashi. Artificial Pheromone System Using RFID for Navigation of Autonomous Robots. *Journal of Bionic Engineering*, Elsevier Limited and Science Press, 2007.
- [43] D. Kurabayashi. Toward Realization of Collective Intelligence and Emergent Robotics. *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, 1999.
- [44] Edelstein-Keshet L. Simple model for trail-following behaviour; trunk trails versus individual foragers. *Journal of Mathematical Biology*, 1994.
- [45] Payton D, Daily M, Estowski R, Howard M, and Lee C. Pheromone robotics. *Autonomous Robots*, 2001.
- [46] Sugawara K, Kazama T, and Watanabe T. Foraging behavior of interacting robots with virtual pheromone. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.
- [47] Panurak H V D. Go to the ant. *Engineering principles for natural multi-agent systems*. *Annals of Operations Research*, 1997.
- [48] Tanaka K, Yo Ishigaki, Inoue H, and Itoh M. Safety monitoring system by autonomous mobile sensors utilizing pheromone communication. *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation; International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, 2005.
- [49] Mamei M, Quaglieri R, and Zambonelli F. Making tuple space physical with RFID tags. *Proceedings of ACM Symposium on Applied Computing*, 2006.
- [50] Mamei M and Zambonelli F. Physical deployment of digital pheromones through RFID technology. *Swarm Intelligence Symposium*, 2005.



Carlos Camargo es Ingeniero Electricista de la Universidad Nacional de Colombia, Magister en Ingeniería Eléctrica de la Universidad de los Andes y Candidato a Doctor en Ingeniería Eléctrica en la Universidad Nacional de Colombia. Es profesor del Departamento de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Colombia en el área de los sistemas Digitales.

**PLATAFORMAS ABIERTAS HARDWARE/SOFTWARE PARA APLICACIONES EN ROBOTICA, V Congreso Internacional de Ingeniería Mecánica y III de Ingeniería Mecatrónica.**



## PLATAFORMAS ABIERTAS HARDWARE/SOFTWARE PARA APLICACIONES EN ROBOTICA

Carlos Camargo\*

\*Universidad Nacional de Colombia,  
[cicamargoba@unal.edu.co](mailto:cicamargoba@unal.edu.co)

### RESUMEN

La robótica móvil es un campo de investigación que crece rápidamente; en la actualidad existe un gran número de grupos que trabajan en diferentes áreas [1], [2], [3], [4], de sus experiencias, se concluye, que para desarrollar algoritmos aplicables en robótica, es necesario contar con plataformas físicas que permitan validar los algoritmos y modelos computacionales propuestos. El Departamento de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Colombia ha venido trabajando en la creación y adaptación de metodologías de diseño y técnicas de fabricación de sistemas digitales [5], [6], [7], como resultado, se diseñaron y fabricaron plataformas de desarrollo abiertas que permiten ser estudiadas, utilizadas y modificadas (incluso con fines comerciales), ahorrando años de investigación a los usuarios (cualquier persona interesada) de estos conocimientos. Esto hace parte de una metodología para la transferencia tecnológica en el área de diseño de sistemas embebidos [8] que gira entorno al conocimiento como un bien común, por lo que el acceso a dicho bien es un derecho y no existe ninguna restricción para acceder a sus beneficios; este concepto novedoso va contra el modelo tradicional de considerar el conocimiento como un recurso por el que se debe pagar. En este trabajo se presentan dos plataformas de desarrollo abiertas para el estudio de sistemas embebidos con aplicaciones en robótica móvil.

Research on mobile robotics has been rapidly growing , there are many research groups currently working in this area; the experiences gained by these projects concludedes that for algorithmic development in robotics, is necessary to have hardware and software platforms to validate the proposed algorithms and computational models. The Department of Electrical and Electronic Engineering, of the National University of Colombia has been working on the development and adaptation of methodologies for design and manufacturing techniques of digital systems in the local industry, as result of this work, have been released some open hardware platforms that allow to be studied, used and modified (even commercially), saving years of research to anyone interested. This, is part of a new methodology for technology transfer in embedded system design based on the knowledge as a commons, so, the access to that resource is a right and is garantized, this new concept goes against the traditional model of knowledge as an economic resource. This paper discusses two open hardware platforms for the study of embedded systems with applications in mobile robotics.

**Palabras Clave:** Diseño digital (digital design), robótica (robotics), sistemas embebidos (embedded systems), transferencia tecnológica (technology transfer), conocimiento como bien común (knowledge as a commons)

## 1 INTRODUCCIÓN

La utilización de la robótica en la educación básica y media ha venido en aumento, y su uso adecuado permite el desarrollo de habilidades asociadas al constructivismo (aprender haciendo), y al aprendizaje significativo (aprendizaje relacionado con necesidades, intereses propios) haciendo que el estudiante aprenda el valor de la nueva tecnología y como esta puede ser utilizada para dar solución a problemas comunes. Existen dos formas de utilizar la robótica en la educación [10]: La robótica como objeto de aprendizaje: enfocado a aspectos relacionados con el robot como construcción, programación e inteligencia artificial y la robótica como herramienta de aprendizaje: visto como proyecto interdisciplinario que involucra ciencias, matemáticas, tecnologías de la información y comunicaciones.

En la mayoría de los estudios en robótica realizados en el país se utilizan plataformas comerciales provenientes del extranjero para la implementación de los algoritmos de control, esta práctica genera dependencia tecnológica hacia economías más desarrolladas y limita el campo de acción de nuestros profesionales a la integración de software y hardware. Por otro lado, los grupos de investigación están compuestos por un solo tipo de profesionales (ing. eléctrico/electrónico, de sistemas, mecánico o mecatrónico) lo que ocasiona que los problemas sean atacados desde un solo punto de vista, descuidando los otros factores: los electrónicos dedican mayor esfuerzo a la arquitectura de la plataforma electrónica que implementa la funcionalidad; los mecánicos se ocupan de la parte física; y los ingenieros de sistemas se preocupan por la programación eficiente de la funcionalidad.

Este artículo muestra un trabajo interdisciplinario realizado por ingenieros de sistemas, electrónicos y mecatrónicos; cuyo resultado es el diseño de dos plataformas robóticas abiertas fáciles de programar con gran capacidad de computo y de comunicaciones: ECBOT y SIEBOT; las cuales permiten estudiar su funcionamiento, realizar modificaciones y desarrollar aplicaciones comerciales en otros campos diferentes a la robótica; para que esto sea posible: se suministran los archivos necesarios para su fabricación, es decir, los esquemáticos y los archivos de la placa de circuito impreso (utilizando herramientas abiertas como Kicad o geda); se proporciona la cadena de herramientas de compilación y depuración para desarrollo de aplicaciones; se proporcionar el código fuente de: el programa que inicializa la plataforma (*bootloader*), la herramienta que carga dicho programa en la memoria no volátil (*usbboot*), el sistema de archivos y aplicaciones (*openwrt*); se suministra la documentación completa que indica como fue diseñada, construida, como utilizarla; y se permite el acceso a aplicaciones y tutoriales que explican el funcionamiento de los diferentes componentes y finamente se cuenta con la posibilidad de fabricación y montaje. Estas características definen el concepto *hardware copyleft*, el cual, al ser inspirado en el movimiento FOSS, comparte su filosofía [8] y son su complemento perfecto.

En la actualidad existen plataformas comerciales que proporcionan la documentación necesaria para realizar la programación completa de sus dispositivos, pero no existen muchas que permitan realizar modificaciones en su hardware o crear nuevos productos a partir de ellas. La plataforma más completa que se acerca a los requerimientos del hardware copyleft fué desarrollada por el EPFL [3] y proporciona los archivos gerber con los que se puede reproducir la placa de circuito impreso, pero no suministra los archivos de diseño necesarios para modificarla.

## 2 MATERIALES Y MÉTODOS

La metodología utilizada para adquirir y absorber los conocimientos necesarios para el diseño y fabricación de estas plataformas robóticas se basa en el uso de herramientas abiertas para el diseño de las placas de circuito impreso y para la creación de la aplicación software; las plataformas de desarrollo son liberadas bajo el esquema de licencias *Creative Commons CC: BY-SA*, lo que permite su distribución, uso, reproducción y modificación incluso para fines comerciales con los requisitos de dar crédito al autor del trabajo original y que los trabajos derivados posean el mismo esquema de licencias. Esto se hace con el fin de crear una comunidad que aporte conocimientos que pueden ser utilizados sin restricción alguna por quien esté interesado. A continuación se listan las actividades que se realizaron para adquirir, absorber, asimilar y aplicar los conocimientos necesarios para el diseño, implementación y programación de las plataformas robóticas[8]:

- Elección: Identificación del estado de la plataforma tecnológica existente para identificar facilidades y necesidades; identificación de niveles de complejidad de la tecnología; selección de una alternativa que pueda implementarse y de resultados a mediano y corto plazo con no muy altas inversiones de capital.
- Adquisición: Adquisición de plataformas adecuadas de desarrollo HW y SW; identificación de herramientas de desarrollo HW y SW y su origen (abierto/cerrado).
- Adopción: Utilización de plataformas de desarrollo para estudio de metodologías de diseño; uso de ingeniería inversa para entender la arquitectura, funcionamiento y programación de productos comerciales; utilización de productos comerciales para adaptarlos a problemas locales (Integración); tomar conciencia de la importancia del uso de esta tecnología.
- Absorción: Desarrollo o adaptación de metodologías de diseño y procesos de fabricación; desarrollo de productos tecnológicos propios; enseñanza de metodologías de diseño y procesos de fabricación en centros de educación consolidados.
- Aplicación: Desarrollo de soluciones a problemas locales; uso de metodologías de diseño en la concepción, diseño e implementación de sistemas digitales utilizando la tecnología; utilización de procesos de fabricación adaptados al entorno local; desarrollo de proyectos académicos utilizando esta tecnología.
- Difusión: Vinculación de la academia para incluir los conocimientos generados en los programas académicos de las carreras relacionadas; capacitación a la industria local sobre el uso de la tecnología, las metodologías de diseño y procesos de producción; creación de una comunidad que utilice, mejore y aumente el conocimiento generado; hacer que el conocimiento generado en los pasos anteriores este disponible a todos los interesados; dar a conocer los procesos, productos y conocimientos creados a los generadores de políticas de estado.
- Desarrollo: Aumento de la demanda de productos, bienes y servicios relacionados; compra de maquinaria que permita la fabricación masiva de forma local; diseño de nuevos componentes (Circuitos Integrados, IPs, software CAD); creación de políticas gubernamentales que protejan la producción local; participación activa de la academia en la solución de problemas y en la formulación de políticas de gobierno relacionadas.

### 3 RESULTADOS

El resultado de este proceso de transferencia tecnológica y de conocimientos en el área de diseño digital aplicado a la robótica, fue el desarrollo de las plataformas hardware copyleft ECBOT, SIE y dos interfaces de programación que permiten su uso a cualquier nivel de formación; la información necesaria para reproducir y modificar dichas plataformas se puede encontrar en el portal web *linuxencaja* (<http://www.linuxencaja.com>); adicionalmente, este portal proporciona dos canales de comunicación: listas de correo y wiki que pueden ser utilizadas por cualquier persona para solicitar soporte o para publicar proyectos abiertos relacionados con el diseño digital.

#### 3.1 ECBOT

##### 3.1.1 Arquitectura Software

La columna vertebral del componente Software es el proyecto Player/Stage, creado por el grupo de robótica de la University of Southern California. Este proyecto esta dividido en dos partes:

1) Player: Servidor que proporciona una interfaz flexible a una gran variedad de sensores y actuadores, utiliza un modelo cliente/servidor basado en sockets TCP lo que permite que los programas de control del robot sean escritos en cualquier lenguaje y puedan ser ejecutados en cualquier plataforma (cliente) que posea una conexión de red con el robot (servidor). Además, soporta múltiples conexiones concurrentes de clientes, lo cual es muy útil en estudios de control descentralizado.

2) Stage: simulador escalable, trabaja con robots que se mueven, realizan operaciones de sensado en un entorno de dos dimensiones y son controlados por Player; proporciona robots virtuales que pueden interactuar con dispositivos físicos simulados. Una de las características más atractivas del proyecto Player-Stage es que permite la creación de sensores y actuadores que simulan el comportamiento de los dispositivos reales teniendo en cuenta aspectos físicos como forma, peso y material.

##### 3.1.2 Arquitectura Hardware

Uno de los requerimientos iniciales fue la capacidad de ejecutar de forma nativa el cliente/servidor Player; para esto, es necesario que ECBOT pueda correr aplicaciones Linux; por esta razón, se diseñó la plataforma abierta ECB AT91 [5], la cual fué la base del desarrollo de ECBOT.

##### 3.1.3 Especificaciones de ECBOT

1) Sensores y Actuadores: ECBOT cuenta con (ver Figura 1) un bus I2C encargado de realizar la interfaz con el mundo real (ya que el procesador central AT91RM9200 no posee conversores A/D), 2 microcontroladores de 8 bits (AVR de ATMEL) permiten el manejo de: seis sensores infrarrojos de proximidad y luz de ambiente; 2 motores DC utilizados en tareas de evasión de obstáculos y 8 LEDs (RGB) que representan el estado interno del Robot. Para reducir el costo de los componentes ECBOT implementa un control de posición y velocidad de motores utilizando un método basado en la fuerza electromotriz que no requiere partes mecánicas. Adicionalmente,



ECBOT dispone de una cámara digital que permite la implementación de algoritmos de seguimiento de color, una FPGA se encarga de: controlar el sensor de imagen y de la comunicación con el procesador; proporcionar 10 señales digitales de propósito general que pueden ser utilizadas para el manejo de servos motores u otros actuadores.

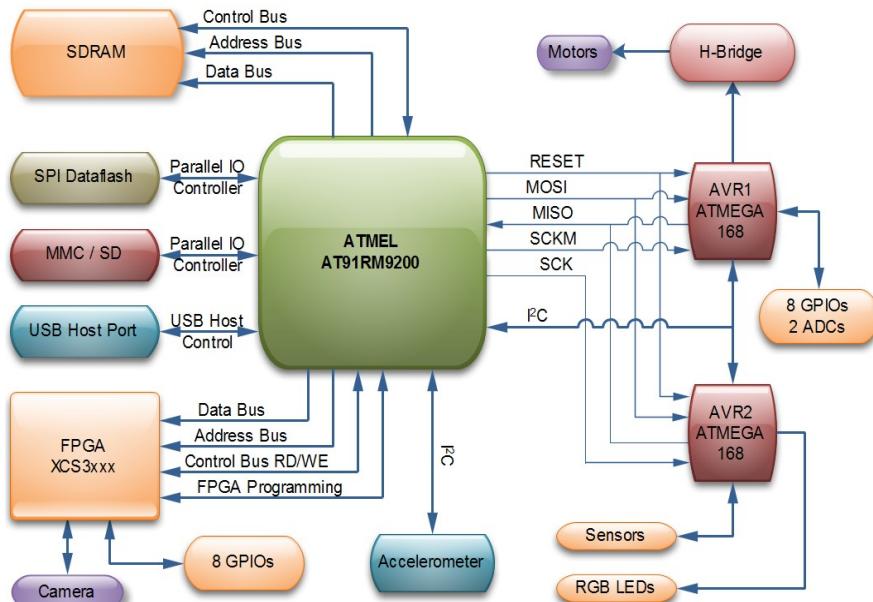


Figura 1: Diagrama de bloques de la plataforma robótica ECBOT.

Los microcontroladores de 8 bits y la FPGA son programados/configurados por el procesador central a través de pines de entrada/salida de propósito general (GPIOs), los archivos de programación/configuración son almacenados en el sistema de archivos de la plataforma y son transmitidos utilizando un enlace inalámbrico (WiFi) y pueden ser modificados en cualquier momento sin necesidad de conectores adicionales, ni de conexiones físicas con la plataforma; esto hace que ECBOT sea autónomo y que pueda ser programada “en caliente” y de forma remota.

2) Comunicaciones: Se dispone de tres canales de comunicación: un puerto USB host que permite la conexión de cualquier dispositivo *USB device*, como adaptadores USB-WiFi 802.11, modems 3G, EDGE, GSM, tranceivers ZigBee, etc. permitiendo el control y reconfiguración de forma remota; un puerto serie que en las primeras etapas del desarrollo se utiliza para cargar las imágenes del *loader*, *bootloader* e imagen del *kernel*, y se puede utilizar para conectar un GPS o un sensor con protocolo serial asíncrono; y un puerto I<sub>2</sub>C que puede ser utilizado para adicionar cualquier sensor que cumpla con este protocolo.

3) Memorias y sistema de archivos: Se cuenta con una memoria flash serial de 2 Mbytes donde se almacenan las imágenes del *loader*, *bootloader* y *kernel*, esta memoria puede ser modificada utilizando un *loader* que se ejecuta al inicializar la plataforma, o puede modificarse desde Linux a través de un dispositivo MTD (Memory Technology Device). Permite la utilización de una memoria RAM tipo SDRAM de hasta 64 MBytes, suficiente para ejecutar una gran variedad de aplicaciones.

ECBOT utiliza la distribución de Linux *buildroot*, basada en la librería de C *uClibc*, concebida para trabajar con sistemas embebidos, es altamente configurable y proporciona una gran variedad de aplicaciones que pueden ser instaladas siguiendo las instrucciones de un script de instalación, lo que facilita la adición de aplicaciones no soportadas por la distribución oficial; en nuestro caso se agregaron 3 aplicaciones: el servidor *player* portado a la plataforma ECBOT, el programador de microcontroladores para microcontroladores AVR *uisp* y el programador para FPGA *xc3sprog*. El sistema de archivos generado por *buildroot* y las aplicaciones necesarias para el funcionamiento de ECBOT son almacenadas en una memoria SD.

4) Unidad Central de Procesamiento: El cerebro de ECBOT es un procesador de 32 Bits de la familia ARM de ATMEL el AT91RM9200, que corre a 180 MHz. Este procesador goza de gran popularidad dentro del grupo de desarrolladores de drivers para Linux, por lo que casi todos sus periféricos están soportados.

### 3.2 SIEBOT

SIEBOT integra la plataforma de desarrollo para co-diseño Hardware/Software SIE [11] con una interfaz que permite el control de sensores y actuadores. La Figura 2 muestra su diagrama de bloques, en ella, podemos encontrar un procesador que posee periféricos para comunicación serial (UART), memorias micro-SD, un puerto I2C, un LCD a color de 3 pulgadas, 2 entradas y salidas de audio estéreo, 2 entradas análogas; una FPGA que proporciona 25 señales de entrada/salida digitales de propósito general (GPIOs) y controla un conversor análogo digital de 8 canales. Existen dos canales de comunicación entre la FPGA y el procesador: uno para controlar el puerto JTAG, lo que permite la configuración de la FPGA desde el procesador (eliminando la necesidad de cables de programación); y otro que proporciona el bus de datos, dirección y control para comunicarse con las tareas HW o periféricos implementadas en la FPGA. El procesador utilizado es un *Ingenic JZ4725* (XBURST - MIPS) corriendo a 400MHz, se dispone de una memoria NAND de 2GB para almacenamiento de datos y programas, así como de una memoria SDRAM de 32 MB, lo que permite la ejecución de una gran variedad de aplicaciones Linux.

1) Arquitectura: Como ya se mencionó la FPGA de SIE utiliza los buses de datos, dirección y control para comunicarse con el procesador, por lo que es posible implementar un número indeterminado de periféricos en ella como, controladores de sensores, controladores de motores servo, generadores de señales PWM, interfaces de comunicaciones seriales, I2C o SPI.

2) Comunicaciones: SIE proporciona tres canales de comunicación: un puerto USB device, permite la comunicación con un computador personal a través de un puerto USB device, este canal permite la programación de la memoria NAND no volátil con el lanzador de Linux *u-boot*, la imagen del kernel y el sistema de archivos (*openwrt*), adicionalmente, permite el inicio de consolas remotas utilizando el protocolo *ssh*; dos puertos seriales, con niveles de voltaje RS232 uno conectado al procesador (actualmente se conecta un GPS) y otro a la FPGA 3; un puerto I2C que puede ser utilizado para conectar una amplia gama de sensores que proporcionan su salida en este formato.

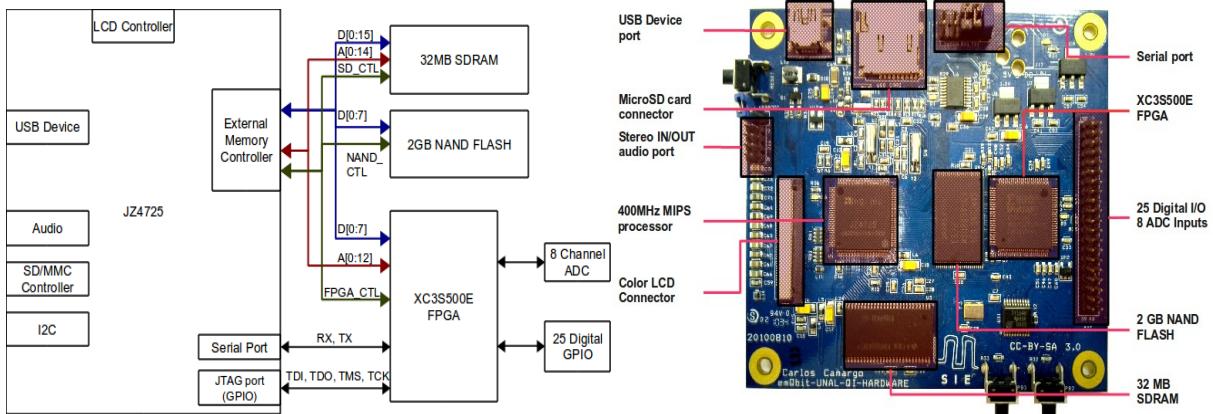


Figura 2: Diagrama de bloques de la plataforma robótica SIEBOT.

3) Sensores y actuadores: La tarjeta de sensores y actuadores soporta: un controlador puente-h para manejar dos motores de 6.8V, 2 encoders de cuadratura, 2 sensores de proximidad, una etapa de amplificación para 4 servomotores y 8 entradas analógicas.

### 3.3 Interfaces Para Programación de Aplicaciones

Se desarrollaron dos entornos gráficos de programación abiertos con el fin de abstraer las tediosas operaciones de programación a bajo nivel y configuración de las plataformas robóticas: SIE-Blocks y SIE-CodeGenerator, desarrollados en Java y QT respectivamente, lo que facilita su portabilidad. Estos entornos permiten la programación de aplicaciones utilizando lenguajes gráficos y su posterior descarga a las plataformas, sus simplicidad permite que sean utilizadas en diferentes niveles de formación, desde la educación básica primaria hasta la formación profesional y de esta forma difundir el uso de esta tecnología y concienciar a la sociedad de su importancia.

#### 3.3.1 SIE Blocks:

SIE Blocks está basado en el proyecto openblocks [12] desarrollado en el MIT, el cual a su vez sigue el trabajo en entornos de programación gráficos Starlogo [13], los que buscan reducir las barreras que presentan los lenguajes de programación tradicionales al ser utilizados por principiantes. La aproximación a la programación gráfica del MIT recibe el nombre de *block programming* y en ella, los usuarios manipulan y conectan piezas de rompecabezas que representan objetos para construir programas; la silueta de estos bloques representa implícitamente la sintaxis del lenguaje y estos solo pueden conectarse con bloques con siluetas complementarias, eliminando la necesidad de memorizar reglas complejas y de esta forma ayudar a reducir la curva de aprendizaje. Esta aproximación ha demostrado ser muy efectiva aplicaciones comerciales como LEGO Mindstorm, Cricket LOGO y Labview.

En la Figura 3 se muestra una captura de pantalla de la ejecución de SIE Blocks, en ella podemos observar un menú que contiene una serie de bloques constructores separados según su función: *GPIOs, Functions, Movement, Loops, Data, Display y Math*; en la zona central se pueden colocar los diferentes bloques constructores para formar un determinado algoritmo. En la parte superior, se encuentra un menú con las operaciones que puede realizar SIEBlocks. *Load/save*:

permite almacenar y cargar diferentes algoritmos en formato XML; *connect with SIE*: establece una comunicación con la plataforma SIEBOT utilizando el protocolo ssh; y *Upload and compile*: Convierte el algoritmo a un formato XML para después ser convertido en un script que ejecutará un intérprete *Lua* que reside en SIEBOT.

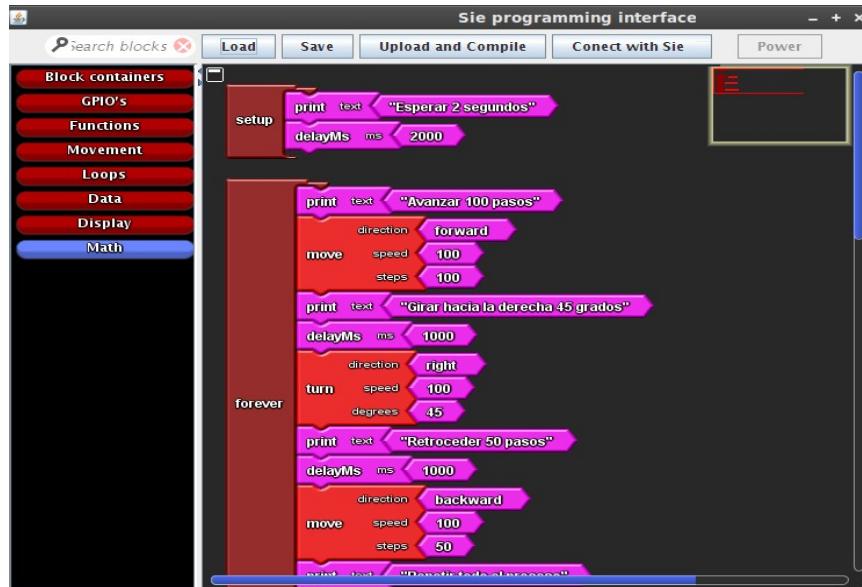


Figura 3: Entorno de programación SIEBlocks.

### 3.3.2 SIECG

SIE Code Generator permite la creación de nuevos elementos y librerías en tiempo de ejecución, no utiliza ningún lenguaje intermedio tipo Lua o Python, genera código en C a partir de la entrada gráfica y lo compila para la arquitectura XBURST - MIPS; adicionalmente, proporciona una serie de utilidades para transferir los archivos de configuración de la FPGA y el ejecutable generado por el proceso de compilación a la plataforma SIEBOT. En la Figura 4 se puede ver una captura de pantalla.

SIE CD permite la creación de componentes y librerías que pueden ser utilizados como bloques constructores de aplicaciones; existen dos tipos de bloques básicos: software - que ejecutan una función determinada (secuencia de instrucciones) en el procesador de la plataforma, o software-hardware - que controla una tarea hardware implementada en la FPGA (periféricos dedicados); en la descripción del componente es necesario incluir el segmento de código en C que implementa la función deseada, SIE CG se encarga de unir estos segmentos de código en un solo código fuente y generar la aplicación que debe ser ejecutada en la plataforma robótica para obtener la funcionalidad requerida. En la figura 4 podemos observar: el componente software *while* equivalente a la sentencia *while (1)* de C; el componente *Frame Buffer Print Line* imprime dos mensajes de texto en la pantalla utilizando el buffer de video de Linux /dev/fb0 “Executing Code... y *CHA[0]*”; el segundo componente *ADC Single Channel* controla un periférico que maneja un

conversor serial de ocho canales, el resultado de la conversión del canal 0 se imprime en consola y se muestra en la pantalla de SIEBOT en forma numérica y en una gráfica de barras.

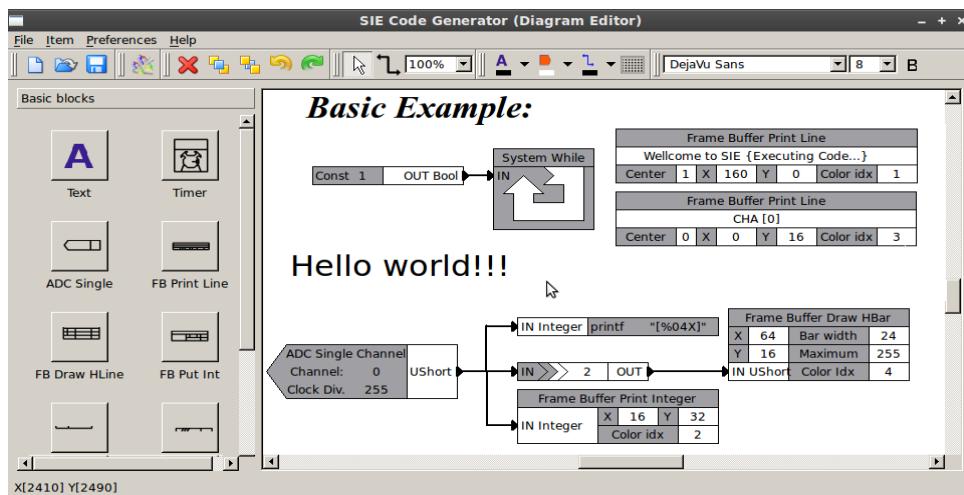


Figura 4: Entorno de programación SIECG.

#### 4 DISCUSIÓN

Este artículo presenta dos plataformas robóticas *copyleft hardware* diseñadas, construidas y programadas como parte de un proyecto de investigación que busca difundir el diseño y construcción de sistemas digitales en el país; la información necesaria para reproducirlas y modificarlas se encuentra disponible en un servidor público y quien esté interesado puede utilizarlas en centros de formación de varios niveles, o como base de desarrollos comerciales.

Se presentaron dos entornos de programación gráfico que facilitan el proceso de aprendizaje, ayudando a adquirir habilidades en programadores principiantes relacionadas con algoritmia y pensamiento estructurado; abstrayendo la complejidad de la sintaxis en una interfaz gráfica de alto nivel.

Con estas dos plataformas se demuestra que en el país es posible realizar productos que utilizan tecnologías de punta y se encuentran en el mismo nivel de complejidad de productos diseñados y construidos en países desarrollados.

Las plataformas hardware copyleft son una herramienta poderosa en el proceso de transferencia tecnológica ya que permiten entender, adaptar, absorber y asimilar el conocimiento asociado a la tecnología necesaria para diseñar y construir sistemas digitales.

En este momento ambas plataformas se encuentran en proceso de producción en masa, y se está trabajando en el contenido de los programas académicos que se utilizarán en educación básica y media. El primer piloto se implementará en el Instituto Técnico Central 2, en donde, se presentan los tres ciclos de interés: enseñanza media, técnica y profesional.

## REFERENCIAS

- [1] F. Mondada, G. C. Pettinaro, et al. SWARMBOT: A swarm of autonomous mobile robots with self-assembling capabilities. In C.K. Hemelrijk and E. Bonabeau, editors, Proceedings of the International Workshop on Self-organisation and Evolution of Social Behaviour, pages 307–312, University of Zurich.
- [2] C. Jones and M. Mataric. Adaptive Division of Labor in Large Scale Minimalist Multi-Robot Systems. Proceedings of the IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS). Las Vegas, Nevada, 2003.
- [3] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. Zufferey, Floreano, D. and Martinoli, A. (2009) The e-puck, a Robot Designed for Education in Engineering. Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions.
- [4] J. McLurkin. Speaking Swarmish. AAAI Spring Symposium, 2006.
- [5] C. Camargo. ECBOT y ECB\_AT91 Plataformas Abiertas para el Diseño de Sistemas Embebidos y Co-diseño HW-SW, VIII Jornadas de Computación Reconfigurable y Aplicaciones, 2008.
- [6] C. Camargo. Hardware copyleft como Herramienta para la Enseñanza de Sistemas Embebidos, Simposio Argentino de Sistemas Embebidos 2011, Buenos Aires Argentina.
- [7] C. Camargo. ECBOT: Arquitectura Abierta para Robots Móviles, IEEE Colombian Workshop on Circuits and Systems, Bogotá 2007.
- [8] C. Camargo. Metodología Para la Transferencia Tecnológica en la Industria Electrónica Basada en Software Libre y Hardware Copyleft, VI Congreso Internacional de la Red de Investigación y Docencia en Innovación Tecnológica RIDIT 2011.
- [9] R. M. Stallman. The GNU Operating System and the Free Software Movement Voices from the Open Source Revolution. O'Reilly and Associates, 1999.
- [10] D. Alimisis and C. Kynigos. Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods. School of Pedagogical and Technological Education (ASPETE), 2009.
- [11] C. Camargo. SIE: Plataforma Hardware copyleft para la Enseñanza de Sistemas Digitales. XVII Workshop de Iberchip, Bogotá Colombia, February 2011.
- [12] Ricarose Vallarta Roque. OpenBlocks : an extendable framework for graphical block programming systems. Master's thesis, MIT, 2007.
- [13] M. Resnick. Starlogo: An Environment for Decentralized Modeling and Decentralized Thinking. Conference Companion on Human Factors in Computing Systems. New York, NY, USA: ACM Press, 1996.

## A.2. Sobre la Metodología

**Método de Enseñanza/Aprendizaje en Sistemas Embebidos Utilizando Hardware Copy-left** A publicar en: IEEE Latin America Transactions

# Método de Enseñanza/Aprendizaje en Sistemas Embebidos Utilizando Hardware copyleft

C. Camargo

**Abstract**—Embedded systems are a fundamental part of almost all human activities. This, coupled with the availability of free and open software development tools, opens up great possibilities for developing countries as they are not large capital investments needed for the conception, design, and manufacture of these systems. However, currently very few universities offer courses for building the skills necessary to achieve a marketable product, increasing dependence on Asian technology. This paper presents a methodology for teaching embedded system design using open hardware and software tools.

**Keywords**— *Sistemas Embebidos, educación en ingeniería, hardware copyleft.*

## I. INTRODUCCIÓN

El mercado de los sistemas embebidos continúa en aumento y su campo de acción se ha extendido a casi todas las actividades humanas. Según BBC, inc. el mercado para el software embebido creció de \$1.6 billones a \$3.5 billones en 2009, con una tasa de crecimiento anual (AAGR) del 16%, mientras la tasa de crecimiento para las tarjetas embebidas es del 10%; según Venture Development Corporation (VDC) más de un billón de dispositivos embebidos fueron vendidos en el 2004. Por otro lado, según VDC el porcentaje de dispositivos basados en sistemas operativos comerciales tiende a disminuir del 43.1% en 2001 a 37.1% en 2004, esta tendencia se debe a la utilización de herramientas de libre distribución GNU/Linux [1].

En la actualidad estamos presenciando una tendencia global a delegar las tareas de manufactura de sistemas digitales a países asiáticos, donde la mano de obra calificada es abundante y barata; se presentan casos donde los creadores de una determinada tecnología no la desarrollan y dejan que estos países se beneficien de sus descubrimientos [2] reduciendo de forma considerable la producción local. Esta situación se agrava a medida que las grandes empresas manufactureras asiáticas como Foxconn capturan la producción de los grandes diseñadores como Apple, Nokia, DELL, HP y Microsoft, lo que genera el cierre de empresas manufactureras a lo largo del mundo, con la consecuencia de pérdida de empleo masivo. En la actualidad según el Bureau of Labor Statistics y Thomson Financial Extel Company Report Foxconn emplea a más personas que Apple, Dell, Microsoft, HP, Intel y Sony combinados; esta situación es más grave en países en vía de desarrollo, donde no existe la plataforma tecnológica para diseñar dispositivos digitales, y son importadores de tecnología sin la capacidad de generar productos que satisfagan necesidades locales.

La tendencia moderna en los programas académicos a la utilización de herramientas de alto nivel para la enseñanza en áreas afines al desarrollo de dispositivos digitales [3]

ocasiona que los profesionales no adquieran las habilidades necesarias para completar la cadena concepción - diseño - implementación y operación, en la mayoría de los casos se generan habilidades para la concepción y el diseño a alto nivel y dejan los otros pasos en manos de herramientas especializadas y/o a empresas asiáticas. Esta situación resulta la más atractiva desde el punto de vista económico, ya que no es necesario adquirir maquinaria costosa ni contratar personal calificado para operarlas; sin embargo, limita la generación de empleo local a personas con un nivel de formación alto [2] generando desempleo en las personas menos capacitadas.

Según John Hall presidente y CEO de Linux International “algunas facultades preparan a la gente en el uso de productos en vez de tecnologías de nivel básico” [3]. Esta situación unida al abandono de la implementación hace que la dependencia con las empresas manufactureras asiáticas aumente cada vez más. Según el ex-director ejecutivo de Intel Andy Grove [2] la solución está en hacer de la creación de empleo la política económica gubernamental más importante y hacer que las demás giren en torno a ella. Además, es necesario volver a la producción local con el fin de generar nuevos empleos, y volver a adoptar medidas que protejan la producción interna de los productos asiáticos. Sin embargo, para lograrlo es necesario crear en los profesionales las habilidades para implementar productos comercializables.

En este artículo presentamos un programa académico basado en la utilización de software y hardware libre para el área de electrónica digital que desarrolla las habilidades necesarias para Concebir, Diseñar Implementar y operar sistemas digitales.

### A. Flujo de diseño de sistemas embebidos

Los Sistemas Embebidos son sistemas heterogéneos que contienen componentes Software (microcontroladores, microprocesadores y DSPs) y Hardware (funciones implementadas en Dispositivos lógicos programables PLDs); por este motivo, es necesario adquirir habilidades en la utilización de lenguajes de programación como C o C++ para implementar las funciones software y Verilog o VHDL para la implementación de las tareas hardware; adicionalmente, deben conocer las diferentes formas de comunicación entre estos dos tipos de funciones. Aunque en el mercado existen herramientas que permiten la entrada de diseño utilizando lenguajes de alto nivel como SystemC o SpecC y proporcionan el código para implementar las tareas software, hardware y su interfaz de comunicación; no es recomendable utilizarlas en el ciclo de formación básico ya que impide que se conozca el flujo de diseño completo, suministrando un nivel de abstracción en el cual no es necesario conocer la arquitectura de la plataforma utilizada para la implementación.

En la figura 1 se muestran los conceptos que deben dominar los diseñadores de sistemas embebidos, y las tareas que deben realizarse para la concepción, diseño e implementación de un sistema embebido. En gran parte de los programas académicos se estudian únicamente los temas relacionados con la concepción y diseño centrándose en las especificaciones funcionales del sistema, utilizando herramientas comerciales o COTS (Commercial off-the-shelf) para su implementación. Esta combinación genera dependencia e impide la generación de habilidades necesarias para implementar un sistema digital teniendo en cuenta restricciones económicas, físicas, eléctricas, ergonómicas, comerciales, etc.

### 1) Dominios de Diseño y Niveles de abstracción:

Existen tres dominios en los que se puede describir un sistema digital [5] Funcional: Describe el comportamiento funcional y temporal del sistema Estructural: Describe su composición a partir de bloques básicos y Físico relacionado con la estructura física del sistema a nivel de circuito integrado o placa de circuito impreso [6]. Cada uno de estos dominios puede ser descrito utilizando diferentes niveles de abstracción; un nivel de abstracción alto permite el uso de lenguajes de alto nivel facilitando la entrada de diseño al extraer la funcionalidad de la parte física; por otro lado, los niveles de abstracción bajo utilizan bloques constructores elementales.

En el mercado existen herramientas que permiten entradas de diseño a nivel funcional utilizando especificaciones y algoritmos y de forma automática y optimizada generan representaciones en diferentes niveles de abstracción en los dominios estructural y físico. Es decir, a partir de las especificaciones y algoritmos que indican el funcionamiento del sistema, pueden generar archivos para la fabricación de un circuito integrado o archivos de configuración/programación que pueden ser utilizados en dispositivos comerciales. Desde el punto de vista comercial esto es muy útil ya que permite reducir el tiempo de diseño y los costos asociados. Sin embargo, presentan los inconvenientes mencionados anteriormente y por lo general son muy costosas.

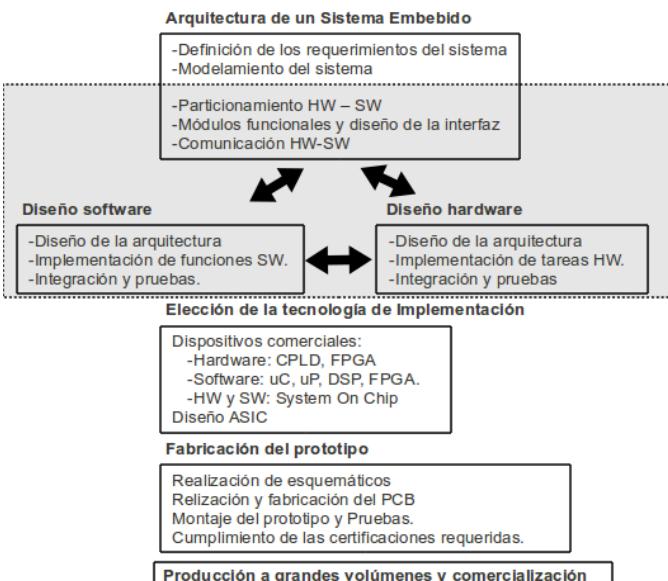


Fig. 1. Educación de sistemas embebidos. Inspirada en [4].

## II. MÉTODO PARA LA ENSEÑANZA/APRENDIZAJE DE SISTEMAS EMBEBIDOS

Basándose en la metodología de diseño para sistemas embebidos [7], en los dominios de diseño y niveles de abstracción de Gajski-Kuhn; se realizó una división de temas que busca crear habilidades de forma gradual e incremental. (en la Figura 5 podemos observar esta división). Como herramienta de desarrollo hardware se utilizará una plataforma que proporcione los archivos y documentos necesarios para replicarla, modificarla y pueda ser utilizada como base de desarrollos comerciales.

### A. SIE: Plataforma abierta para el desarrollo de sistemas embebidos

En el mercado existe una gran variedad de plataformas que pueden ser utilizadas en el estudio de sistemas embebidos, sin embargo, no todas son adecuadas para la implementación del método propuesto ya que se requiere: acceso a los esquemáticos y a los archivos de fabricación del PCB con posibilidad de modificación; acceso a la documentación completa del proceso de fabricación; acceso a la cadena de producción; utilización de herramientas abiertas para su programación; un PLD para la implementación de tareas HW; un procesador para la implementación de tareas SW; un canal de comunicación entre el procesador y el PLD; y una comunidad que desarrolle aplicaciones para dicha plataforma y que proporcione medios para el intercambio de información a través de listas de correo y wikis. Para cumplir con los anteriores requerimientos se creó la plataforma SIE, la cual hace parte de los recursos del proyecto linuxencaja [8], el cual busca definir el concepto hardware copyleft basándose en el movimiento de software libre y código abierto (FOSS [9]).

### 1) Hardware copyleft

El proyecto SIE [10] fué creado para satisfacer las necesidades de los desarrolladores de hardware permitiendo la creación de aplicaciones comerciales bajo la licencia Creative Commons BY - SA [11] la que permite la distribución y modificación del diseño incluso para aplicaciones comerciales, con el único requisito de que los productos derivados deben tener la misma licencia y deben dar crédito al autor del trabajo original. Al ser inspirado en el movimiento FOSS, los dispositivos *hardware copyleft* comparten la misma filosofía [9], y son el complemento perfecto del software libre. Para que un dispositivo HW sea reproducible y modificable es necesario: suministrar los archivos para la fabricación, es decir, los esquemáticos y los archivos de la placa de circuito impreso (preferiblemente para herramientas abiertas como Kicad o geda); la cadena de herramientas de compilación y depuración para desarrollo de aplicaciones; el código fuente de: el programa que inicializa la plataforma (bootloader), la herramienta que carga dicho programa en la memoria no volátil (usbboot), el sistema de archivos y aplicaciones (openwrt); documentación completa

que indique como fue diseñada, construida, como utilizarla, desarrollar aplicaciones y tutoriales que expliquen el funcionamiento de los diferentes componentes [12], [13]. Adicionalmente, se debe contar con la posibilidad de fabricación y montaje, lo que constituye la principal diferencia entre el software y el hardware libre.

### 2) Arquitectura

La Figura 2 muestra el diagrama de bloques de la plataforma SIE, en ella encontramos un procesador que posee periféricos para comunicación serial (UART), memorias micro-SD, un puerto I2C, controlar un LCD a color de 3 pulgadas, 2 entradas y salidas de audio stereo, 2 entradas análogas; una FPGA que proporciona 25 señales de entrada/salida digitales de propósito general (GPIOs) y controla un conversor análogo digital de 8 canales. Existen tres canales de comunicación entre la FPGA y el procesador: uno para controlar el puerto JTAG, lo que permite la configuración de la FPGA desde el procesador (eliminando la necesidad de cables de programación); otro que proporciona el bus de datos, dirección y control para comunicarse con las tareas HW o periféricos implementadas en la FPGA y un puerto serial utilizado para depuración de softcores implementados en la FPGA. El procesador utilizado es un Ingenic JZ4725 (MIPS) corriendo a 400MHz, se dispone de una memoria NAND de 2GB para almacenamiento de datos y programas, así como de una memoria SDRAM de 32 MB, lo que permite la ejecución de una gran variedad de aplicaciones Linux.

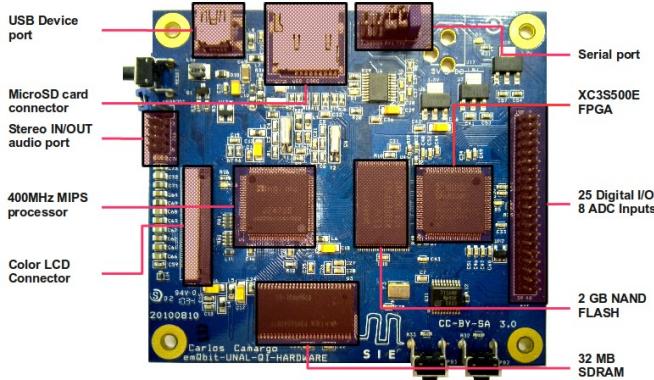


Fig. 2. Estructura de la plataforma de desarrollo SIE

### 3) Comunicaciones:

SIE proporciona un canal de comunicación y alimentación a través del puerto USB-device, y es configurado para ser utilizado como una interfaz de red (usb0), permitiendo la transferencia de archivos y ejecución de una consola remota utilizando el protocolo ssh; este canal de comunicación también se utiliza para programar la memoria NAND no volátil, por lo que para realizar la programación completa de los componentes de la plataforma solo es necesario un cable USB.

### B. Curso básico

En el primer curso de este programa académico, se trabajarán la mayoría de los niveles de abstracción del dominio funcional; partiendo de las especificaciones funcionales se generará un modelo del sistema utilizando

algoritmos que describan el comportamiento de las diferentes tareas que implementan el sistema (bloques funcionales). Estos bloques serán implementados, en dispositivos lógicos programables como FPGAs o CPLDs. A partir de estos algoritmos se identifican las operaciones básicas aritméticas y lógicas que modifican los datos asociados a cada función para generar el camino de datos (datapath); el cuál, adicionalmente proporciona señales que controlan el instante en el que se ejecutan las operaciones, dichas señales deben ser generadas por un módulo que implementa la funcionalidad deseada (máquina de control); estos dos módulos se implementarán con bloques lógicos básicos y máquinas de estados finitos utilizando lenguaje de descripción de hardware (VHDL, verilog). Estas descripciones son la entrada a herramientas que realizarán la transición al dominio estructural generando las compuertas lógicas, flip-Flops y las interconexiones que implementen la funcionalidad requerida. Durante el proceso es necesario realizar simulaciones (utilizando icarus verilog y ghdl) que permitan comprobar el cumplimiento de las especificaciones iniciales, si no se cumple alguna de ellas se debe volver a repetir el proceso.

Durante este curso se estudiarán los conceptos básicos de los sistemas digitales como: sistemas numéricos, operaciones aritméticas y lógicas, lógica combinatoria y secuencial. Se utilizarán lenguajes de descripción de hardware como VHDL y verilog como entrada de diseño a herramientas que realizan la síntesis digital. Para evitar crear dependencia, se enseñará la forma de adecuada de implementar código re-utilizable que no utilice componentes específicos de un determinado fabricante.

### 1) Diseño de aplicaciones utilizando HDL y PLDs

Para generar las habilidades necesarias para concebir, diseñar, implementar y operar sistemas digitales, se realizarán prácticas sencillas que ayuden al estudiante a entender los mecanismos de implementación de tareas HW en un dispositivo lógico programable utilizando la plataforma de desarrollo SIE; la FPGA solo controla un conversor análogo digital serial de 8 canales, y proporciona 25 GPIOs, esto se hizo de forma intencional para que los estudiantes se vean forzados a realizar las conexiones eléctricas de los dispositivos externos a sus aplicaciones; las plataformas comerciales proporcionan una gran variedad de dispositivos conectados a los PLDs, lo que no es muy recomendable ya que el estudiante no aprende a leer la hoja de especificaciones del fabricante de un determinado dispositivo para determinar su forma de conexión, y/o las condiciones que se deben tener en cuenta para su correcto funcionamiento; afectando la generación de habilidades necesarias para la elección de componentes, realización y lectura de esquemáticos y diseño de layouts. El procesador de la plataforma SIE será utilizado como camino de configuración de la FPGA; el archivo de configuración será descargado al sistema de archivos de la plataforma SIE utilizando el protocolo ssh y la interfaz de red USB; un programa en espacio de usuario se encarga de configurar la FPGA con el archivo deseado. Adicionalmente, existe una aplicación basada en el proyecto *urjtag* ejecutándose en el procesador que permite la generación de vectores de prueba y recolección de resultados utilizando el

puerto JTAG [14], lo que permite que el estudiante pueda probar su circuito a baja frecuencia sin instrumentos de medición adicionales. En la Figura 3 se muestra el flujo de diseño utilizado en este curso.

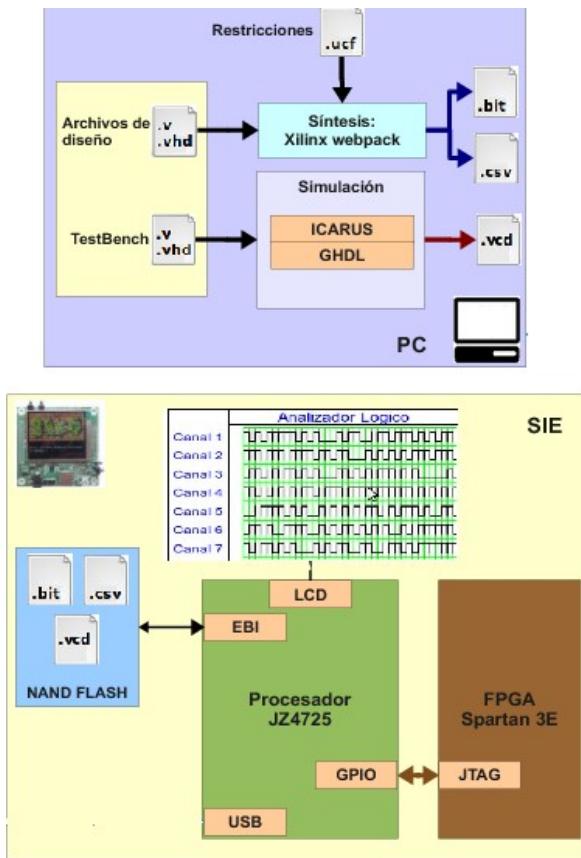


Figura 3. Flujo de diseño hardware utilizando SIE.

### C. Arquitectura de Computadores

En este curso se trabajará en el dominio estructural comenzando desde los componentes básicos de una CPU hasta llegar a la arquitectura de un sistema sobre silicio (SoC), esto con el fin de conocer y entender la arquitectura y funcionamiento de los dispositivos en los que se ejecutan las tareas software. Se utilizarán periféricos conectados a través de buses a la CPU para implementar tareas hardware. Se realizará la implementación de un Sistema sobre silicio (SoC) y se trabajará con herramientas de libre distribución (cadena de herramientas GNU [1]) para programar aplicaciones que involucren el uso de tareas HW y SW.

#### 1) Arquitectura de una CPU y tareas SW

Utilizando procesadores softcore se estudiarán los componentes básicos de la CPU, el camino de datos: banco de registros, bloques aritméticos, lógicos y buses internos, se analizarán las diferentes instrucciones que proporciona la arquitectura bajo estudio y se analizará el funcionamiento de la máquina de control, identificando los componentes que permiten el almacenamiento y ejecución secuencial de instrucciones definidas por el usuario. En este punto se introducirán los conceptos de llamado a funciones, atención de interrupciones, direccionamiento directo e indirecto y acceso a memoria externa. Para este estudio, se utilizarán

procesadores implementados en lenguajes de descripción de hardware que cuenten con herramientas de programación de bajo (assembler) y alto nivel (C, C++), con el fin de realizar simulaciones, aplicaciones y modificaciones. Al finalizar el curso se pretende que el estudiante entienda las diferencias entre tareas software y tareas hardware y podrá realizar experimentos que le permitan comparar las características de ambas implementaciones. En la actualidad se está trabajando con los SoC: plasma basado en un procesador MIPS; MICO 32 de lattice; y openrisc de OpenCores.

Se utilizarán los periféricos para la implementación de tareas HW y se estudiarán las diferentes formas que existen para comunicarse con las tareas SW (buses, interrupciones, polling) presentando los criterios de selección para la implementación de tareas SW y HW. Se introducirá el concepto de mapa de memoria, decodificador de direcciones, memorias volátiles (ejecución de programas y de uso general) y memorias no volátiles (almacenamiento de programas). Se introducirá el concepto de bootloader y su uso en la carga de aplicaciones a las memorias volátiles internas y externas del SoC. Y finalmente se mostrarán los diferentes métodos existentes para programar las memorias no volátiles.

Utilizando la cadena de herramientas GNU, se realizará el flujo de desarrollo para tareas SW desde la entrada de diseño utilizando el lenguaje C, hasta la generación del archivo binario que contiene las instrucciones (para la CPU en estudio) que implementan dicha tarea. Utilizando herramientas propias se generarán los archivos para inicializar la memoria de programa (implementada en la FPGA) con estas instrucciones.

#### 2) Co-diseño HW- SW

Tanto la CPU, los periféricos y las memorias estarán implementados en la FPGA de SIE. Durante todo el periodo académico se desarrollará un proyecto de complejidad media que involucre el uso de tareas HW y SW; para esto se formarán equipos de trabajo de 3 personas que deben hacer una propuesta inicial (concepción y especificaciones), realizar la descripción funcional del sistema utilizando algoritmos (diseño y modelo del sistema), realizar el particionamiento en funciones HW y SW, implementar estas funciones y diseñar una placa de circuito impreso (utilizando kicad) con lo necesario para implementar la funcionalidad requerida (implementación). Se realizarán entregas periódicas para verificar el cumplimiento del cronograma propuesto por el equipo de trabajo y el proceso de diseño debe ser documentado en la página wiki del curso. Esta actividad generará habilidades de trabajo en equipo, elaboración de esquemáticos, fabricación de PCBs, escritura de documentos técnicos e implementación de sistemas digitales.

SIE permite conectar de forma fácil dos GPIOs de la FPGA a las señales de transmisión y recepción del procesador, lo que permite la comunicación serial entre el procesador implementado en la FPGA y el procesador MIPS, proporcionando un canal de depuración para las aplicaciones implementadas en la FPGA, eliminando la necesidad de equipo adicional. En la Figura 4, se muestra el flujo de diseño utilizado en este curso.

## D. Sistemas Embebidos

SIE posee un SoC basado en un procesador MIPS equipado con los recursos necesarios para ejecutar programas Linux; en este curso se estudiará la arquitectura de los SoC comerciales, las herramientas de programación abiertas disponibles (cadena de herramientas GNU, librerías GNU como libQT), cargadores (u-boot), el sistema operativo Linux,

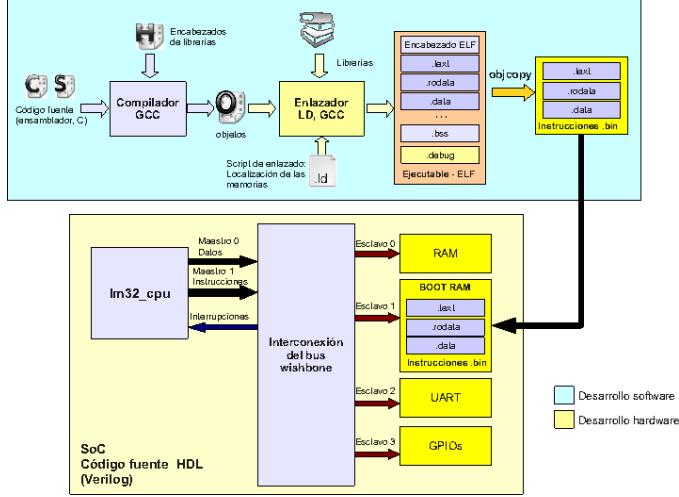


Figura 4. Flujo de diseño Hardware – Software.

drivers de periféricos, distribuciones para sistemas embebidos (openwrt, openembedded, debian). Con esto el estudiante

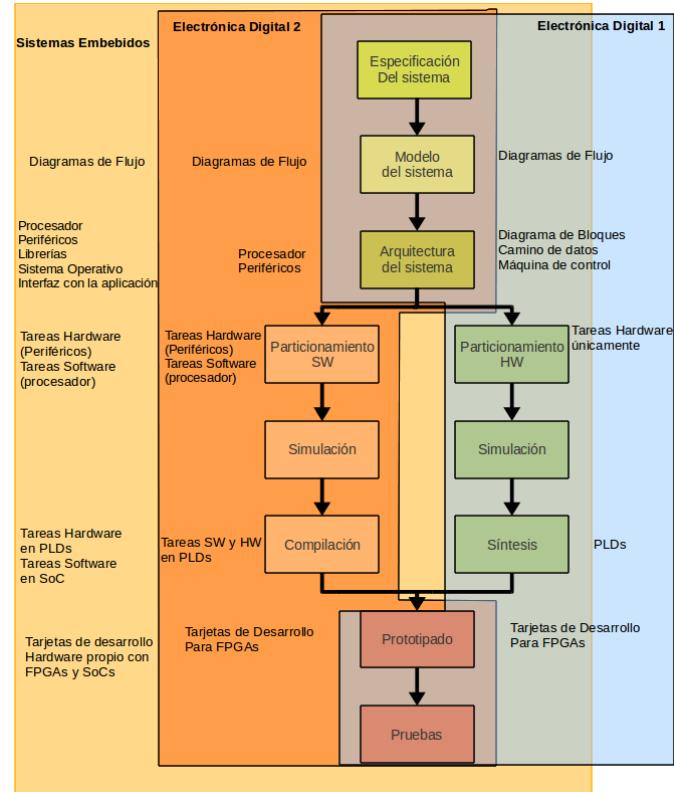


Fig. 5. Metodología de Diseño para el área de Sistemas Digitales

estará en capacidad de crear aplicaciones que utilizan la gran variedad de librerías disponibles en el proyecto FOSS, las mismas que utilizan Nokia, Motorola, Dell, Sony.

### 1) Creación de periféricos y drivers

Además de cubrir el tema de programación de SoC utilizando GNU/Linux, es necesario que se entienda no solo el funcionamiento del sistema operativo Linux, sino como este se comunica y controla los periféricos (tareas HW); para esto, se implementarán periféricos en la FPGA y se estudiará la forma de acceder a ellos utilizando el protocolo establecido por el kernel.

### 2) Concepción, Diseño, Implementación y Operación de Sistemas Embebidos

Durante el periodo académico los estudiantes deben diseñar, implementar y operar un sistema embebido concebido por un equipo de trabajo de tres personas, utilizando como base la plataforma SIE, diseñarán una tarjeta hija (utilizando kicad) que implemente la funcionalidad deseada, todos los proyectos deben integrar tareas HW en la FPGA con módulos del kernel para su control. Los proyectos realizados hasta el momento pueden encontrarse en la página del proyecto.

### E. Comunidad hardware copyleft

Una parte importante de este método de enseñanza es la filosofía del proyecto hardware copyleft, por esta razón, cada grupo debe hacer un aporte, suministrando la información completa del proceso de desarrollo, los archivos necesarios para replicar y/o modificarlo, esto es una consecuencia de la licencia CC-BY-SA. La experiencia del proyecto FOSS indica muchos miembros de estas comunidades ingresan para suplir necesidades, pero muchos de ellos continúan creando código y prestando servicios a la comunidad porque disfrutan programar. Estos aficionados realizan un papel muy importante dentro de la comunidad encargándose de tareas como mejora de la plataforma tecnológica, re-escribiendo secciones de código, documentándolo, respondiendo preguntas, preservando o mejorando la arquitectura [15]. Las actividades de documentación además de contribuir a mejorar las habilidades de escritura de reportes técnicos ayudan a formar una comunidad que contribuye al crecimiento del proyecto copyleft hardware, los estudiantes ingresan a las listas de desarrolladores aprendiendo a utilizar una herramienta muy poderosa en la que pueden compartir sus inquietudes con miembros más experimentados y mientras participan ayudan a crear un banco de preguntas que pueden ser útiles para futuros miembros. Adicionalmente se obliga a expresarse en un idioma diferente.

Crear estos hábitos ayuda a que los jóvenes sean conscientes de su papel dentro de la comunidad y piensen que sus acciones pueden ayudarla o perjudicarla, los proyectos realizados por ellos podrán ser parte de los recursos de la comunidad (si la calidad del trabajo lo amerita) y pueden ser la continuación de un esfuerzo prolongado o el punto de partida de un nuevo conocimiento; la licencia CC-BY-SA garantiza que todos los trabajos derivados de este recurso serán parte del mismo, lo que garantiza su crecimiento, la labor de los estudiantes es vital para el uso del recurso común

y puede crear miembros que en un futuro formularán políticas y reglas de uso del recurso.

Por otro lado, participar en este tipo de proyectos permite crear reputación, la cual puede ser útil para establecer relaciones profesionales, de negocios o personales. El entorno académico es ideal para atraer nuevos miembros a la comunidad hardware copyleft, ya que se trabaja con jóvenes con deseos de ser parte de un grupo y de adquirir conocimientos. Desde el punto de vista comercial este recurso es muy atractivo ya que permite ahorrar mucho tiempo, esfuerzo y dinero para la creación de nuevos productos. Por otro lado, el concepto de hardware copyleft es una herramienta poderosa para transferir tecnología y conocimientos a los países en vía de desarrollo donde la plataforma tecnológica no se lo suficientemente desarrollada.

### III. HABILIDADES CDIO

La iniciativa CDIO (<http://www.cdio.org>) ha sido desarrollada por el MIT con ayuda de académicos, industriales, ingenieros y estudiantes (CDIO, 2009) como respuesta a los diferentes caminos que están tomando la educación de la ingeniería y las demandas del mundo real. La iniciativa CDIO se basa en la suposición de que los egresados de los centros de formación en ingeniería deben ser capaces de: **Concebir, Diseñar, Implementar y Operar** sistemas funcionales en el mundo real. La frase *en el mundo real* resalta la importancia de trabajar en la solución de problemas que pueden encontrarse en el ejercicio profesional, lo que es muy difícil de determinar cuando los docentes no han tenido contacto con la industria. La figura 6. muestra las habilidades que se pretenden generar al aplicar el presente método.

HABILIDADES CDIO	Nivel 1		
	E. Digital I	E. Digital II	Sist. Emb.
<b>Contexto Externo, Social, Económico y Ambiental</b>			IEU
22 Rol y responsabilidad de los Ingenieros			IEU
23 Impacto sobre la sociedad y el medio ambiente			IEU
24 Cuestiones y valores actuales			IEU
44 Sostenibilidad y necesidad de un desarrollo sostenible	IE	IE	IE
<b>Empresa y contexto empresarial</b>	EU		
25 Interés en la empresa, metas y objetivos	I		
26 Espíritu Empresarial Técnico	I		
27 Trabajo exitoso en organizaciones	I		
45 Finanzas y Economía de los Proyectos de Ingeniería	IE	IE	IE
<b>Concepción y Administración de Sistemas en Ingeniería.</b>	IEU		
28 Entender las necesidades y establecer las metas	IEU	EU	U
29 Definir la función, concepto y arquitectura	IEU	EU	U
<b>Diseño</b>	IEU		
30 Proceso de Diseño	IEU	EU	U
31 Fases del proceso de Diseño y enfoques	IEU	EU	U
32 Utilización de conocimiento científico en el diseño	IEU	EU	U
33 Diseño específico	IEU	EU	U
34 Diseño multi-disciplinario	I	E	U
<b>Implementación</b>	EU		
35 Proceso de Fabricación Hardware	IEU	EU	U
36 Proceso de Implementación de Software	I	EU	U
37 Integración Software - Hardware	I	EU	U
38 Pruebas, verificación, validación y certificación	IE	EU	U

Figura 6. Habilidades CDIO desarrolladas con la aplicación del método propuesto.

### IV. CONCLUSIONES

El hardware copyleft es una herramienta poderosa para la creación de habilidades necesarias para concebir, diseñar, implementar y operar sistemas digitales, ya que proporciona la información necesaria para entender el ciclo completo de diseño, (lo cual no es posible obtener cuando se trabaja con plataformas de desarrollo comerciales); proporcionando

información detallada sobre el proceso de diseño de plataformas abiertas, que pueden ser utilizadas como referencia para generar nuevos productos comerciales; el acceso a aplicaciones software que permiten la creación de aplicaciones; un canal de comunicación que permite utilizar a la industria manufacturera asiática para la producción en masa; conocimiento de los procesos de fabricación y producción.

Las actividades propuestas en las tres asignaturas del área tienen como objetivo generar en el estudiante las habilidades necesarias que le permitan diseñar sistemas digitales con grado de complejidad creciente, hasta llegar a un sistema que puede ser comercializable y satisfacer una necesidad de una determinada comunidad, con esto, se evita que el último paso en el proceso de enseñanza sea la simulación; se ilustra el proceso que debe seguirse para que un prototipo se convierta en un producto comercial, lo que contribuirá con la creación de nuevos productos y la generación de empleo.

La utilización de herramientas de bajo nivel permite que el estudiante conozca y controle los diferentes pasos de la metodología de diseño y sea capaz de ajustarlas para diferentes situaciones, esto hace que se adquiera un conocimiento sobre la tecnología sin crear dependencia hacia las herramientas comerciales que realizan la mayoría de los pasos de la metodología de forma automática.

### REFERENCIAS

- [1] R. M. Stallman. The GNU Operating System and the Free Software Movement Voices from the Open Source Revolution. O'Reilly and Associates, 1999.
- [2] A. Grove. How America Can Create Jobs. <http://www.businessweek.com/magazine/content/1028/b4186048358596.htm>, May 2010.
- [3] Jon Hall. POR GRANDES QUE SEAN...: ASEGURE EL FUTURO DE SU NEGOCIO. Linux magazine, ISSN 1576-4079(58):92, 2009.
- [4] H. Mitsui, H. Kambe, and H. Koizumi. Use of Student Experiments for Teaching Embedded Software Development Including HW/SW Co-Design. IEEE TRANSACTIONS ON EDUCATION., 52(3), August 2009.
- [5] A. Gerstlauer, D. Gajski., . Technical Report CECS-02-17, and 2002. CECS, UC Irvine. System-level abstraction semantics, Technical Report CECS-02-17. Technical report, CECS, UC Irvine, 2002.
- [6] Gajski D.D., Abdi S., Gerstlauer A., and Schirner G. Embedded System Design: Modeling, Synthesis, Verification. Springer, 2009.
- [7] Luis Alejandro Cortés. Verification and Scheduling Techniques for Real-Time Embedded Systems. PhD thesis, Linköpings universitet Institute of Technology, 2005.
- [8] Proyecto linuxencja URL: <http://en.qi-hardware.com/>.
- [9] R. Stallman. Philosophy of the GNU project. URL: <http://www.gnu.org/philosophy/>, 2007.
- [10] C. Camargo. SIE: Plataforma Hardware copyleft para la enseñanza de Sistemas Digitales. XVII Workshop de Iberchip, Bogotá 2011.
- [11] Creative Commons. Licencias Creative Commons. URL: <http://creativecommons.org/licenses/>, 2004.
- [12] C. Camargo. Metodología Para la Transferencia Tecnológica en la Industria Electrónica Basada en Software Libre y Hardware Copyleft. XVII Workshop de Iberchip, 2011.
- [13] C. Camargo. Proyecto SIE. URL: <http://linuxencaja.com/wiki/SIE> 2011
- [14] Texas Instruments. IEEE Std 1149.1 (JTAG) Testability. 1997 Semiconductor Group, 1996.
- [15] S. Shah. Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. Management Science, July 2006.



**Carlos Camargo** Candidato a Doctor en ingeniería Eléctrica de la Universidad Nacional de Colombia. Docente del Departamento de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Colombia . Su

área de investigación es el diseño digital utilizando sistemas embebidos que ejecutan aplicaciones para el sistema operativo Linux.

**Hardware copyleft como Herramienta para la Enseñanza de Sistemas Embebidos**

Congreso Argentino de Sistemas Embebidos CASE 2011, Buenos Aires Argentina ISBN 978-987-9374-69-6

# El papel del Hardware *copyleft* en la Enseñanza de Sistemas Embebidos

Carlos I. Camargo Bareño  
Universidad Nacional de Colombia,  
Email: cicamargoba@unal.edu.co

**Abstract**—El gran avance de las técnicas de fabricación de Circuitos Integrados ha permitido que los sistemas embebidos sean parte fundamental de nuestras vidas, aún sin darnos cuenta diariamente interactuamos con decenas de ellos. Esto unido a la disponibilidad de herramientas software de desarrollo gratuitas abre grandes posibilidades comerciales para países en vía de desarrollo ya que no son necesarias grandes inversiones de capital para la concepción, diseño, y fabricación de estos sistemas. Sin embargo, en la actualidad muy pocas universidades ofrecen cursos que permitan crear las habilidades necesarias para la realización de un producto comercializable, lo que se traduce en un abandono de la producción local y el aumento de la dependencia con la industria manufacturera asiática, por otro lado, las herramientas utilizadas en la actualidad (tanto SW como HW) proporcionan un nivel de abstracción relativamente alto impidiendo que el estudiante entienda el funcionamiento global de un sistema digital, lo que le impide generar habilidades (específicamente las relacionadas con la concepción e implementación) necesarias para realizar el proceso completo. En este artículo se presenta una metodología para la enseñanza de diseño de sistemas embebidos utilizando herramientas hardware y software abiertas que ayuda a resolver los problemas mencionados anteriormente.

**Index Terms**—Sistemas Embebidos, educación en ingeniería, hardware copyleft.

## I. INTRODUCCIÓN

El mercado de los sistemas embebidos continúa en aumento y su campo de acción se ha extendido en casi todas las actividades humanas. Según BBC, inc. el mercado para el software embebido creció de \$1.6 billones a \$3.5 billones en 2009, con una tasa de crecimiento anual (AAGR) del 16%, mientras la tasa de crecimiento para las tarjetas embebidas es del 10%; según *Venture Development Corporation (VDC)* más de un billón de dispositivos embebidos fueron vendidos en el 2004,. De acuerdo con VDC el porcentaje de dispositivos basados en sistemas operativos comerciales tiende a disminuir del 43.1% en 2001 a 37.1% en 2004, esta tendencia se debe a la utilización de herramientas de libre distribución GNU/Linux [1].

En la actualidad estamos presenciando una tendencia global a delegar las tareas de manufactura de sistemas digitales a países asiáticos, donde la mano de obra calificada es abundante y barata; se presentan casos donde los creadores de una determinada tecnología no la desarrollan y dejan que estos países se beneficien de sus descubrimientos [2] reduciendo de forma considerable la producción. Esta situación se agrava a medida que las grandes empresas manufactureras asiáticas como Foxconn capturan la producción de los grandes diseñadores

como Apple, Nokia, DELL, HP y Microsoft, lo que genera el cierre de empresas manufactureras a lo largo del mundo, con la consecuencia de pérdida de empleo masivo. En la actualidad según *Bureau of Labor Statistics* y *Thomson Financial Extel Company Report* Foxconn emplea a más personas que Apple, Dell, Microsoft, HP, Intel y Sony combinados; esta situación es más grave en países en vía de desarrollo, donde no existe la plataforma tecnológica para diseñar dispositivos digitales, y son importadores de tecnología sin la capacidad de generar productos que satisfagan necesidades locales. Lo anterior lleva a preguntarse por la función y la situación de un profesional en áreas afines a la ingeniería electrónica en países donde no existe la capacidad de concepción y diseño.

La tendencia moderna en los programas académicos a la utilización de herramientas de alto nivel para la enseñanza en áreas afines al desarrollo de dispositivos digitales [3] ocasiona que los profesionales no adquieran las habilidades necesarias para completar la cadena concepción - diseño - implementación y operación, en la mayoría de los casos se generan habilidades para la concepción y el diseño a alto nivel y dejan los otros pasos en manos de herramientas especializadas y/o a empresas asiáticas. Esta situación resulta la más atractiva desde el punto de vista económico, ya que no es necesario adquirir maquinaria costosa ni contratar personal calificado para operarlas; sin embargo, limita la generación de empleo local a personas con un nivel de formación alto [2] generando desempleo en las personas menos capacitadas. Según John Hall presidente y CEO de Linux International “algunas facultades preparan a la gente en el uso de productos en vez de tecnologías de nivel básico” [3]. Esta situación unida al abandono de la implementación hace que la dependencia con las empresas manufactureras asiáticas aumente cada vez más.

Según el ex-director ejecutivo de Intel Andy Grove [2] la solución está en hacer de la creación de empleo la política económica gubernamental más importante y hacer que las demás giren en torno a ella. Además, es necesario volver a la producción interna con el fin de generar nuevos empleos, y volver a adoptar medidas que protejan la producción interna de los productos asiáticos. Sin embargo, para lograrlo es necesario crear en los profesionales las habilidades para implementar productos comercializables.

En este artículo presentamos un programa académico basado en la utilización de software y hardware libre para el área de electrónica digital que desarrolla las habilidades nece-

sarias para Concebir, Diseñar Implementar y operar sistemas digitales.

#### A. Flujo de diseño de sistemas embebidos

Los Sistemas Embebidos son sistemas heterogéneos que contienen componentes Software (microcontroladore, microprocesadores y DSPs) y Hardware (funciones implementadas en Dispositivos lógicos programables PLDs); por este motivo, es necesario adquirir habilidades en la utilización de lenguajes de programación como C o C++ para implementar las funciones software y Verilog o VHDL para la implementación de las tareas hardware; adicionalmente, deben conocer las diferentes formas de comunicación entre estos dos tipos de funciones. Aunque en el mercado existen herramientas que permiten la entrada de diseño utilizando lenguajes de alto nivel como *SystemC* o *SpecC* y proporcionan el código para implementar las tareas software, hardware y su interfaz de comunicación; no es recomendable utilizarlas en el ciclo de formación básico ya que impide que se conozca el flujo de diseño completo, suministrando un nivel de abstracción en el cual no es necesario conocer la arquitectura de la plataforma utilizada para la implementación.

En la figura 1 se muestran los conceptos que deben dominar los diseñadores de sistemas embebidos, y las tareas que deben realizarse para la concepción, diseño e implementación de un sistema embebido. En gran parte de los programas académicos se estudian únicamente los temas relacionados con la concepción y diseño centrándose en las especificaciones funcionales del sistema, utilizando herramientas comerciales o COTS (Commercial off-the-shelf) para su implementación. Esta combinación genera dependencia e impide la generación de habilidades necesarias para implementar un sistema digital teniendo en cuenta restricciones económicas, físicas, eléctricas, ergonómicas, comerciales, etc. Nuestra propuesta se basa en la utilización de herramientas abiertas tanto hardware como software que permiten recorrer todo el proceso de concepción, diseño e implementación y obtener un entendimiento integral del proceso sin generar dependencia a productos comerciales.

1) *Dominios de Diseño y Niveles de abstracción*: Existen tres dominios en los que se puede describir un sistema digital [5] *Funcional*: Describe el comportamiento funcional y temporal del sistema *Estructural*: Describe su composición a partir de bloques básicos y *Físico* relacionado con la estructura física del sistema a nivel de circuito integrado o placa de circuito impreso [6]. Cada uno de estos dominios puede ser descrito utilizando diferentes niveles de abstracción; un nivel de abstracción alto permite el uso de lenguajes de alto nivel facilitando la entrada de diseño al extraer la funcionalidad de la parte física; por otro lado, los niveles de abstracción bajo utilizan bloques constructores elementales.

En el mercado existen herramientas que permiten entradas de diseño a nivel funcional utilizando especificaciones y algoritmos y de forma automática y optimizada generan representaciones en diferentes niveles de abstracción en los dominios estructural y físico. Es decir, a partir de las especificaciones

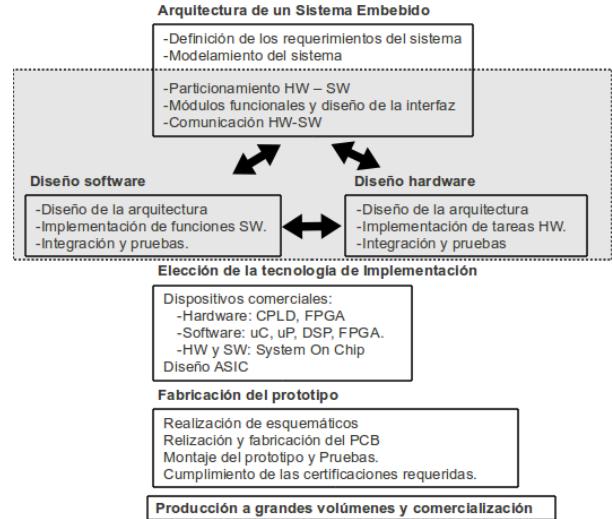


Fig. 1. Educación de sistemas embebidos. Tomada de: [4] y modificada

y algoritmos que indican el funcionamiento del sistema pueden generar archivos para la fabricación de un circuito integrado o archivos de configuración/programación que pueden ser utilizados en dispositivos comerciales. Desde el punto de vista comercial esto es muy útil ya que permite reducir el tiempo de diseño y los costos asociados. Sin embargo, presentan los inconvenientes mencionados anteriormente y por lo general son muy costosas.

## II. MÉTODO PARA LA ENSEÑANZA DE SISTEMAS EMBEBIDOS

Basándose en la metodología de diseño para sistemas embebidos [7], en los dominios de diseño y niveles de abstracción de Gajski-Kuhn, se realizó una división de temas que busca crear habilidades de forma gradual e incremental. En la Figura 4 podemos observar esta división y las herramientas que se utilizarán en cada curso, como herramienta de desarrollo hardware se utilizará una plataforma que proporcione los archivos y documentos necesarios para replicarla, modificarla y pueda ser utilizada como base de desarrollos comerciales.

### A. SIE: Plataforma abierta para el desarrollo de sistemas embebidos

En el mercado existe una gran variedad de plataformas que pueden ser utilizadas en el estudio de sistemas embebidos, sin embargo, no todas son adecuadas para la implementación del método que proponemos ya que se requiere: acceso a los esquemáticos y a los archivos de fabricación del PCB con posibilidad de modificación; acceso a la documentación completa del proceso de fabricación; acceso a la cadena de producción; utilización de herramientas abiertas para su programación; un PLD para la implementación de tareas HW; un procesador para la implementación de tareas SW; un canal de comunicación entre el procesador y el PLD; y una comunidad que desarrolle aplicaciones para dicha plataforma y que proporcione medios

para el intercambio de información a través de listas de correo y wikis.

Después de una búsqueda minuciosa no se encontraron plataformas que cumplieran con estas condiciones, en especial con las relacionadas con el proceso de diseño y de producción, esto es normal, ya que la mayoría de las empresas no quieren que se fabriquen sus plataformas y los proyectos individuales no poseen la infraestructura necesaria para la producción masiva. Por este motivo, se decidió crear una plataforma que cumpliera con los requerimientos (plataforma *SIE*), para ello se buscaron proyectos similares que permitieran su creación y que el producto creado sea una extensión de dicho proyecto. El proyecto Qi-Hardware [8] busca definir el concepto *copyleft hardware* basándose en el movimiento de software libre y código abierto (FOSS [9]) y proporciona un enlace con la industria manufacturera asiática.

1) *Hardware copyleft*: El proyecto SIE [10] fué creado para satisfacer las necesidades de los desarrolladores de hardware permitiendo la creación de aplicaciones comerciales bajo la licencia Creative Commons BY - SA [11] la que permite la distribución y modificación del diseño (incluso para aplicaciones comerciales), con el único requisito de que los productos derivados deben tener la misma licencia y deben dar crédito al autor del trabajo original. Lo que constituye la base de los productos *hardware copyleft*.

Al ser inspirado en el movimiento FOSS, los dispositivos *hardware copyleft* comparten la misma filosofía [9], y son el complemento perfecto del software libre. Para que un dispositivo HW sea reproducible y modificable es necesario: suministrar los archivos necesarios para la fabricación, es decir, los esquemáticos y los archivos de la placa de circuito impreso (preferiblemente para herramientas abiertas como Kicad o geda); la cadena de herramientas de compilación y depuración para desarrollo de aplicaciones; el código fuente de: el programa que inicializa la plataforma (*bootloader*), la herramienta que carga dicho programa en la memoria no volátil (*usbboot*), el sistema de archivos y aplicaciones (*openwrt*); documentación completa que indique como fué diseñada, construida, como utilizarla, desarrollar aplicaciones y tutoriales que expliquen el funcionamiento de los diferentes componentes. (esto puede ser descargado de [12] y [13]). Adicionalmente, se debe contar con la posibilidad de fabricación y montaje, lo que constituye la principal diferencia entre el software y el hardware libre.

2) *Arquitectura*: La Figura 2 muestra el diagrama de bloques de la plataforma SIE, en ella encontramos un procesador que posee periféricos para comunicación serial (UART), memorias micro-SD, un puerto I2C, controlar un LCD a color de 3 pulgadas, 2 entradas y salidas de audio stereo, 2 entradas analógicas; una FPGA que proporciona 25 señales de entrada/salida digitales de propósito general (GPIOs) y controla un conversor análogo digital de 8 canales. Existen tres canales de comunicación entre la FPGA y el procesador: uno para controlar el puerto JTAG, lo que permite la configuración de la FPGA desde el procesador (eliminando la necesidad de cables de programación); otro que proporciona el bus de

datos, dirección y control para comunicarse con las tareas HW o periféricos implementadas en la FPGA y un puerto serial utilizado para depuración de softcores implementados en la FPGA. El procesador utilizado es un Ingenic JZ4725 (MIPS) corriendo a 400MHz, se dispone de una memoria NAND de 2GB para almacenamiento de datos y programas, así como de una memoria SDRAM de 32 MB, lo que permite la ejecución de una gran variedad de aplicaciones Linux.

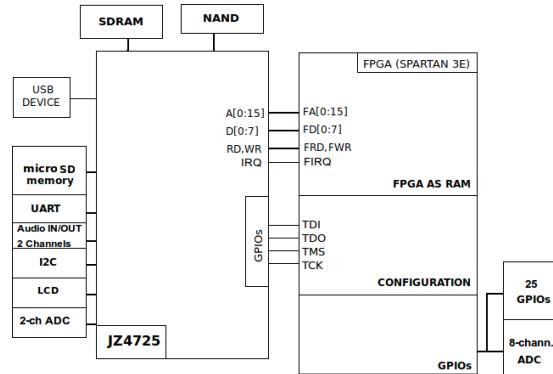


Fig. 2. Estructura de la plataforma de desarrollo SIE

3) *Comunicaciones*: SIE proporciona un canal de comunicación y alimentación a través del puerto USB-device, y es configurado para ser utilizado como una interfaz de red (*usb0*), permitiendo la transferencia de archivos y ejecución de una consola remota utilizando el protocolo ssh; este canal de comunicación también se utiliza para programar la memoria NAND no volátil, por lo que para realizar la programación completa de los componentes de la plataforma solo es necesario un cable USB.

4) *Especificaciones físicas*: Las dimensiones de SIE son 8cm de largo, 8 cm de ancho, 1cm de altura; su placa de circuito impreso es de dos capas, utiliza líneas de 8 mils, vías de 12 mils de diámetro, los componentes se encuentran en una sola capa y son TQFP o SMD, no posee componentes BGA o QFN lo que facilita el montaje manual. Todo esto hace que sea posible la reproducción y modificación de esta plataforma a un precio muy bajo. El costo de fabricación de esta tarjeta se estima en 70 usd para 50 unidades. La figura 3 muestra la apariencia de la plataforma de desarrollo SIE.

## B. Curso básico

Para el curso básico se trabajará la mayor parte de los niveles de abstracción del dominio funcional; partiendo de unas especificaciones funcionales se generará un modelo del sistema utilizando algoritmos que describan el comportamiento de las diferentes tareas que implementan el sistema (bloques funcionales). Estos bloques serán implementados, en dispositivos lógicos programables como FPGAs o CPLDs. A partir de estos algoritmos se identifican las operaciones básicas aritméticas y lógicas que modifican los datos asociados a cada función para generar el camino de datos (*datapath*;

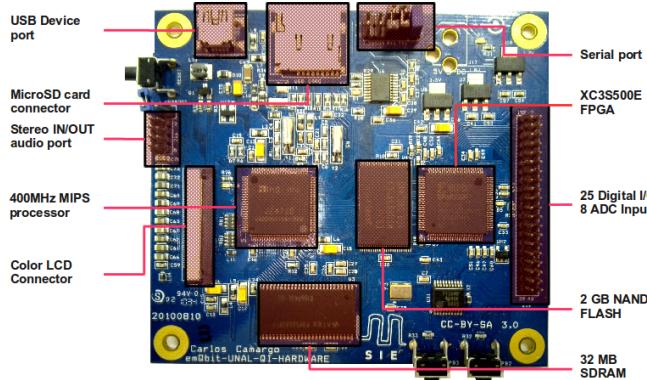


Fig. 3. Plataforma de desarrollo SIE

el datapath proporciona señales que controlan el instante en el que se ejecutan las operaciones soportadas, dichas señales deben ser generadas por un módulo diseñado para implementar el algoritmo deseado; estos dos módulos se implementaran con bloques lógicos básicos y máquinas de estados finitos utilizando lenguaje de descripción de hardware (VHDL, verilog estándard). Estas descripciones son la entrada a herramientas que realizarán la transición al dominio estructural generando las compuertas lógicas, flip-Flops y las interconexiones que implementen la funcionalidad requerida. Durante el proceso es necesario realizar simulaciones (utilizando *icarus verilog* y *ghdl*) que permitan comprobar el cumplimiento de las especificaciones iniciales, si no se cumple alguna de ellas se debe volver a repetir el proceso.

Durante el desarrollo del primer curso se estudiaran los conceptos básicos de los sistemas digitales como sistemas numéricos, operaciones aritméticas y lógicas, lógica combinatoria y secuencial. Se utilizaran lenguajes de descripción de hardware como VHDL y verilog como entrada de diseño a herramientas que realizan la síntesis digital. Para evitar crear dependencia, se enseñará la forma de adecuada de implementar código re-utilizable que no utilice componentes específicos de un determinado fabricante.

1) *Diseño de aplicaciones utilizando HDL y PLDs*: Para generar las habilidades necesarias para concebir, diseñar, implementar y operar sistemas digitales, se realizarán prácticas sencillas que ayuden al estudiante a entender los mecanismos de implementación de tareas HW en un dispositivo lógico programable utilizando la plataforma de desarrollo SIE; como puede verse en la figura 2 la FPGA solo controla un conversor analógico digital serial de 8 canales, y proporciona 25 GPIOs, esto se hizo de forma intencional para que los estudiantes se vean forzados a realizar las conexiones eléctricas de los dispositivos externos a sus aplicaciones; las plataformas comerciales proporcionan una gran variedad de dispositivos conectados a los PLDs, lo que no es muy recomendable ya que el estudiante no aprende a leer la hoja de especificaciones del fabricante de un determinado dispositivo para determinar su forma de conexión, y/o las condiciones que se deben

tener en cuenta para su correcto funcionamiento; afectando la generación de habilidades necesarias para la elección de componentes, realización y lectura de esquemáticos y diseño de layouts.

El procesador de la plataforma SIE será utilizado como camino de configuración de la FPGA; el archivo de configuración será descargado al sistema de archivos de la plataforma SIE utilizando el protocolo *ssh* y la interfaz de red USB; un programa en espacio de usuario se encarga de configurar la FPGA con el archivo deseado. Adicionalmente, existe una aplicación basada en el proyecto *urjtag* ejecutándose en el procesador que permite la generación de vectores de prueba y recolección de resultados utilizando el puerto JTAG [14], lo que permite que el estudiante pueda probar su circuito a baja frecuencia sin instrumentos de medición adicionales.

### C. Arquitectura de Computadores

En este curso se trabajará en el dominio estructural comenzando desde los componentes básicos de una CPU hasta llegar a la arquitectura de un sistema sobre silicio (SoC), esto con el fin de conocer y entender la arquitectura y funcionamiento de los dispositivos en los que se ejecutan las tareas software. Se utilizarán periféricos conectados a través de buses a la CPU para implementar tareas hardware. Se realizará la implementación de un Sistema sobre silicio (SoC) y se trabajará con herramientas de libre distribución (cadena de herramientas GNU [1]) para programar aplicaciones que involucren el uso de tareas HW y SW.

1) *Arquitectura de una CPU y tareas SW*: Utilizando procesadores *softcore* se estudiarán los componentes básicos de la CPU, el camino de datos: banco de registros, bloques aritméticos, lógicos y buses internos, se analizarán las diferentes instrucciones que proporciona la arquitectura bajo estudio y se analizará el funcionamiento de la máquina de control, identificando los componentes que permiten el almacenamiento y ejecución secuencial de instrucciones definidas por el usuario. En este punto se introducirán los conceptos de llamado a funciones, atención de interrupciones, direccionamiento directo e indirecto y acceso a memoria externa.

Para este estudio, se utilizarán procesadores implementados en lenguajes de descripción de hardware que cuenten con herramientas de programación de bajo (assembler) y alto nivel (C, C++), con el fin de realizar simulaciones, aplicaciones y modificaciones. Al finalizar el curso se pretende que el estudiante entienda las diferencias entre tareas software y tareas hardware y podrá realizar experimentos que le permitan comparar las características de ambas implementaciones. En la actualidad se está trabajando con los SoC *plasma* basado en un procesador MIPS, *MICO 32* de lattice, y *openrisc* de OpenCores, implementados en VHDL o Verilog.

Se utilizarán los periféricos para la implementación de tareas HW y se estudiarán las diferentes formas que existen para comunicarse con las tareas SW (buses, interrupciones, polling) presentando los criterios de selección para la implementación de tareas SW y HW. Se introducirá el concepto de mapa de memoria, decodificador de direcciones, memorias volátiles

(ejecución de programas y de uso general) y memorias no volátiles (almacenamiento de programas). Se introducirá el concepto de *bootloader* y su uso en la carga de aplicaciones a las memorias volátiles internas y externas del SoC. Y finalmente se mostrarán los diferentes métodos existentes para programar las memorias no volátiles.

Utilizando la cadena de herramientas GNU, se realizará el flujo de desarrollo para tareas SW desde la entrada de diseño utilizando el lenguaje C, hasta la generación del archivo binario que contiene las instrucciones (para la CPU en estudio) que implementan dicha tarea. Utilizando herramientas propias se generarán los archivos para inicializar la memoria de programa (implementada en la FPGA) con estas instrucciones.

**2) Diseño de aplicaciones que involucren co-diseño HW-SW:** Tanto la CPU, los periféricos, la memoria ram y la memoria de programa estarán implementados en la FPGA de SIE. Durante todo el periodo académico se desarrollará un proyecto de complejidad media que involucre el uso de tareas HW y SW, (en la página de la plataforma [13] se pueden observar los proyectos realizados hasta el momento); para esto se formarán equipos de trabajo de 3 personas que deben hacer una propuesta inicial (concepción y especificaciones), realizar la descripción funcional del sistema utilizando algoritmos (diseño y modelo del sistema), realizar el particionamiento en funciones HW y SW, implementar estas funciones y diseñar una placa de circuito impreso (utilizando *kicad*) con lo necesario para implementar la funcionalidad requerida (implementación). Se realizarán entregas periódicas para verificar el cumplimiento del cronograma propuesto por el equipo de trabajo y el proceso de diseño debe ser documentado en la página wiki del curso. Esta actividad generará habilidades de trabajo en equipo, elaboración de esquemáticos, fabricación de PCBs, escritura de documentos técnicos e implementación de sistemas digitales.

SIE permite conectar de forma fácil (usando dos jumpers) dos GPIOs de la FPGA a las señales de transmisión y recepción del procesador, lo que permite la comunicación serial entre el procesador implementado en la FPGA y el procesador MIPS, creando de esta forma un canal de depuración para las aplicaciones implementadas en la FPGA, eliminando la necesidad de equipo adicional.

#### D. Sistemas Embebidos

El tercer y último curso de la línea integra los conocimientos adquiridos en los cursos anteriores e introduce un elemento nuevo un SoC comercial, SIE posee un SoC basado en un procesador MIPS equipado con los recursos necesarios para ejecutar programas Linux; en este curso se estudiará la arquitectura de los SoC comerciales, las herramientas de programación abiertas disponibles (cadena de herramientas GNU, librerías GNU como libQT), cargadores de Linux (u-boot), el sistema operativo Linux, drivers de periféricos, distribuciones para sistemas embebidos (openwrt, openembedded, debian). Con esto el estudiante estará en capacidad de crear aplicaciones que utilizan la gran variedad de librerías disponibles en

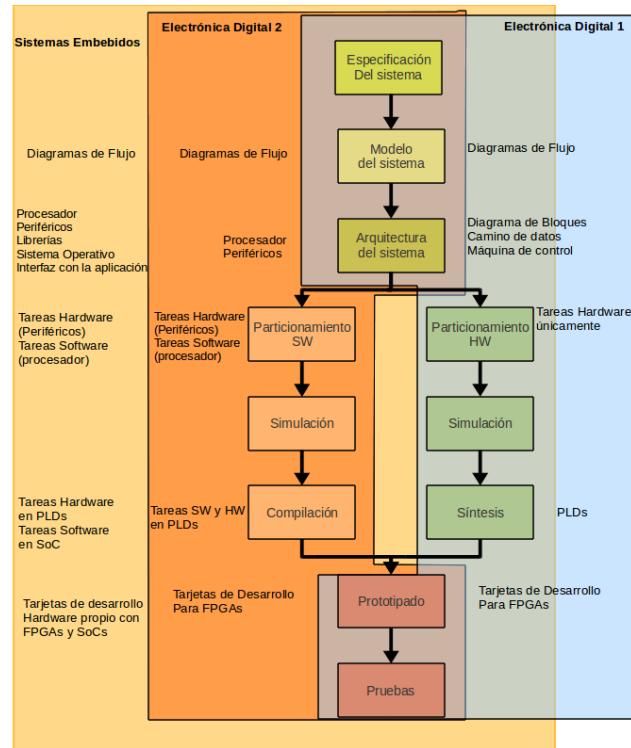


Fig. 4. Metodología de Diseño para el área de Sistemas Digitales

el proyecto FOSS, las mismas que utilizan Nokia, Motorola, Dell, Sony.

**1) Creación de periféricos y drivers :** Además de cubrir el tema de programación de SoC utilizando GNU/Linux, es necesario que se entienda no solo el funcionamiento del sistema operativo Linux, sino como este se comunica y controla los periféricos (tareas HW); para esto, se implementarán periféricos en la FPGA y se estudiará la forma de acceder a ellos utilizando el protocolo establecido por el kernel.

**2) Concepción, Diseño, Implementación y Operación de Sistemas Embebidos:** Durante el periodo académico los estudiantes deben diseñar, implementar y operar un sistema embebido concebido por un equipo de trabajo de tres personas, utilizando como base la plataforma SIE, diseñarán una tarjeta hija (utilizando kicad) que implemente la funcionalidad deseada, todos los proyectos deben integrar tareas HW en la FPGA con módulos del kernel para su control. Los proyectos realizados hasta el momento pueden encontrarse en la página del proyecto.

#### E. Comunidad hardware copyleft

Una parte importante de este método de enseñanza es la filosofía del proyecto hardware copyleft, por esta razón, cada grupo debe hacer un aporte, suministrando la información completa del proceso de desarrollo, los archivos necesarios para replicar y/o modificarlo, esto es una consecuencia de la licencia CC-BY-SA.

La experiencia del proyecto FOSS indica muchos miembros de estas comunidades ingresan para suplir necesidades, pero muchos de ellos continuan creando código y prestando servicios a la comunidad porque disfrutan programar. Estos *aficionados* realizan un papel muy importante dentro de la comunidad encargándose de tareas como mejora de la plataforma tecnológica, re-escribiendo secciones de código, documentandolo, respondiendo preguntas, preservando o mejorando la arquitectura [15]. Las actividades de documentación además de contribuir a mejorar las habilidades de escritura de reportes técnicos ayudan a formar una comunidad que contribuye al crecimiento del proyecto copylef harware, los estudiantes ingresan a las listas de desarrolladores aprendiendo a utilizar una herramienta muy poderosa en la que pueden compartir sus inquietudes con miembros más experimentados y mientras participan ayudan a crear un banco de preguntas que pueden ser útiles para futuros miembros. Adicionalmente se obliga a expresarse en un idioma diferente.

Crear estos hábitos ayuda a que los jóvenes sean conscientes de su papel dentro de la comunidad y piensen que sus acciones pueden ayudarla o perjudicarla, los proyectos realizados por ellos podrán ser parte de los recursos de la comunidad (si la calidad del trabajo lo amerita) y pueden ser la continuación de un esfuerzo prolongado o el punto de partida de un nuevo conocimiento; la licencia CC-BY-SA garantiza que todos los trabajos derivados de este recurso serán parte del mismo, lo que garantiza su crecimiento, la labor de los estudiantes es vital para el uso del recurso común y puede crear miembros que en un futuro formularán políticas y reglas de uso del recurso. Por otro lado, participar en este tipo de proyectos permite crear reputación, la cual puede ser útil para establecer relaciones profesionales, de negocios o personales. El entorno académico es ideal para atraer nuevos miembros a la comunidad hardware copyleft, ya que se trabaja con jóvenes con deseos de ser parte de un grupo y de adquirir conocimientos. Desde el punto de vista comercial este recurso es muy atractivo ya que permite ahorrar mucho tiempo, esfuerzo y dinero para la creación de nuevos productos. Por otro lado, el concepto de hardware copyleft es una herramienta poderosa para transferir tecnología y conocimientos a los países en vía de desarrollo donde la plataforma tecnológica no se lo suficientemente desarrollada.

### III. CONCLUSIONES

El hardware copyleft es una herramienta poderosa para la creación de habilidades necesarias para concebir, diseñar, implementar y operar sistemas digitales, ya que proporciona la información necesaria para entender el ciclo completo de diseño, (lo cual no es posible obtener cuando se trabaja con plataformas de desarrollo comerciales); proporcionando información detallada sobre el proceso de diseño de plataformas abiertas, que pueden ser utilizadas como referencia para generar nuevos productos comerciales; el acceso a aplicaciones software que permiten la creación de aplicaciones; un canal de comunicación que permite utilizar a la industria manufacturera asiática para la producción en masa; conocimiento de los procesos de fabricación y producción.

Las actividades propuestas en las tres asignaturas del área tienen como objetivo generar en el estudiante las habilidades necesarias que le permitan diseñar sistemas digitales con grado de complejidad creciente, hasta llegar a un sistema que puede ser comercializable y satisface una necesidad de una determinada comunidad, con esto, se evita que el último paso en el proceso de enseñanza sea la simulación; se ilustra el proceso que debe seguirse para que un prototipo se convierta en un producto comercial, lo que contribuirá con la creación de nuevos productos y la generación de empleo.

La utilización de herramientas de bajo nivel permite que el estudiante conozca y controle los diferentes pasos de la metodología de diseño y sea capaz de ajustarlas para diferentes situaciones, esto hace que se adquiera un conocimiento sobre la tecnología sin crear dependencia hacia las herramientas comerciales que realizan la mayoría de los pasos de la metodología de forma automática.

### REFERENCES

- [1] R. M. Stallman. *The GNU Operating System and the Free Software Movement Voices from the Open Source Revolution*. O'Reilly and Associates, 1999.
- [2] A. Grove. How America Can Create Jobs. [http://www.businessweek.com/magazine/content/10\\_28/b4186048358596.htm](http://www.businessweek.com/magazine/content/10_28/b4186048358596.htm), May 2010.
- [3] Jon Hall. POR GRANDES QUE SEAN...: ASEGURE EL FUTURO DE SU NEGOCIO. *Linux magazine*, ISSN 1576-4079(58):92, 2009.
- [4] H. Mitsui, H. Kambe, and H. Koizumi. Use of Student Experiments for Teaching Embedded Software Development Including HW/SW Co-Design. *IEEE TRANSACTIONS ON EDUCATION*, 52(3), August 2009.
- [5] A. Gerstlauer, D. Gajski, . Technical Report CECS-02-17, and 2002. CECS, UC Irvine. System-level abstraction semantics, Technical Report CECS-02-17. Technical report, CECS, UC Irvine, 2002.
- [6] Gajski D.D., Abdi S., Gerstlauer A., and Schirner G. *Embedded System Design: Modeling, Synthesis, Verification*. Springer, 2009.
- [7] Luis Alejandro Cortés. *Verification and Scheduling Techniques for Real-Time Embedded Systems*. PhD thesis, Linköpings universitet Institute of Technology, 2005.
- [8] Qi Hardware. Qi Hardware Copyleft Hardware Project. URL: <http://en.qi-hardware.com/>.
- [9] R. Stallman. Philosophy of the GNU project. URL: <http://www.gnu.org/philosophy/>, 2007.
- [10] W. Spraul, C. Camargo, and A. Wang. Proyecto SAKC. URL:<http://en.qi-hardware.com/wiki/SAKC>.
- [11] Creative Commons. Licencias Creative Commons. URL: <http://creativecommons.org/licenses/>, 2004.
- [12] C. Camargo. Proyecto SIE: Descargas. <http://projects.qi-hardware.com/index.php/p/mm-usb-fpga/source/tree/master/>, 2010.
- [13] C. Camargo. Proyecto SIE: Documentación. <http://en.qi-hardware.com/wiki/SAKC>, 2010.
- [14] Texas Instruments. IEEE Std 1149.1 (JTAG) Testability. *1997 Semiconductor Group*, 1996.
- [15] S. Shah. Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. *Management Science*, July 2006.

**Metodología Para la Transferencia Tecnológica en la Industria Electrónica Basada en Software Libre y Hardware Copyleft.** VI Congreso Internacional de la Red de Investigación Y Docencia en Innovación Tecnológica RIDIT “Innovación, Empresa Y Región”.

**VI CONGRESO INTERNACIONAL DE LA RED DE INVESTIGACIÓN Y DOCENCIA  
EN INNOVACIÓN TECNOLÓGICA RIDIT  
“INNOVACIÓN, EMPRESA Y REGIÓN”  
MANIZALES – COLOMBIA**

**METODOLOGÍA PARA LA TRANSFERENCIA TECNOLÓGICA  
EN LA INDUSTRIA ELECTRÓNICA BASADA EN SOFTWARE  
LIBRE Y HARDWARE COPYLEFT\*.**

**Carlos I. Camargo Bareño\*\***

**RESUMEN**

Los canales tradicionales para la transferencia tecnológica en el área del diseño de sistemas embebidos no han sido exitosos en los países en vía de desarrollo donde la plataforma tecnológica no está lo suficientemente desarrollada para absorber esta nueva tecnología, esto debido a la escasa transferencia de conocimiento que pueda ser utilizado para generación de productos locales. Por otro lado, existe una sobre-oferta de profesionales afines con la industria electrónica, una gran parte de ellos provienen de entidades poco consolidadas; la unión de estos factores genera una tasa de desempleo muy alta, salarios bajos, aumento de la dependencia a los productos extranjeros y una desconfianza hacia los productos generados localmente, lo que afecta de forma considerable el número de estudiantes que ingresan a los programas de formación relacionados con la electrónica, llegando hasta el punto del cierre de programas acreditados.

Este artículo presenta una metodología para la transferencia tecnológica en el diseño de sistemas embebidos desarrollada en el Departamento de ingeniería eléctrica y electrónica de la Universidad Nacional de Colombia; esta metodología tiene como pilares el conocimiento como bien común, el movimiento de software libre y un concepto nuevo desarrollado en conjunto con

\* Estudio financiado por la Universidad Nacional de Colombia, hace parte del trabajo de tesis doctoral:  
Metodología para la Transferencia Tecnológica y de Conocimientos en el Diseño de Sistemas Embebidos.

\* \* Magíster en Ingeniería Eléctrica, Candidato a Doctor por la Universidad Nacional de Colombia, profesor asistente del Departamento de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Colombia sede Bogotá e-mail: cicamargoba@unal.edu.co

un grupo de desarrolladores hardware y software: el hardware copyleft.

**PALABRAS CLAVES** Sistemas Embebidos, educación en ingeniería, hardware copyleft, transferencia tecnológica..

## Introducción

La transferencia de tecnología ha introducido técnicas de alta productividad y en muchos casos cambios técnicos en países menos desarrollados. La adquisición de tecnología foránea contribuye a mejorar la competitividad en los mercados locales e internacionales en estos países, en los que debe ser considerada como un proceso vital. Este proceso presenta problemas cuando se pierde capacidad de absorción por parte del país receptor y la renuencia del país que transfiere a transferir tecnología real y el *know-how*. Por lo que es necesario que estos países promuevan sus capacidades tecnológicas con el fin de absorber las tecnologías foráneas de forma eficiente en función de sus necesidades locales y de esta forma generar un rápido proceso de industrialización. La transferencia de tecnología según Van Gigch involucra la adquisición de "actividad Inventiva" por parte de usuarios secundarios. Es decir, la transferencia tecnológica no involucra necesariamente maquinaria o dispositivos físicos; el conocimiento puede ser transferido a través de entrenamiento y educación, y puede incluir temas como manejo efectivo de procesos y cambios tecnológicos (Bar, 2007). No debe confundirse la transferencia tecnológica con la apropiación de tecnología que se define como el proceso de interacción con la tecnología, la modificación de la forma como es usada y el marco social dentro del cual es usada. Un ejemplo de apropiación de tecnología lo podemos encontrar en la telefonía celular, nuestras sociedades han cambiado drásticamente su forma de comunicarse y han generado nuevas actividades alrededor de esta tecnología, los usuarios pueden generar aplicaciones que adicionan funcionalidades y servicios.

## Tecnología

La tecnología es definida como el factor más significativo para mejorar la productividad, calidad y competitividad (Cohen, 2004) y puede verse como un proceso de transformación que tiene como entrada recursos naturales, bienes, o productos semi-manufacturados y como salida se obtienen bienes consumibles de capital y semi-manufacturados. El *Technology Atlas team* identifica cuatro componentes de la tecnología (Bar, 2007): Techno-ware relacionado con objetos, herramientas, equipos, máquinas, vehículos, facilidades físicas, instrumentos, dispositivos y fábricas; Human-ware relacionado con personas, habilidades en conocimiento experimental, sabiduría y creatividad, experiencia, competencia; Info-ware Relacionado con la información, incluye todo tipo de documentación y datos acumulados relacionados con especificación de procesos, procedimientos, diseños, teorías, y observaciones; orgaware relacionado con la organización, acuerdos y alianzas necesarias para facilitar la integración de los componentes técnico, humano, y de información. La tecnología se encuentra fuertemente relacionada con un espectro amplio de las necesidades humanas, las condiciones físicas existentes o por factores culturales derivados de las especificidades históricas de diferentes grupos sociales (Goel, 1995).

## **Transferencia tecnológica**

(Odedra, 1994) define la transferencia tecnológica como el problema de transferencia de conocimiento (o know-how) sobre un número de aspectos (que incluyen el conocimiento) sobre como funciona un determinado sistema, como operarlo y desarrollar sus aplicaciones, como mantenerlo y si es necesario, como producir sus componentes y implementar un sistema similar. La transferencia tecnológica se considera exitosa cuando los receptores de la tecnología asimilan los conceptos anteriores para suplir sus necesidades locales. Según Jolly, 1997, La innovación tecnológica es entendida como un nuevo método, medio o capacidad del individuo para realizar una determinada actividad. El resultado de la transferencia tecnológica puede ser la aceptación de

una práctica común en otros lugares, o la aplicación de una técnica diseñada para otro uso en la solución de problemas locales. La transferencia tecnológica incluye la difusión de conocimiento científico y la preocupación por la transformación del conocimiento en innovaciones útiles. El conocimiento es lo que queda al final de un proceso documentado y difundido de forma apropiada. Para que la transferencia tecnológica sea exitosa es necesario transferir sus componentes.

1) Tipos de Transferencia Tecnológica: Mansfield, 1975, clasifica la transferencia tecnológica en transferencia de material: artefactos tecnológicos, materiales, productos finales, componentes, equipos; transferencia de diseño: diseños, proyectos, know-how para fabricar productos diseñados previamente, los productos son copiados para producirlos localmente (ingeniería inversa); transferencia de capacidades: proporciona know-how y software no solo para fabricar componentes existentes, sino para innovar y adaptar tecnologías existentes para generar nuevos productos. La transferencia de material no constituye una transferencia tecnológica real, ya que no genera el conocimiento necesario para transformarlos y generar nuevos productos que cumplan con las necesidades locales. La transferencia de diseños permite adquirir mayor conocimiento sobre la tecnología transferida, sin embargo, es necesario que el país receptor cuente con la plataforma tecnológica adecuada para absorber estos conocimientos, de lo contrario no se generarán nuevos productos y las actividades se limitarán al ensamblaje de productos manufacturados, La transferencia de capacidades es ideal, ya que proporciona las herramientas necesarias para que la transferencia sea exitosa, está asociada a una transferencia de conocimiento, lo cual es vital para entender plenamente la tecnología, mejorando las habilidades de los profesionales del receptor, creando una demanda de bienes y servicios relacionados con el conocimiento transferido; lo que se traduce en generación de empleo y aumento del bienestar general.

## **Canales para la transferencia de tecnología**

Grimpe, C. y Hussinger, K. (2008) clasifican los mecanismos en Formales: acuerdos de licenciamiento, inversión extranjera, compañías conjuntas, acuerdos de cooperación en investigación, arreglos de producción conjunta e Informales: No involucran acuerdos entre las partes y son difíciles de detectar y monitorear, por ejemplo, exportación de productos tecnológicos o bienes de capital, ingeniería inversa, intercambio de personal técnico y científico, conferencias de ciencia y tecnología, ferias y exposiciones, educación y entrenamiento realizado por extranjeros, visitas comerciales, literatura abierta (artículos, revistas, libros técnicos), espionaje industrial. Adicionalmente, existe una división basada en la naturaleza de la institución que proporciona los recursos para que se realice la transferencia, la institución puede ser de carácter Abierta: en donde la tecnología y el conocimiento son considerados bienes públicos, no existen restricciones para acceder a la información necesaria para adquirir, usar y transformar estos conocimientos en productos comerciales, y su éxito radica en obtener la máxima difusión posible para que los usuarios de este conocimiento mejoren el material existente y contribuyan a su crecimiento con experiencias personales; Cerrada La tecnología y el conocimiento se genera para fines privados, la utilización de este conocimiento esta sometida a acuerdos comerciales, no es posible entender las bases de la tecnología, por lo que no se pueden generar productos derivados.

Las actividades realizadas durante este estudio están enmarcadas dentro del concepto: El conocimiento es un *bien común*, toda la documentación necesaria para reproducir, entrenar, entender y modificar los productos generados se encuentran disponibles en servidores públicos (Camargo, 2011) y se proporciona soporte a través de listas de discusión, adicionalmente se proporciona soporte comercial para permitir la producción de estas modificaciones. A continuación se realiza una descripción de los canales más utilizados para la transferencia de

tecnología y conocimiento en países en vía de desarrollo ( Odedra, 1990; 1991; 1994) indicando en cada caso sus ventajas, limitaciones y desventajas.

1) Adquisición de IT: Con la venta de equipos se transmite únicamente el conocimiento para operar, programar o mantener, sin embargo, este conocimiento sobre el sistema puede ayudar a concientizarse sobre la tecnología e impulsar la formación de capital humano. Colombia ha realizado un proceso de transformación tecnológica pero no ha diseñado políticas efectivas y eficientes para la transferencia de tecnologías de alto nivel.

2) Educación y Entrenamiento: Educar a las personas enviándolas al extranjero es una forma de adquirir know-how sobre nuevas tecnologías, sin embargo, no se presenta una transferencia cuando estudiantes formados en el exterior no pueden aplicar sus conocimientos en su país de origen, por lo que es necesario crear políticas que definan que áreas de estudio son prioritarias para el país. Por otro lado, muchas instituciones que ofrecen carreras en ingeniería electrónica, ciencias de la computación y afines, utilizan modelos pedagógicos copiados de países desarrollados, los que no han sido adaptados plenamente a la infraestructura tecnológica local, y no es raro encontrar estudiantes que al finalizar sus estudios no están satisfechos con su profesión (Odedra, 1994). Programas académicos inapropiados, acceso limitado, falta de facilidades para capacitación, reduce la efectividad de la educación y capacitación como canal para la transferencia tecnológica.

3) Asistencia Técnica: La ventaja de contratar consultores externos radica en el ahorro de tiempo y dinero, ya que, utilizar personal local implicaría un gran esfuerzo y posiblemente se tendrían que asumir errores costosos en el proceso. Sin embargo, no es bueno confiar a consultores externos la responsabilidad de construir habilidades locales, ya que reduce el desarrollo local de estas, especialmente, la del personal encargado de manejar proyectos. La falta de personal calificado hace que los consultores se encarguen de todas las tareas del proyecto, lo que aumenta

su carga de trabajo y disminuye la posibilidad de entrenamiento de personal local (Odedra, 1990).

4) Licenciamiento: El licenciamiento es un canal que se utiliza para transferencia de know-how sobre productos o procesos, sin embargo, no es efectivo si no se acompaña de habilidades administrativas y de producción. Adicionalmente, es necesario contar con una infraestructura tecnológica adecuada, capacidades locales de fabricación de hardware y software y políticas de gobierno adecuadas (Odedra, 1991).

5) Inversión Extranjera Directa: La inversión directa de multinacionales es una forma de obtener tecnología externa, asegurando una rápida transferencia de información, pero no necesariamente de know-how, lo que hace que la tecnología transferida a través de este canal sea mínima. Las grandes multinacionales pueden tener cierto control político en los países en vía de desarrollo, hasta tal punto que son asesores de instituciones encargadas de fijar políticas para la transferencia tecnológica (Odedra, 1994).

## **1. Movimiento software libre y hardware copyleft**

El movimiento de Software Libre y Código Abierto (FOSS) es la estructura auto-gobernada más exitosa, su principal innovación radica en un nuevo esquema de licencias unido a herramientas de colaboración basadas en Internet, lo que se convirtió en una nueva forma de bien común donde los miembros de forma colectiva generan un beneficio común el software. El desafío de FOSS es realizar acciones colectivas para crear y mantener este bien público. A diferencia de la creencia popular, en este movimiento existen derechos de autor y propiedad intelectual, se poseen derechos legales sobre el código (recurso), tienen control sobre las nuevas versiones del software y pueden excluir a otros que aportan código a las nuevas distribuciones.

*Transferencia tecnológica:* El movimiento FOSS puede ser considerado como una institución

abierta que permite y promueve la transferencia tecnológica; gracias a la disponibilidad del código fuente es posible aprender nuevas técnicas de programación, utilizar aplicaciones existentes como herramientas de desarrollo y entender su funcionamiento y principio de operación. En la actualidad las grandes multinacionales dedicadas a la generación de dispositivos electrónicos de consumo masivo están utilizando productos derivados del proyecto FOSS, esto permite tener acceso a la tecnología de estos fabricantes para aplicarlas en productos locales. Los foros de discusión permiten tener contacto directo con los creadores de los proyectos FOSS todas las conversaciones, preguntas/respuestas son almacenadas y se proporcionan herramientas que permiten realizar búsquedas en ellas, lo que constituye una fuente de información valiosa para cualquier persona que quiera mejorar sus conocimientos o habilidades en el uso de una determinada herramienta, lenguaje de programación o aplicación. El movimiento FOSS permite que cualquier persona interesada pueda estudiar y entender la estructura del código de una determinada aplicación, permitiendo su modificación para uso particular o comercial, y de esta forma beneficiar a un sector de la sociedad; lo que no hubiera sido posible con los productos comerciales.

*Conocimiento como bien común:* El Software no es un recurso típico porque no es sustraible, no existe un costo si un usuario decide usarlo o no; sin embargo, es un bien común, ya que es prospera o decae gracias a la contribución de sus miembros; sus usuarios son contribuyentes, en lugar de un grupo de propietarios; un aporte de un usuario que mejore el recurso se traduce en un beneficio colectivo. Los contribuyentes pueden ser empresas que pagan a una persona para que adicione una nueva característica o adicione soporte a los productos de dichas empresas, sin embargo, ellos pueden decidir no compartir estos cambios para tener una ventaja competitiva. En la actualidad observamos las tendencias de multinacionales como Nokia, Intel, Google (Android), a participar, crear y promover proyectos de software libre.<sup>1</sup>

---

1 Es más rentable utilizar herramientas que han sido desarrolladas, probadas, y depuradas por miles de usuarios

El recurso del proyecto FOSS está compuesto por un gran grupo de desarrolladores y voluntarios que aportan nuevas aplicaciones software, depuran, documentan y actualizan las aplicaciones existentes; una plataforma que se encarga de administrar los proyectos software: control de revisiones, listas de discusión, wikis; un sistema de licenciamiento que permite entender, utilizar y modificar proyectos existentes. La "situación de acción" o la decisión que estos programadores deben tomar es contribuir o no al desarrollo de este software (Hess y Ostrom 2006). La interacción de programadores trabajando de forma conjunta en internet puede ser visto como un resultado que puede cambiar en el tiempo. Schweik y Semenov 2003, identificaron tres estados de este tipo de bien común, una fase inicial, seguida por un estado de apertura y un estado más maduro de gran crecimiento (en términos de usuarios y participación), estabilización donde el número de participantes (normalmente pequeño) no varía, o el proyecto se estanca y muere (sin participantes). La clave del éxito está en la disponibilidad de un programador para contribuir al esfuerzo colectivo de por lo menos un pequeño grupo de actores que producen y mantienen el software.

## **2. Harware Copyleft**

El movimiento FOSS proporciona todas las herramientas necesarias para el desarrollo de aplicaciones en sistemas digitales, sin embargo, este software debe ejecutarse sobre una plataforma hardware compuesta por microcontroladores, memorias volátiles y no volátiles, dispositivos de comunicación e interfaces hombre - máquina. En la actualidad existen proyectos aislados como Arduino, Beagle Board, chumby y NanoNote que son presentados como *open-source*, todos ellos proporcionan: esquemáticos detallados en formato PDF, código fuente del software necesario para su operación, soporte a través de listas de discusión, tutoriales sobre su funcionamiento y programación, software de desarrollo, archivos de diseño: esquemático, layout, archivos para la fabricación de las placas de circuito impreso. Los costos necesarios para calificados que pagar a un grupo de decenas de desarrolladores para desarrollar desde cero una plataforma software

reproducir una de estas plataformas es muy elevado, esto se debe a las herramientas propietarias utilizadas en el diseño (10000 USD); las características de la placa de circuito impreso, (400 USD), el costo de los componentes (400 USD), y el montaje de los componentes (100 USD). Adicionalmente, algunos componentes son difíciles de obtener ya que requieren la firma de acuerdos de confidencialidad. Esto hace que los proyectos *open-source hardware* no presenten una gran dinámica en desarrollo hardware y las contribuciones se limitan a la fabricación de tarjetas de expansión para propósitos específicos. Desde el punto de vista comercial, estos proyectos son muy atractivos ya que permiten modificar un diseño validado, ahorrando mucho tiempo y dinero, desde el punto de vista académico, tener acceso a los archivos de diseño, permite estudiar las técnicas utilizadas en su diseño para poder aplicarlas en proyectos propios.

El uso de estas plataformas de desarrollo para aplicaciones comerciales genera dependencia hacia ellas y elimina la necesidad de desarrollo hardware; la dependencia puede llegar hasta el punto de eliminar por completo la generación local; multinacionales como Microsoft, Dell, Apple, Nokia utilizan a empresas asiáticas como Foxconn para la fabricación de todos sus productos; lo que trae como consecuencia que el número de empleados de Foxconn sea mayor que el de Microsoft, Dell, Apple, Nokia combinado (fuente: Bureau of Labor Statistics). Las políticas económicas deben estar centradas en la creación de empleo a todo nivel, al eliminar la actividad manufacturera se eliminan una gran cantidad de empleos directos e indirectos dejando únicamente los relacionados con diseño de aplicaciones, comercialización y ventas.

Los países en vía de desarrollo tienen un elevado grado de dependencia hacia los productos tecnológicos relacionados con los sistemas digitales; esto se debe en gran parte a los problemas mencionados anteriormente y a la no existencia de una plataforma capaz de absorber las nuevas tecnologías, esto en gran parte se debe a que la información se ha centralizado en grupos privados que no están interesados en la difusión masiva de este conocimiento. Adicionalmente,

conseguir el nivel de conocimiento necesario para la realización de dispositivos comerciales utilizando tecnologías modernas requiere un gran esfuerzo que puede llevar varios años; por otro lado, la producción masiva de un determinado dispositivo requiere relaciones en el mercado de proveedores de componentes, una infraestructura especializada en la manufactura y montaje de placas de circuito impreso, y personal especializado en la realización de pruebas así como la infraestructura necesaria para comercializar los productos finales; muchas empresas en países en vía de desarrollo no poseen estas relaciones y no existen proveedores locales de estos servicios; esto se debe a la poca demanda y a la gran dependencia de los productos extranjeros. Adquirir la infraestructura necesaria para la producción local de productos electrónicos requiere una fuerte inversión económica, y una demanda interna que estimule nuevas inversiones para aumentar y mejorar dichas facilidades, lo que no puede realizarse a corto plazo sin la intervención de la academia, la industria y el gobierno. Una opción que puede generar resultados a mediano plazo es utilizar la infraestructura manufacturera asiática con el fin de competir en precio y aumentar la demanda interna de bienes y servicios hasta llegar al punto de eliminar la dependencia.

*Transferencia tecnológica* La iniciativa hardware copyleft ofrece un canal para la transferencia tecnológica en el área de diseño y producción de sistemas digitales proporcionando: Diseños de referencia realizados con herramientas libres, documentación completa del proceso de diseño y fabricación, herramientas de desarrollo abiertas, tutoriales y diseños documentados, listas de discusión, contacto con proveedores de componentes, acceso a servicio de prototipado y producción; ahorrando tiempo y dinero en la creación de nuevos productos que satisfacen necesidades locales. El hardware copyleft es el complemento perfecto del software libre ya que permite la realización de plataformas completamente abiertas que pueden ser reproducidas y modificadas incluso para fines comerciales.

*Creative Commons* organización no gubernamental sin ánimo de lucro ha desarrollado una serie

de licencias basadas en principios similares a los del movimiento software libre, que pueden ser aplicadas a trabajos realizados en música, arte, video, texto y notas de clase<sup>2</sup>. Estas licencias permiten que el autor de un trabajo conserve la propiedad intelectual, permitiendo su copia y distribución, con la única condición de dar créditos al autor del trabajo original y que el trabajo derivado posea la misma licencia. Se pueden elegir diferentes permisos dependiendo de los deseos del autor, para definir dichos permisos Creative Commons proporciona una serie de preguntas que buscan determinar los derechos que se desean conservar y los que desean liberar. Las preguntas claves son: 1) Los lectores pueden copiar y distribuir su trabajo libremente? 2) Es permitido a los usuarios crear trabajos derivados del contenido digital? Si es permitido, estas modificaciones deben tener la misma licencia que el trabajo original (esquema de licencia viral), o deben ser distribuidos bajo un esquema de licencias diferente? 3) Es necesario atribuir el trabajo al autor original?. Estas preguntas son la base para definir un esquema de licencias modular; existen cuatro bloques constructivos para las licencias Creative Commons estos son:

Atribución (BY) Se permite la distribución dando crédito al autor. No comercial (NC) Se permite la distribución del trabajo sin fines comerciales, si se desea utilizar este trabajo para obtener dinero es necesaria una autorización del autor; No a trabajos derivados (ND) Permite la copia y la distribución del trabajo original sin modificaciones. Compartir de la misma forma (SA) Exige que todo trabajo derivado del uso de proyectos que con este esquema de licencias deben tener la misma licencia de los trabajos originales.

Al compartir el trabajo realizado con otras personas para que puedan aumentar sus habilidades o con empresas para que puedan utilizar los proyectos existentes y ahorrar tiempo y dinero en la realización de nuevos productos, los cuales a su vez, pasarán a ser parte de este recurso

---

<sup>2</sup>Para reproducir una plataforma Hardware, es necesario suministrar los archivos de las herramientas CAD de diseño y fabricación (PCB, Lista de materiales), por lo que esta iniciativa resulta adecuada para distribuir este tipo de proyectos

(conocimiento), se proporciona una canal de transferencia sin restricciones que permitirá la difusión de los conocimientos adquiridos; este canal novedoso va en contra de la forma tradicional de capacitación ya que no limita la difusión a factores económicos. La iniciativa *hardware copyleft* adopta los bloques constructivos BY (Atribución) y SA (Compartir de la misma forma), lo que indica que se puede utilizar, distribuir y modificar el resultado de esta investigación pero dando crédito al autor original; y todo trabajo derivado del uso de los resultados de este proyecto debe tener la misma licencia lo que garantiza el crecimiento del conocimiento generado.

### **3. Metodología para la transferencia tecnológica en el área de diseño de sistemas embebidos.**

A continuación se describirán los pasos de una propuesta metodológica que tiene como objetivo permitir una transferencia tecnológica exitosa en el área de diseño de sistemas embebidos. Esta metodología está compuesta por los siguientes 7 pasos y actividades:

#### **Elección y adquisición**

*Niveles de complejidad de la tecnología:* Existen varias alternativas para la implementación de un sistema embebido: FPGA, sistema sobre silicio (SoC), SoC + FPGA y ASIC, la utilización de FPGAs o SoCs está determinada por el cumplimiento de restricciones temporales y funcionales, mientras que el uso de ASICs depende de el número de unidades producidas, se estima que a partir de 10 mil unidades se debe utilizar un ASIC para reducir los costos de producción. Debido a que los niveles de producción de los países en vía de desarrollo no son muy grandes, se abordará el problema de la transferencia utilizando FPGAs y SoCs comerciales, sin descuidar el estudio e implementación de ASICs. Adicionalmente, la inversión necesaria para construir un circuito integrado es muy alta, y puede ser considerada como un punto final.

*Diagnóstico de la Industria Local:* Para determinar el estado de la industria electrónica en

Colombia, se creó la empresa emQbit LTDA. en asociación con profesionales en ingeniería de sistemas, ingeniería eléctrica e ingeniería electrónica. Se identificaron los siguientes obstáculos para el desarrollo y comercialización de sistemas digitales: Falta de proveedores de bienes y servicios relacionados con la actividad (venta de dispositivos semiconductores, fabricación de placas de circuito impreso, montaje automático de componentes, etc); desconocimiento de la tecnología (alcances y limitaciones); uso de tecnologías y metodologías de diseño obsoletas; competencia con productos asiáticos de muy bajo costo; falta de confianza en los productos nacionales; desconexión de la academia con el sistema productivo; inexistencia de reglamentación de la industria manufactura electrónica; profesionales con pocos conocimientos en procesos de diseño y fabricación. Estos resultados confirman estudios consultados: Odedra, 1990; Duque. y Gauthier, 1999 Zuluaga, et.al 2007; Tovar y Rodríguez, 2007; Martínez, 2008.

*Diagnóstico de la Academia:* La tendencia moderna en los programas académicos a la utilización de herramientas de alto nivel para la enseñanza en áreas afines al desarrollo de dispositivos digitales; unido a la utilización de tecnologías y metodologías de diseño obsoletas, programas académicos centrados en el análisis y no al diseño, donde el paso final es la simulación y el personal docente no tiene ninguna experiencia en el sector productivo; origina una deficiencia de habilidades necesarias para realizar el proceso completo para el diseño de dispositivos, lo que se traduce en profesionales que no disponen de las herramientas necesarias para resolver los problemas del país y al mismo tiempo competir con los productos importados.

*Adquisición:* En la actualidad es muy fácil adquirir productos implementados con tecnología de punta, existe una gran variedad de plataformas de desarrollo y de dispositivos comerciales a los que se les puede aplicar ingeniería inversa para entender y modificar su funcionamiento. Al comienzo de este estudio se trabajó productos comerciales como consolas de juegos, reproductores MP4, portaretratos digitales, agendas electrónicas, para entender la composición y

funcionamiento de los dispositivos digitales modernos.

## **Adopción**

En esta etapa se utilizó la ingeniería inversa para: identificar las diferentes arquitecturas de los sistemas digitales adquiridos; identificar los mecanismos que permitan modificar su funcionamiento; identificar las herramientas que permiten crear nuevas funcionalidades; aprender la forma de utilizar los recursos tecnológicos existentes de forma adecuada; conocer la metodología de diseño. Se desarrollaron y/o adaptaron metodologías de diseño que utilizan herramientas de libre distribución.

## **Absorción**

La absorción es una actividad de aprendizaje que integra conocimiento que es nuevo para el país pero que no es nuevo para el mundo; en esta etapa se pasó de las plataformas comerciales al diseño de aplicaciones propias, para lo que se desarrollaron y/o adaptaron técnicas de fabricación al entorno local, se realizó la transferencia de los conocimientos adquiridos a la academia y se iniciaron contactos con empresas manufactureras locales y extranjeras. Se desarrollaron aplicaciones académicas como: Osciloscopio Digital utilizando FPAAs (Camargo 2005), Automatización de un Puente grúa a escala (Castillo, Camargo y Perez 2007), Control Adaptativo Embebido (Pedraza, Camargo, 2005), Control de un horno de reflujo para componentes de montaje superficial (Camargo 2005), Dispositivo de visualización de variables que suministra el computador de un automóvil (Camargo 2005), herramienta para realizar reconfiguración parcial de FPGAs (Camargo, Sanchez 2006), evolución de un arreglo de células utilizando algoritmos genéticos (Espinoza y Camargo, 2005) y comerciales como: Adquisición de datos para medición de calidad de energía; plataforma robótica didáctica; registro de aceleración de vehículos para compañías de seguros; sistema de seguimiento vehicular; monitor de signos vitales; diccionario basado en Wikipedia; menú electrónico y consola de juegos. Se

desarrollaron 6 plataformas de desarrollo bajo a licencia CC-BY-SA Camargo, 2008; Camargo, 2007; Camargo 2006, los archivos necesarios para su reproducción, tutoriales sobre su funcionamiento, notas de aplicación y diseños de referencia se encuentran disponibles a todo interesado en el sitio web: <http://wiki.linuxencaja.net>

## **Aplicación**

En esta etapa se transfieren los conocimientos adquiridos en las fases anteriores a la industria creada (emQbit) como parte de la metodología; esta empresa con la asesoría del autor de este trabajo realizaron los siguientes proyectos comerciales: control de tornos industriales, plataforma robótica didáctica, monitoreo de temperatura, sistema de seguimiento vehicular, sistema de medición de la calidad del suministro de energía eléctrica (EDEC), monitor de signos vitales (UNAL), sistema de comunicación encriptada utilizando el canal GSM (MICROENSAMBLE), switch de 4 canales de radio frecuencia (TESAMERICA), sistema de archivos encriptado para máquinas de votación (VOTING solutions), control de acceso en las estaciones de transmieno (SAR), todos estos proyectos involucraban el diseño y fabricación de prototipos y la programación de la aplicación. Adicionalmente, se creó, un programa académico para la enseñanza de sistemas digitales que crea las habilidades necesarias para concebir, diseñar, implementar y operar dispositivos digitales modernos y la definición del concepto hardware copyleft y su utilización como herramienta en la enseñanza de diseño de sistemas embebidos (Camargo, 2011b); este programa está siendo utilizado de forma oficial en el departamento de Ing. Eléctrica y electrónica de la Universidad Nacional de Colombia sede Bogotá,

## **Difusión**

Con las etapas previas se adquirieron los conocimientos necesarios para concebir, diseñar, implementar y operar dispositivos digitales, y la experiencia necesaria para realizar la producción a grandes escalas. La etapa de difusión (se esta realizando en el momento de escribir este

artículo) busca hacer llegar este conocimiento a todos los interesados, y de esta forma crear una comunidad que lo utilice como un recurso común; proporcionando a los centros de formación un programa académico actualizado que permite generar en los estudiantes las habilidades necesarias para innovar y generar empleo y a la industria le suministra herramientas que puede utilizar para desarrollo de nuevos productos comerciales y para la capacitación de su recurso humano. El proceso de difusión se realiza en dos frentes, en el plano académico se realizaron conferencias y cursos de actualización donde se presentan las plataformas *hardware copyleft* desarrolladas; hasta el momento 4 de las principales universidades públicas del país están comenzando a implementar total o parcialmente el programa académico de las asignaturas del área de electrónica digital (Camargo 2011a, Camargo 2011b); en el plano industrial se realizó una alianza entre el SENA, la Universidad Nacional de Colombia, empresas manufactureras de Taiwan, empresas de base tecnológica y universidades de diferentes regiones del país; esta alianza busca difundir los conocimientos adquiridos durante 5 años de desarrollo e investigación, para ello se creará una plataforma hardware flexible que cumpla con las condiciones del *hardware copyleft* y que permita la implementación de aplicaciones comerciales en diferentes áreas. Durante el desarrollo de este proyecto las empresas participantes propondrán ideas para la realización de productos comerciales que den solución a problemas de su región, se vincularán estudiantes de los centros de formación que estén realizando su trabajo de grado para formar un equipo que diseñe y construya un dispositivo que implemente estas ideas en la plataforma flexible. Todo el proceso se documentará en un servidor de libre acceso, con el fin de hacerlo accesible a cualquier interesado, se suministrarán los archivos necesarios para posibilitar la reproducción de la capacitación, el diseño y fabricación de la plataforma flexible. Cada grupo de trabajo está encargado de llevar una bitácora donde se detalla el proceso de diseño y de suministrar los archivos necesarios para reproducir el producto final.

Adicionalmente, se creó el portal público *linuxencaja* para almacenar todos los achivos, documentos, tutoriales, notas de aplicación y diseños de referencia de las plataformas diseñadas en este estudio; se dispone de una wiki que puede ser utilizada por cualquier persona para documentar nuevos proyectos<sup>3</sup> y consultar los existentes; además se cuenta con una lista de correo que permite tener contacto con los desarrolladores y con otros interesados en el diseño hardware.

#### **4. Conclusiones y trabajo futuro**

El problema de los canales tradicionales para la transferencia tecnológica es la poca cantidad de conocimiento transferido a la sociedad receptora; la metodología aquí presentada se centra en la adquisición de conocimiento y en su difusión a todo el que este interesado, lo cual es totalmente opuesto a iniciativas similares que terminan en cursos de actualización que limitan la difusión de conocimiento a quien pueda pagar por él.

Este artículo presenta los resultados de la aplicación de una metodología que permite una transferencia tecnológica exitosa en el área de diseño de sistemas embebidos, obteniendo como resultados de este proceso la capacitación a empresas de base tecnológica en el uso de herramientas y metodologías de diseño modernas basadas en software libre y hardware copyleft; la actualización del contenido de las asignaturas relacionadas en los centros de formación participantes; la posibilidad de generación de empleo a diferentes niveles (de servicios, técnico, profesional, comercial, administrativo); la adopción y apropiación de esta tecnología por parte del sector productivo.

A futuro se debe observar los efectos que generan el cambio introducido en los programas académicos; los actuales usuarios del recurso deben encargarse de cuidarlo, mejorarlo, aumentarlo, y difundirlo para aumentar el número de usuarios/beneficiarios.

---

<sup>3</sup> En la actualidad los estudiantes de ing. Electrónica de la UNAL utilizan esta wiki para documentar los proyectos de las asignaturas del área de digitales y algunos trabajos de grado.

Proporcionando herramientas modernas a los futuros profesionales y alternativas de diseño y fabricación a futuros industriales es posible aumentar la oferta de productos tecnológicos locales.

## Referencias Bibliográficas

- Bar, F. Pisani, F., y Weber, M. (2007) Mobile technology appropriation in a distant mirror: baroque infiltration, creolization and cannibalism. *Seminario sobre Desarrollo Económico, Desarrollo Social y Comunicaciones Móviles en América Latina*. Buenos Aires.
- Camargo, C. (2005) Implementación de Sistemas Digitales Complejos Utilizando Sistemas Embebidos. *Memorias del XI Workshop de Iberchip*.
- Camargo, C. y O. Sanchez (2006). Linux embebido como herramienta para realizar reconfiguración parcial. *XII Workshop Iberchip*.
- Camargo, C. (2006). *First Colombian Linux SBC runs Debian*. Recuperado el 13 de enero de 2011 de <http://www.linuxfordevices.com/c/a/News/First-Colombian-Linux-SBCruns-Debian/>.
- Camargo, C. (2007). ECBOT: Arquitectura Abierta para Robots Móviles. *IEEE Colombian Workshop on Circuits and Systems*.
- Camargo, C. (2008). ECBOT y ECB AT91 Plataformas Abiertas para el Diseño de Sistemas Embebidos y Co-Diseño HW/SW. VIII Jornadas de Computación Reconfigurable y Aplicaciones, Madrid España.
- Camargo, C. (2011). Proyecto SIE. Recuperado el 12 de enero de 2011 de <http://www.linuxencaja.net/>
- Camargo, C. (2011a) SIE: Plataforma Hardware copyleft para la Enseñanza de Sistemas Digitales , *Memorias del XVII workshop de Iberchip*.
- Camargo, C. (2011b). Hardware copyleft como Herramienta para la Enseñanza de Sistemas Embebidos. *Simposio Argentino de Sistemas Embebidos*.
- Castillo, I. y Camargo, C. y Perez, C. (2006). Automatización de un puente grúa a escala, mediante una plataforma embebida la cual soporta multiprogramación. *XII Workshop Iberchip*.
- Cohen, G. (2004). Technology transfer: strategic management in developing countries. *Sage Publications inc.*
- Duque, M. y Gauthier, A. (1999) Formación de Ingenieros para la Innovación y el Desarrollo Tecnológico en Colombia. *Revista de la Facultad de Minas - Universidad Nacional de Colombia*, December.
- Espinosa, J y Camargo (2005). Evolución de un Arreglo de Células Utilizando Algoritmos Genéticos. *Memorias del XI Workshop de Iberchip*.

F. Pedraza, F. y Camargo, C. (2005). Control Adaptativo Embebido. *Memorias del XI workshop de Iberchip*.

Goel, K. y Sayers, B. (1995). Modelling Global-Oriented Energy Technology Transfer to DCs. Sixth *Global Warning International Conference*, San Francisco, 1995.

Grimpe, C. y Hussinger, K. (2008) Formal and Informal Technology Transfer from Academia to Industry: Complementarity Effects and Innovation Performance. *Centre for european economical research*.

Hess, C. y Ostrom, E. (2006). *Understanding Knowledge as a Commons: From Theory to Practice*. The MIT Press.

Jolly, A. (1977). The Technology Transfer Process: Concepts, Framework and Methodology. *The Journal of Technology Transfer*. Springer.

Mansfield, E. (1975) East-West technological transfer issues and problems, international technology transfer: Forms, resource requirements, and policies. *American Economic Review*.

Martínez, H. (2004) Apropiación de conocimiento en Colombia. El caso de los contratos de importación de tecnología. *Revista Cuadernos de Economía*.

Odedra, M. (1990). *Information Technology Transfer to Developing Countries: Case studies from Kenya, Zambia and Zimbabwe*. PhD thesis, London School of Economics.

Odedra, M. (1991). Information Technology Transfer to Developing Countries Is it really taking place? *The 4th IFIFTC9 International Conference on Human Choice and Computers*, North Holland, Amsterdam, Netherlands, HCC 4 held jointly with the CEC FAST Programme.

Odedra, M. (1994). The Myths and Illusions of Technology Transfer. *IFIP World Congress Proceedings*.

Schweik, C. y Semenov, A. (2003) *The Institutional Design of 'Open Source' Programming: Implications for Addressing Complex Public Policy and Management Problems*. Recuperado el 12 de enero de 2011 [http://www.firstmonday.org/issues/issue8\\_1/schweik/](http://www.firstmonday.org/issues/issue8_1/schweik/).

Tovar, M. y Rodríguez, R. (2007) *Prospectiva y vigilancia tecnológica de la electrónica en Colombia*. Master's thesis, Universidad Nacional de Colombia.

Zuluaga, D., Campos, S., Tovar, M., Rodríguez, R., Sánchez, J., Aguilera, A., Landínez, L., y Medina, J (2007) *Informe de Vigilancia Tecnológica: Aplicaciones de la Electrónica en el Sector Agrícola*. Technical report, COLCIENCIAS.

**ECBOT y ECB\_AT91 Plataformas Abiertas para el Diseño de Sistemas Embebidos y Co-Diseño HW/SW**, VIII Jornadas de Computación Reconfigurable y Aplicaciones, Madrid España.

# ECBOT y ECB\_AT91 Plataformas Abiertas Para el Diseño de Sistemas Embebidos y Co-Diseño HW/SW

**Abstract**—Este artículo presenta el trabajo realizado en el diseño de dos plataformas abiertas: ECBOT: Plataforma para el desarrollo de aplicaciones de robótica móvil, y ECB\_AT91: Sistema de Desarrollo para aplicaciones de Linux Embebido. Estas plataformas permiten la ejecución de Linux de forma nativa, lo que permite ser utilizadas en la enseñanza de metodologías de diseño de Sistemas Embebidos y en otras áreas del conocimiento donde pueda ser utilizado un sistema digital, como: control, instrumentación, procesamiento de información, comunicaciones, etc. La arquitectura de ECBOT permite crear aplicaciones que utilizan co-diseño HW/SW, lo cual, la convierte en una Herramienta pedagógica muy poderosa para la enseñanza en el área de Diseño de sistemas Digitales y por supuesto, en la creación de algoritmos de control para sistemas Multi-Robot. ECB\_AT91 permite ser adaptada a una gran variedad de aplicaciones, gracias a la gran cantidad de puertos de comunicaciones, y Pines de Entrada/Salida de propósito general disponibles.

**Index Terms**—Linux Embebido, Co-diseño HW/SW, Robótica móvil, Diseño de Sistemas Digitales, Instrumentación, FPGAs en Educación

## I. INTRODUCCIÓN

Según Campusano [1], en 1996 el New York Times estimó que el Americano promedio estaba en contacto con cerca de 60 microprocesadores cada día, y año tras año esta cifra va en aumento; esto nos indica, que los sistemas digitales están invadiendo las actividades humanas y hoy en día es muy difícil encontrar una situación en la que no se utilice un dispositivo digital. Existe un gran mercado que mueve miles de millones de dólares al año, el tamaño de este mercado es tal, que el número de procesadores vendidos para este fin en el 2000, según Venture Development Corporation<sup>1</sup> fué de 6.4 Billones y se estima que en el 2010 llegarán a 16 Billones.

Por otro lado, la robótica móvil es un campo de investigación que crece rápidamente y en la actualidad existe un gran número de grupos de investigación que trabajan en el desarrollo de técnicas de auto-organización para sistemas Multi-Robot, entre los que se encuentran: El proyecto Swarm-bots [2], [3], Interaction Labs (USC) [4] [5], Autonomous System Lab (EPFL), MIT Computer Science and Artificial Intelligence Laboratory [6]. Los robots utilizados por estos y otros grupos no son ajenos a la anteriormente mencionada invasión digital, aunque los primeros robots se construyeron con unidades de procesamiento basadas en microncontroladores, las nuevas plataformas disponibles presentan una gran variedad de ayudas al desarrollo de los algoritmos de control: poseen procesadores de 32 bits con una gran capacidad de computo, mecanismos “robustos” de comunicación inalámbricos, Sistemas Operativos con capacidades de Tiempo Real.

<sup>1</sup><http://www.vdc-corp.com/>

ECBOT fue desarrollado en el Departamento de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Colombia, con el objetivo de proporcionar una plataforma Hardware y Software **abierta** de bajo costo para el desarrollo de Sistemas Embebidos utilizando co-diseño HW/SW, con aplicación en Sistemas Multi-Robot (MRS). Gracias a que ECBOT utiliza el sistema operativo Linux, existe una gran cantidad de aplicaciones y librerías que pueden ser utilizadas en el desarrollo de algoritmos de control. Se eligió el proyecto *Player/Stage* [7], como plataforma de desarrollo de dichos algoritmos, *player* suministra funciones de alto nivel que encapsulan el manejo de bajo nivel de los sensores y actuadores, haciendo posible que la programación del Robot se realice en un lenguaje de programación de alto nivel como Java, Lisp, Python, C/C++, etc.

Gracias a su carácter abierto, ECBOT y ECB\_AT91 pueden ser construidas por cualquier persona, en cualquier lugar del mundo, los esquemáticos y los archivos de fabricación de la placa de circuito impreso se encuentran disponibles, así como el código fuente de las aplicaciones necesarias para obtener un dispositivo funcional. ECBOT permite la adopción de nuevos periféricos através de 7 señales análogas disponibles, 10 pines de Entrada/Salida digitales y un Puerto I2C, controlado por el procesador central. Los drivers necesarios para controlar estos puertos análogos y digitales de expansión también se encuentran disponibles.

## II. ARQUITECTURA GLOBAL DE ECBOT

ECBOT está compuesto por un procesador Central ARM920T de 32 bits, encargado de las comunicaciones (WiFi, serial), el procesamiento de la información, y la configuración/programación de la FPGA y de 5 microncontroladores AVR de 8 bits; los cuales se comunican con el procesador central através de una interfáz I2C. Adicionalmente ECBOT posee un acelerómetro que también se comunica por el puerto I2C. Se escogió el puerto I2C ya que este permite la conexión de hasta 128 diferentes dispositivos y utiliza solo dos líneas para controlarlos, lo cual reduce la interconexión y la complejidad de la placa de circuito impreso.

### A. Arquitectura Software

1) *Control de Sensores y Actuadores*: Como se mencionó anteriormente, ECBOT posee 6 microcontroladores de 8 bits (AVR de Atmel) que realizan el control de los sensores y actuadores (Motores DC, Sensores Infra-rojos y LEDs), Cada uno de estos microcontroladores implementa el protocolo

I2C esclavo y obedece a una única dirección (entre 0 y 127), adicionalmente implementa el control completo de su respectivo sensor o actuador.

2) *Linux Embebido*: Uno de los requerimientos de ECBOT fué la capacidad de ejecución en forma nativa del sistema operativo Linux. Esto debido a que Linux ya posee una gran trayectoria y miles de programadores han creado aplicaciones en una gran variedad de áreas del conocimiento, es un sistema operativo abierto que permite la creación de drivers para manejar dispositivos Hardware y es de libre distribución. Por esta razón, el primer paso en el desarrollo de ECBOT fue el estudio de plataformas que soportaran Linux, como resultado de este estudio se construyó la plataforma abierta: ECB\_AT91 [8], la cual fué la base del desarrollo que dio como resultado ECBOT.

Durante el desarrollo de ECB\_AT91 y de ECBOT se realizó el *port* de Linux a esta familia de plataformas, y este esfuerzo está incluido en la distribución “oficial” de los procesadores AT91 de Atmel<sup>2</sup>. Así mismo se realizaron variaciones sobre el loader de Linux u-boot<sup>3</sup>, y una serie de aplicaciones que se utilizan para programar y configurar los microcontroladores y la FPGA.

Se adaptaron 3 nuevos paquetes a las distribuciones Buildroot y OpenEmbedded: El servidor Player, el programador de microcontroladores para AVR: *uisp*<sup>4</sup> y el programador para FPGAs: *xc3sprog*<sup>5</sup>. Esto con el fin de hacer que ECBOT no requiera ser conectado a un computador personal para realizar cambios de programación en alguno de sus componentes, los microcontroladores de 8 bits y la FPGA, son programados por el procesador central a través de pines de Entrada/Salida de propósito general, un driver emula el comportamiento de un puerto paralelo, los archivos de programación/configuración (transferidos de forma inalámbrica) son almacenados en la plataforma y pueden ser modificados en cualquier momento; esto hace que ECBOT sea una plataforma independiente y que pueda ser programada “en caliente”.

## B. Arquitectura Hardware

Uno de los criterios de diseño de ECBOT fué la capacidad de expansión; su arquitectura debe permitir la adición de nuevos sensores y actuadores de forma fácil; por este motivo, se escogió el puerto I2C como protocolo de comunicación entre la Unidad de procesamiento Central y los sensores y actuadores. En la actualidad, la mayoría de los microcontroladores disponibles en el mercado implementan este protocolo, lo cual permite adaptar cualquier sensor a ECBOT. Las características de ECBOT se listan a continuación:

- Procesador: ARM 920 AT91RM9200 @ 200MHz
- Movimiento: 2 Servo motores DC
- Sensores, Entrada/Salida:

<sup>2</sup>El patch oficial para la Arquitectura AT91 de Atmel puede ser descargada de: [http://maxim.org.za/at91\\_26.html](http://maxim.org.za/at91_26.html)

<sup>3</sup><http://sourceforge.net/projects/u-boot>

<sup>4</sup><http://www.nongnu.org/uisp/>

<sup>5</sup><http://www.rogerstech.co.uk/xc3sprog/>

- 8 sensores Infrarrojos de proximidad y luz de ambiente
- 10 I/O Digitales, 8 Analogas
- 8 Leds Programables
- Sensor de Imagen de hasta 640x480 pixels.

- Comunicación:

- Puerto serie Standard
- Comunicación USB
- Ethernet Inalámbrico

- Simuladores: Player/Stage

- Herramientas de Desarrollo: GNU TOOLS

- Costo: 400 USD

Como puede verse en La Figura 1, seis microcontroladores de 8 bits (AVR de Atmel) permiten el manejo de los sensores y actuadores a través del bus común I2C. Cuatro microcontroladores de 8 bits, manejan cada uno de ellos: dos sensores infrarrojos de proximidad y luz de ambiente, utilizados en tareas de evasión de obstáculos y 6 LEDs (2-rojos, 2-verdes, 2-azules) que representan el estado interno del Robot; el quinto microcontrolador de 8 bits se encarga de manejar la velocidad y posición de 2 motores DC, para reducir el costo de los componentes ECBOT implementa un control de posición y velocidad de motores utilizando un método de medición de la velocidad basado en la fuerza electromotriz [9], [10]; el sexto procesador se encuentra disponible para que el usuario utilice sus entradas analógicas.

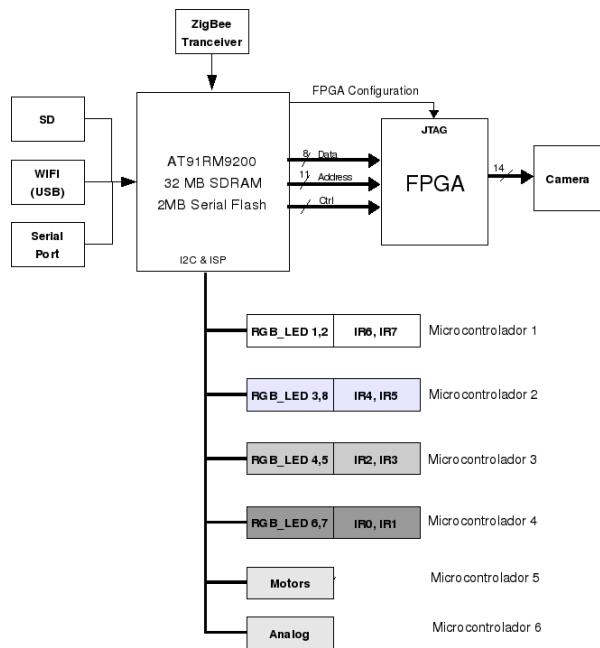


Fig. 1. Diagrama de Bloques del componente Hardware de ECBOT.

Adicionalmente ECBOT cuenta con una FPGA que comparte el bus de Datos, dirección y control con el procesador central, lo que permite la creación de periféricos dedicados y de aplicaciones HW/SW. En la placa se encuentran dos conectores que permiten que el estudiante tenga acceso a 26 pines (divididos en 16 y 10 señales) de Entrada/Salida de propósito General (GPIOs). En la plataforma robótica, se utilizan 16 GPIOs para manejar un sensor de imagen (KAC9628), y los 10 restantes para controlar 10 servo-motores.

*1) Comunicaciones:* ECBOT proporciona 2 formas de comunicación:

- Módulo WiFi: ECBOT permite la conexión de un adaptador USB 802.11, en la actualidad solo los adaptadores basados en el chip ZD1211 están soportados.
- Puerto Serie: En las primeras etapas del desarrollo es necesario utilizar el puerto serie para cargar las imágenes del loader, bootloader y kernel.

*2) Memorias y Dispositivos de Almacenamiento:* Se cuenta con una memoria Flash serial de 2 Mbytes donde se almacenan las imágenes del loader, bootloader y kernel, esta memoria puede ser modificada utilizando un loader que se ejecuta al inicializar la plataforma, o puede modificarse desde linux a través de un dispositivo MTD (Memory Technology Device). ECBOT permite la utilización de una memoria SDRAM de hasta 64 MBytes, la cual es utilizada como memoria de propósito general para las aplicaciones que corren bajo Linux. Adicionalmente se proporcionan dos conectores para memoria SD, uno para una SD standard y el otro para una tarjeta micro-SD.

*3) Unidad Central de Procesamiento:* El cerebro de ECBOT es un procesador de 32 Bits de la familia ARM de ATMELO el AT91RM9200, que corre a 180 MHz. Este procesador goza de gran popularidad dentro del grupo de desarrolladores de drivers para Linux, por lo que casi la totalidad de sus periféricos están soportados. Esto facilita la creación del soporte necesario para la board; la información necesaria sobre el *port* de Linux a la plataforma se puede encontrar en [11].

*4) Fotografías de la tarjeta principal de ECBOT:* En las Figuras 2 y 3 se muestra el lado de componentes y el lado de soldadura de la tarjeta principal de ECBOT; en ellas podemos observar la localización de los componentes, la cual está fuertemente influenciada por el robot *e-puck* [12] del EPFL.

### III. APLICACIONES DE LA PLATAFORMA ECBOT

#### A. Robótica

Resumiendo lo visto hasta el momento: Contamos con una plataforma HW que posee una serie de recursos para el desarrollo de aplicaciones en robótica móvil, tenemos una serie de sensores y actuadores que se comunican por el puerto I2C con el procesador central de dicha plataforma. Solo falta crear aplicaciones que utilicen la información proveniente de los sensores para generar acciones en los actuadores. Esto se puede realizar de dos formas: Utilizar los mecanismos de Entrada/Salida de los drivers de Linux, lo cual, aunque no es difícil es un poco engorroso y provoca que el programador se pierda entre los diferentes llamados a funciones, no recuerde con qué formato debe comunicarse con cada dispositivo, o qué información suministran, etc. La otra alternativa es utilizar un API que se encargue de este manejo tedioso y suministre al programador funciones de fácil utilización. Aquí es donde entra en acción el servidor Player:

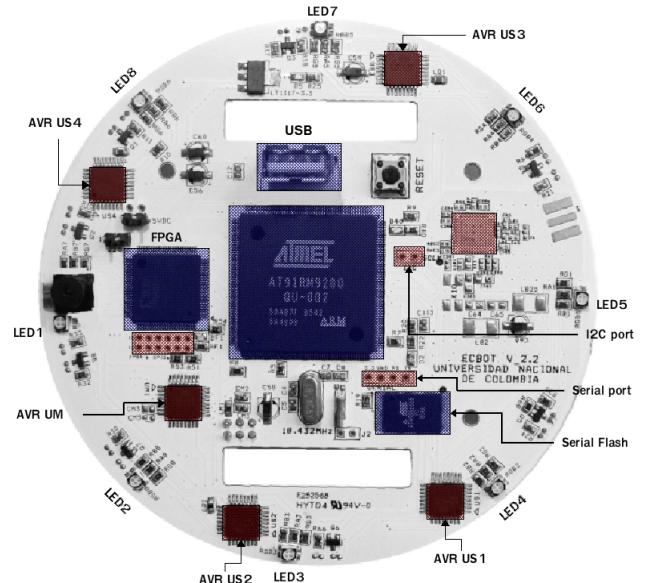


Fig. 2. Lado de Componentes de la tarjeta principal de ECBOT.

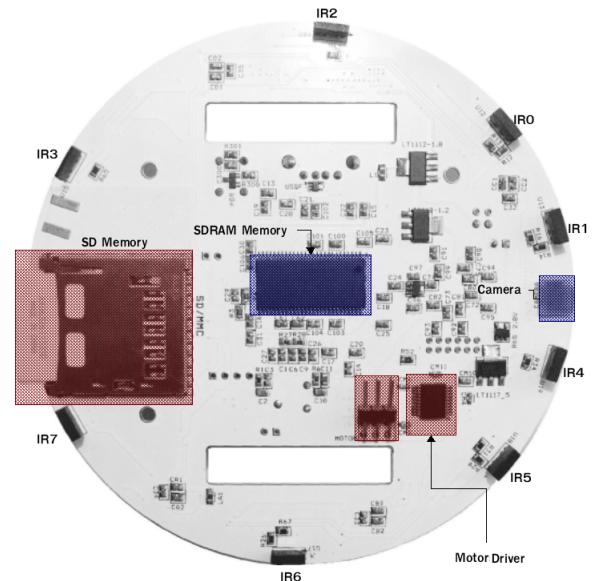


Fig. 3. Lado de soldadura de la tarjeta principal de ECBOT.

*1) Servidor Player:* El proyecto Player/Stage, es el resultado de un estudio del Grupo de Investigación en Robótica de la *University of Southern California* [7], y esta dividido en tres partes:

- 1) **Player** Es un servidor que proporciona una interfaz de red flexible a una gran variedad de sensores y actuadores. Como puede verse en la Figura 4 utiliza un modelo cliente/servidor basado en sockets TCP, su principal característica es el manejo de los sensores y actuadores a través de una interfaz de programación de alto nivel, lo cual permite que los programas de control del robot sean escritos en cualquier lenguaje y puedan ser ejecutados en

cualquier plataforma<sup>6</sup> (cliente) que posea una conexión de red con el robot (servidor). Además, Player soporta múltiples conexiones concurrentes de clientes, lo que es muy útil en estudios de control descentralizado.

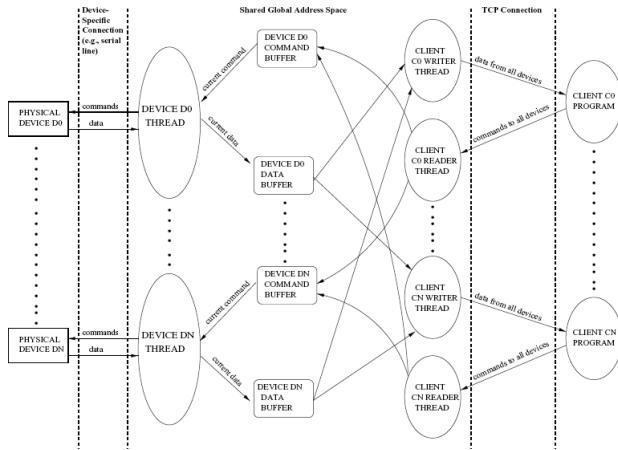


Fig. 4. Arquitectura de Player. [13]

- 2) **Stage** Es un simulador multi-robot; que simula robots que se mueven y realizan operaciones de sensado en un entorno de dos dimensiones, estos robots son controlados por Player. Stage proporciona robots virtuales los cuales interactúan con dispositivos simulados. En la Figura 5 se puede observar una simulación multi-robot utilizando las librerías de Stage; una de las características más atractivas del proyecto Player-Stage es la creación de sensores y actuadores que simulan el comportamiento de los dispositivos reales.
- 3) **Gazebo** Es un simulador multi-robot 3D. A diferencia de Stage que fué diseñado para simular grupos numerosos de

<sup>6</sup>También es posible que el cliente y el servidor se ejecuten en el mismo robot.

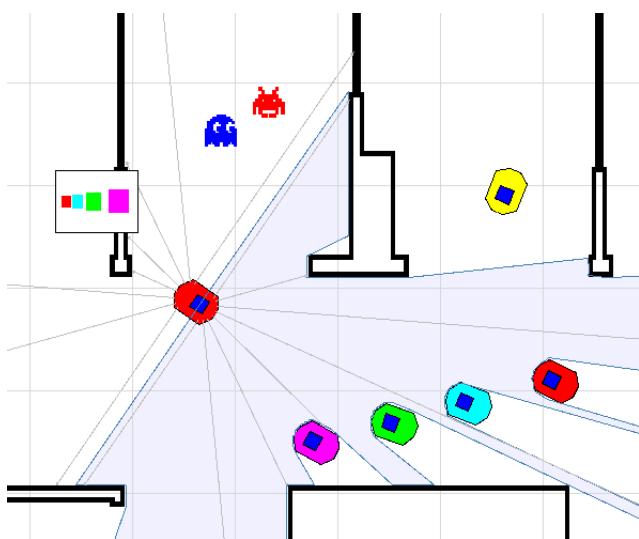


Fig. 5. Captura del simulador Stage.

robots, gazebo simula el comportamiento de poblaciones pequeñas de robots (menos de 10).

Adicionalmente, una de las características más atractivas del proyecto Player/Stage es su carácter libre y que su código fuente está disponible, gracias a esto, fué posible escribir un driver que soportara a ECBOT. Este driver permite controlar:

- 1) El dispositivo de seguimiento de color implementado en la FPGA.
- 2) La velocidad y posición de 2 motores.
- 3) El color de los LEDs.
- 4) La activación y la lectura de los sensores InfraRojos de proximidad.

La figura 6 muestra gráficamente la arquitectura del servidor Player de ECBOT, como puede verse, el servidor está encargado de comunicarse con los sensores (cámara y sensores InfraRojos), procesar esta información y enviarla al cliente, donde el algoritmo de control por intermedio de un generador de comandos del servidor controla los actuadores (LEDs y motores).

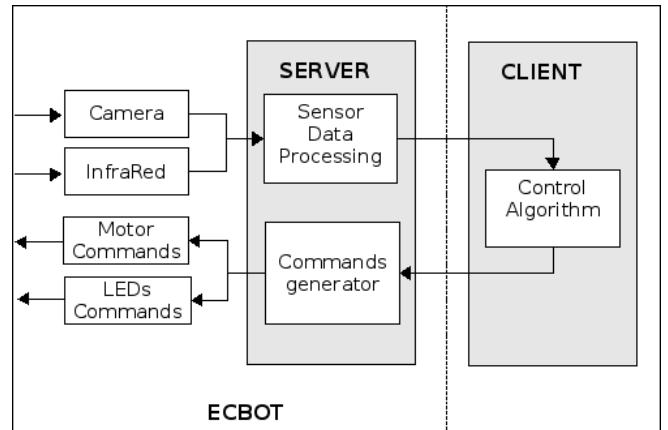


Fig. 6. Servidor Player de ECBOT.

#### IV. ARQUITECTURA GENERAL DE ECB\_AT91

##### A. Componente Hardware

Como se mencionó anteriormente ECB\_AT91<sup>7</sup> fué la plataforma que se diseñó para dar inicio al estudio de Sistemas Embebidos con Linux como sistema operativo, su importancia radica en ser el primer SBC fabricado en Colombia. Y gracias a que los archivos de fabricación y el código fuente de las aplicaciones requeridas para su funcionamiento se encuentran disponibles; ha sido utilizada en varias universidades en diferentes países de mundo; se tienen reportes de la utilización de ECB\_AT91 en: Perú, Venezuela, Portugal, USA, Bélgica, España, Italia, Alemania y Brasil. Nuestra empresa <sup>8</sup> recibe a diario muchos reportes y preguntas relacionadas con el diseño y funcionamiento de esta placa.

Las Características de esta placa son (favor ver Figuras 7 y 8):

<sup>7</sup><http://wiki.emqbit.com/free-ecb-at91>

<sup>8</sup><http://www.emqbit.com>

- procesador ARM9 de 180 MHz (Atmel AT91RM9200).
- 2 MBytes de memoria flash serial.
- Hasta 64 MBytes de memoria SDRAM (Soporta 8M/16M/32M/64M).
- 1 slot SD/MMC.
- 1 puerto host USB 2.0.
- 1 puerto I2C.
- 1 Interfaz Ethernet 10/100.
- 4 Interfaces SPI.
- 2 Interfaces seriales (RS232).
- Soporte JTAG.
- Bus de Datos (16 bits), Dirección (7 bits) y Control Disponibles.
- 8 Pines de Entrada/Salida de Propósito General Disponibles.

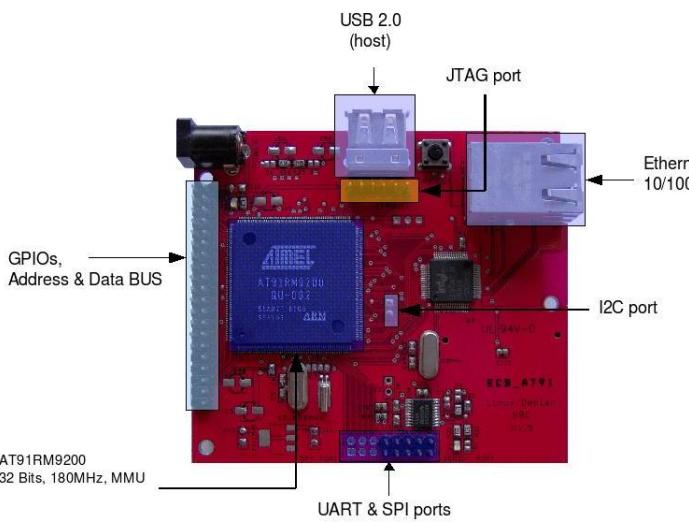


Fig. 7. Lado de Componentes de la placa ECB\_AT91.

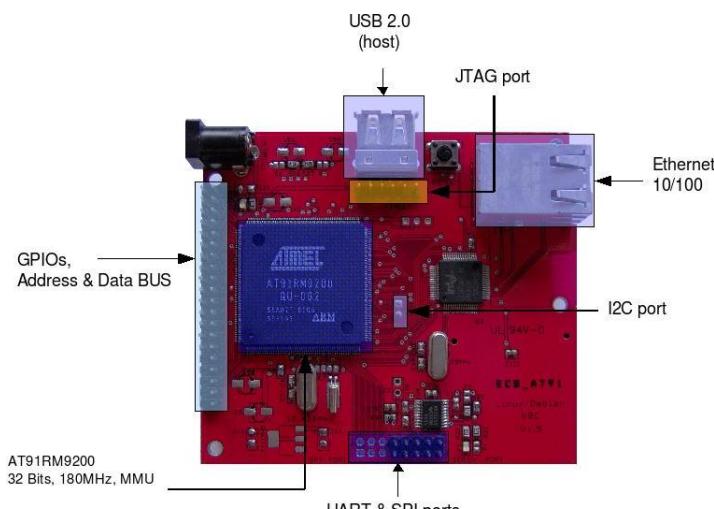


Fig. 8. Lado de Soldadura de la placa ECB\_AT91.

### B. Aplicaciones Realizadas con ECB\_AT91

ECB\_AT91 ha sido utilizada como base para el desarrollo de diversas aplicaciones, dentro de las que se destacan:

- 1) Sistema de Recolección, Procesamiento y Transferencia de Información para molinos de Campo Eléctrico.
- 2) Sistema de Automatización para Tornos Industriales.
- 3) Sistema de seguimiento Vehicular Utilizando la Red Celular.
- 4) Sistema de Monitoreo de Temperatura.
- 5) Sistema de Medición de Signos Vitales.
- 6) Gateway Para Red de Sensores Inalámbricos.

### V. CONCLUSIONES Y TRABAJO FUTURO

Se presentaron dos plataformas abiertas HW/SW para el estudio de sistemas embebidos y sistemas multi-robot, los archivos necesarios para la fabricación de las placas de circuito impreso, esquemáticos, el firmware de los microcontroladores y su respectivo código fuente, cambios requeridos al kernel de Linux se encuentran disponibles.

El estudio del diseño de Sistemas Embebidos es muy importante en el mundo de hoy, ya que existe una gran demanda en constante aumento por parte de la sociedad.

Linux ha demostrado ser una plataforma ideal para el desarrollo de este tipo de aplicaciones ya que gracias a la gran cantidad de aplicaciones y librerías disponibles reduce el tiempo requerido para la implementación de las aplicaciones.

ECBOT y ECB\_AT91 son dos herramientas poderosas en la enseñanza de sistemas digitales, ya que pueden ser utilizadas en diferentes niveles y en diferentes contextos (tecnología, ingeniería).

### VI.

#### REFERENCES

- [1] R. Camposano and W. Wolf. *Design Automation for Embedded Systems*. Springer, January 1996.
- [2] F. Mondada, G. C. Pettinari, I. Kwee, A. Guignard, L. Gambardella, D. Floreano, S. Nolfi, J.-L. Deneubourg, and M. Dorigo. SWARM-BOT: A swarm of autonomous mobile robots with self-assembling capabilities. In C.K. Hemelrijk and E. Bonabeau, editors, *Proceedings of the International Workshop on Self-organisation and Evolution of Social Behaviour*, pages 307–312, Monte Verità, Ascona, Switzerland, September 8-13, 2002. University of Zurich.
- [3] G. Baldassarre, S. Nolfi, and D. Parisi. Evolving mobile robots able to display collective behaviours. In C.K. Hemelrijk and E. Bonabeau, editors, *Proceedings of the International Workshop on Self-Organisation and Evolution of Social Behaviour*, pages 11–22, Monte Verità, Ascona, Switzerland, September 8-13, 2002. University of Zurich.
- [4] C. Jones and M. Mataric. Adaptive Division of Labor in Large-Scale Minimalist Multi-Robot Systems. *Proceedings of the IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS)*. Las Vegas, Nevada, 2003.
- [5] Jones and Maja J Mataric. *Autonomous Mobile Robots: Sensing, Control, Decision-Making, and Applications*, chapter Behavior-Based Coordination in Multi-Robot Systems. Marcel Dekker, Inc, 2005.
- [6] J. McLurkin. Speaking Swarmish. *AAAI Spring Symposium*, 2006.
- [7] B.P. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage project: Tools for Multi-robot and Distributed Sensor Systems,. *Proceedings of the International Conference on Advanced Robotics (ICAR)*, pages 317–323, 2003.
- [8] C. Camargo. First Colombian Linux SBC runs Debian. <http://www.linuxdevices.com/news/NS6698241512.html>, 24 April 2006.
- [9] Abu Zaharin and Mohd Nasir. A Study On The DC Motor Speed Control By Using Back-EMF Voltage. *ASIAN CONFERENCE ON SENSORS*, July 2003.

- [10] R. LeGrand. Closed-Loop Motion Control for Mobile Robotics. *Circuit Cellar Ink*, August 2004.
- [11] C. Camargo, N. Castillo, and A. Calderón. Free ECB\_AT91. <http://wiki.emqbit.com/wiki>.
- [12] EPFL. e-puck EPFL Education robot. <http://www.e-puck.org/>.
- [13] B. Gerkey, K. Stoy, and R. T. Vaughan. Player Robot Server. Technical report, USC Robotics Labs, 22 November 2000.

**Metodología Para la Transferencia Tecnológica en la Industria Electrónica Basada en Software Libre y Hardware Copyleft XVII Workshop de Iberchip 2011.**

# Metodología Para la Transferencia Tecnológica en la Industria Electrónica Basada en Software Libre y Hardware Copyleft

Carlos I. Camargo Bareño

Universidad Nacional de Colombia, Email: cicamargoba@unal.edu.co

**Abstract**—Los canales tradicionales para la transferencia tecnológica en el área del diseño de sistemas embebidos no han sido exitosos en los países en vía de desarrollo donde la plataforma tecnológica no está lo suficientemente desarrollada para absorber esta nueva tecnología y conocimientos, esto se debe en gran parte a que las empresas de base tecnológica son muy pequeñas, con bajos niveles de producción y utilizan metodologías de diseño desactualizadas; por otro lado, la dependencia hacia los productos asiáticos hace muy difícil que estas pequeñas empresas puedan crecer sin la protección del estado. Así mismo, existe una sobre-oferta de profesionales afines con la industria electrónica, una gran parte de ellos provienen de entidades poco consolidadas. La unión de estos factores genera una tasa de desempleo muy alta, salarios bajos, aumento de la dependencia a los productos extranjeros y una desconfianza hacia los productos generados localmente; lo que afecta de forma considerable el número de estudiantes que ingresan a los programas de formación relacionados con la electrónica, llegando hasta el punto del cierre de programas acreditados.

Este artículo presenta una metodología para la transferencia tecnológica en el diseño de sistemas embebidos desarrollada en el Departamento de ingeniería eléctrica y electrónica de la Universidad Nacional de Colombia; esta metodología tiene como pilares el conocimiento como bien común, el movimiento de software libre y un concepto nuevo desarrollado en conjunto con un grupo de desarrolladores hardware y software: el *hardware copyleft*.

**Index Terms**—Sistemas Embebidos, educación en ingeniería, hardware copyleft.

## I. INTRODUCCIÓN

La transferencia de tecnología ha introducido técnicas de alta productividad y en muchos casos cambios técnicos en países menos desarrollados. La adquisición de tecnología foránea contribuye a mejorar la competitividad en los mercados locales e internacionales en estos países, en los que debe ser considerada como un proceso vital. Este proceso presenta problemas cuando se pierde capacidad de absorción por parte del país receptor y la renuencia del país que transfiere a transferir tecnología real y el *know-how*. Por lo que es necesario que estos países promuevan sus capacidades tecnológicas con el fin de absorber las tecnologías foráneas de forma eficiente en función de sus necesidades locales y de esta forma generar un rápido proceso de industrialización.

La transferencia de tecnología según Van Gogh involucra la adquisición de "actividad Inventiva" por parte de usuarios secundarios. Es decir, la transferencia tecnológica no involucra necesariamente maquinaria o dispositivos físicos; el

conocimiento puede ser transferido a través de entrenamiento y educación, y puede incluir temas como manejo efectivo de procesos y cambios tecnológicos [1]. No debe confundirse la transferencia tecnológica con la apropiación de tecnología que se define como el proceso de interacción con la tecnología, la modificación de la forma como es usada y el marco social dentro del cual es usada. Un ejemplo de apropiación de tecnología lo podemos encontrar en la telefonía celular, nuestras sociedades han cambiado drásticamente su forma de comunicarse y han generado nuevas actividades alrededor de esta tecnología, los usuarios pueden generar aplicaciones que adicionan funcionalidades y servicios.

### A. Tecnología

La tecnología es definida como el factor más significativo para mejorar la productividad, calidad y competitividad [2] y puede verse como un proceso de transformación que tiene como entrada recursos naturales, bienes, o productos semi-manufacturados y como salida se obtienen bienes consumibles de capital y semi-manufacturados. El *Technology Atlas team* identifica cuatro componentes de la tecnología [1]: *Techno-ware* relacionado con objetos, herramientas, equipos, máquinas, vehículos, facilidades físicas, instrumentos, dispositivos y fábricas; *Human-ware* relacionado con personas, habilidades en conocimiento experimental, sabiduría y creatividad, experiencia, competencia; *Info-ware* Relacionado con la información, incluye todo tipo de documentación y datos acumulados relacionados con especificación de procesos, procedimientos, diseños, teorías, y observaciones; *orga-ware* relacionado con la organización, acuerdos y alianzas necesarias para facilitar la integración de los componentes técnico, humano y de información.

El uso efectivo de estos cuatro componentes requiere el cumplimiento de ciertas condiciones. El componente técnico requiere de personal con habilidades específicas para poder ser utilizado. Para mejorar la eficiencia del sistema el componente humano necesita de adaptación y motivación. A medida que la organización cambia para adaptarse a nuevas condiciones o requerimientos se debe actualizar el sector de la información. No es posible realizar operaciones de transformación ante la ausencia de uno de estos cuatro componentes. La tecnología se encuentra fuertemente relacionada con un espectro amplio de las necesidades humanas, las condiciones físicas existentes o por factores culturales derivados de las especificidades

históricas de diferentes grupos sociales [3].

### B. Transferencia tecnológica

Odedra [4] define la transferencia tecnológica como el problema de transferencia de conocimiento (o know-how) sobre un número de aspectos (que incluyen el conocimiento) sobre como funciona un determinado sistema, como operarlo y desarrollar sus aplicaciones, como mantenerlo y si es necesario, como producir sus componentes y implementar un sistema similar. La transferencia tecnológica se considera exitosa cuando los receptores de la tecnología asimilan los conceptos anteriores para suplir sus necesidades locales.

Según Jolly [5] La innovación tecnológica es entendida como un nuevo método, medio o capacidad del individuo para realizar una determinada actividad. El resultado de la transferencia tecnológica puede ser la aceptación de una práctica común en otros lugares, o la aplicación de una técnica diseñada para otro uso en la solución de problemas locales. La transferencia tecnológica incluye la difusión de conocimiento científico y la preocupación por la transformación del conocimiento en innovaciones útiles. El conocimiento es lo que queda al final de un proceso documentado y difundido de forma apropiada. Para que la transferencia tecnológica sea exitosa es necesario transferir los componentes de la tecnología.

1) *Tipos de Transferencia Tecnológica*: Mansfield [6] clasifica la transferencia tecnológica en *transferencia de material*: artefactos tecnológicos, materiales, productos finales, componentes, equipos; *transferencia de diseño*: diseños, proyectos, know-how para fabricar productos diseñados previamente, los productos son copiados para producirlos localmente (ingeniería inversa); *transferencia de capacidades*: proporciona know-how y software no solo para fabricar componentes existentes, sino para innovar y adaptar tecnologías existentes para generar nuevos productos. La transferencia de material no constituye una transferencia tecnológica real, ya que no genera el conocimiento necesario para transformarlos y generar nuevos productos que cumplan con las necesidades locales. La transferencia de diseños permite adquirir mayor conocimiento sobre la tecnología transferida, sin embargo, es necesario que el país receptor cuente con la plataforma tecnológica adecuada para absorber estos conocimientos, de lo contrario no se generarán nuevos productos y las actividades se limitarán al ensamblaje de productos pre-manufacturados. La transferencia de capacidades es ideal, ya que proporciona las herramientas necesarias para que la transferencia sea exitosa, está asociada a una transferencia de conocimiento, lo cual es vital para entender plenamente la tecnología, mejorando las habilidades de los profesionales del receptor, creando una demanda de bienes y servicios relacionados con el conocimiento transferido; lo que se traduce en generación de empleo y aumento del bienestar general.

### C. Canales para la transferencia de tecnología

Grimpe y Hussinger [7] clasifican los mecanismos en *Formales*: acuerdos de licenciamiento, inversión extranjera,

compañías conjuntas, acuerdos de cooperación en investigación, arreglos de producción conjunta e *Informales*: No involucran acuerdos entre las partes y son difíciles de detectar y monitorear, por ejemplo, exportación de productos tecnológicos o bienes de capital, ingeniería inversa, intercambio de personal técnico y científico, conferencias de ciencia y tecnología, ferias y exposiciones, educación y entrenamiento realizado por extranjeros, visitas comerciales, literatura abierta (artículos, revistas, libros técnicos), espionaje industrial. Adicionalmente, existe una división basada en la naturaleza de la institución que proporciona los recursos para que se realice la transferencia, la institución puede ser de carácter *Abierta*: en donde la tecnología y el conocimiento son considerados bienes públicos, no existen restricciones para acceder a la información necesaria para adquirir, usar y transformar estos conocimientos en productos comerciales, y su éxito radica en obtener la máxima difusión posible para que los usuarios de este conocimiento mejoren el material existente y contribuyan a su crecimiento con experiencias personales; *Cerrada* La tecnología y el conocimiento se genera para fines privados, la utilización de este conocimiento esta sometida a acuerdos comerciales, no es posible entender las bases de la tecnología, por lo que no se pueden generar productos derivados.

Las actividades realizadas durante este estudio están enmarcadas dentro del concepto: *El conocimiento es un Bien Común*, toda la documentación necesaria para reproducir, entrenar, entender y modificar los productos generados se encuentran disponibles en servidores públicos [8] [9] y se proporciona soporte a través de listas de discusión, adicionalmente se proporciona soporte comercial para permitir la producción de estas modificaciones. A continuación se realiza una descripción de los canales más utilizados para la transferencia de tecnología y conocimiento en países en vía de desarrollo ([10] [4] [11]) indicando en cada caso sus ventajas, limitaciones y desventajas.

1) *Adquisición de IT*: La adquisición de equipo ha sido uno de los mecanismos de transferencia más importantes para los países en desarrollo. Estos equipos se entregan con el software requerido para su funcionamiento con lo que no es necesario que los usuarios generen aplicaciones, por lo que solo adquieren el conocimiento necesario para utilizar estas máquinas, y por lo tanto no saben como funcionan. Con la venta de equipos se transmite únicamente el conocimiento para operar, programar o mantener, sin embargo, este conocimiento sobre el sistema puede ayudar a concientizarse sobre la tecnología e impulsar la formación de capital humano.

La experiencia de países que lograron un rápido desarrollo económico e industrial muestra que la adquisición de una gran cantidad de tecnología foránea jugó un papel importante en este proceso. Colombia ha realizado un proceso de transformación tecnológica pero no ha diseñado políticas efectivas y eficientes para la transferencia de tecnologías de alto nivel.

2) *Educación y Entrenamiento*: Educar a las personas a través de cursos y entrenamiento en el país y enviándolas al extranjero para otros estudios es una forma de adquirir know-how sobre nuevas tecnologías. Por otro lado, muchas institu-

ciones que ofrecen carreras en ingeniería electrónica, ciencias de la computación y afines, utilizan modelos pedagógicos copiados de países desarrollados, los que no han sido adaptados plenamente a la infraestructura tecnológica local, y no es raro encontrar estudiantes que al finalizar sus estudios no están satisfechos con su profesión [4]. No se presenta una transferencia tecnológica exitosa cuando estudiantes formados en el exterior no pueden aplicar sus conocimientos en su país de origen, por lo que es necesario crear políticas que definen que áreas de estudio son prioritarias para el país.

Las multinacionales también ofrecen cursos de capacitación, sin embargo, se limitan al uso de sus productos, creando dependencia hacia sus herramientas. Adicionalmente, existen centros privados de capacitación que ofrecen cursos para el manejo de paquetes y lenguajes de programación, estos centros aprovechan la falta de centros de enseñanza tecnológica y personal calificado para cobrar altas sumas de dinero, lo cual limita el acceso a la información. Programas académicos inapropiados, acceso limitado a libros y computadores, falta de facilidades para capacitación, reduce la efectividad de la educación y capacitación como canal para la transferencia tecnológica.

3) *Asistencia Técnica*: La ventaja de contratar consultores externos radica en el ahorro de tiempo y dinero, ya que, utilizar personal local implicaría un gran esfuerzo y posiblemente se tendrían que asumir errores costosos en el proceso. Sin embargo, no es bueno confiar a consultores externos la responsabilidad de construir habilidades locales, ya que reduce el desarrollo de estas habilidades, especialmente, la del personal encargado de manejar proyectos. En algunas ocasiones los consultores no están familiarizados con las condiciones y requerimientos locales, por lo que diseñan soluciones que no se ajustan perfectamente a las necesidades, lo que significa que el sistema es sub-utilizado y la transferencia de tecnología es mínima. La falta de personal calificado hace que los consultores se encarguen de todas las tareas del proyecto, lo que aumenta su carga de trabajo y disminuye la posibilidad de entrenamiento de personal local [10].

4) *Licenciamiento*: El licenciamiento es un canal que se utiliza para transferencia de know-how sobre productos o procesos, es aplicado de forma individual o en combinación con otros instrumentos como investigación con inversión extranjera, importación de maquinaria o de técnicos. Sin embargo, esto no es efectivo si no se acompaña de habilidades administrativas y de producción. Adicionalmente, es necesario contar con una infraestructura tecnológica adecuada, capacidades locales de fabricación de hardware y software y políticas de gobierno adecuadas [11]. Un ejemplo de este tipo de práctica se presenta en el ensamblaje de dispositivos electrónicos, todos los componentes se importan completamente terminados y el dispositivo final es ensamblado probado y se carga la configuración inicial, no se producen actividades de ingeniería inversa con lo que se transmite muy poco conocimiento.

5) *Inversión Extranjera Directa*: La inversión directa de multinacionales es una forma de obtener tecnología externa. Esto asegura una rápida transferencia de información, pero

no necesariamente del entendimiento o know-how, lo que hace que la tecnología transferida a través de este canal sea mínima. Las grandes multinacionales pueden tener cierto control político en los países en vía de desarrollo, hasta tal punto que son asesores de instituciones encargadas de fijar políticas para la transferencia tecnológica [4]. El objetivo de la transferencia tecnológica debe ser el aumento de la autosuficiencia en el país receptor, unido a un uso compartido de recursos y experiencia entre los países desarrollados y en vía de desarrollo. La compra de equipo, plataformas de desarrollo o de software transfiere muy poco conocimiento sobre la tecnología, el servicio post-venta y el mantenimiento es realizado por el proveedor. Por otro lado, si las facilidades en educación y capacitación son limitadas se limita la formación de capital humano. La asistencia técnica utilizada para suplir la falta de personal especializado y ayudar con el proceso de transferencia no ha sido muy efectiva.

## II. METODOLOGÍA PARA LA TRANSFERENCIA TECNOLÓGICA EN EL ÁREA DE DISEÑO DE SISTEMAS EMBEBIDOS

En esta sección se describirán los pasos de una propuesta metodológica que tiene como objetivo permitir una transferencia tecnológica exitosa en el área de diseño de sistemas embobidos. Esta metodología está compuesta por los siguientes 7 pasos y actividades:

- **Elección:** Identificación del estado de la plataforma tecnológica existente para identificar facilidades y necesidades; identificación de niveles de complejidad de la tecnología; selección de una alternativa que pueda implementarse y de resultados a mediano y corto plazo con no muy altas inversiones de capital.
- **Adquisición:** Adquisición de plataformas adecuadas de desarrollo HW y SW; identificación de herramientas de desarrollo HW y SW y su origen (abierto/cerrado).
- **Adopción:** Utilización de plataformas de desarrollo para estudio de metodologías de diseño; uso de ingeniería inversa para entender la arquitectura, funcionamiento y programación de productos comerciales; utilización de productos comerciales para adaptarlos a problemas locales (Integración); tomar conciencia de la importancia del uso de esta tecnología.
- **Absorción:** Desarrollo o adaptación de metodologías de diseño y procesos de fabricación; desarrollo de productos tecnológicos propios; enseñanza de metodologías de diseño y procesos de fabricación en centros de educación consolidados.
- **Aplicación:** Desarrollo de soluciones a problemas locales; uso de metodologías de diseño en la concepción, diseño e implementación de sistemas digitales utilizando la tecnología; utilización de procesos de fabricación adaptados al entorno local; desarrollo de proyectos académicos utilizando esta tecnología.
- **Difusión :** Vinculación de la academia para incluir los conocimientos generados en los programas académicos de las carreras relacionadas; capacitación a la industria

local sobre el uso de la tecnología, las metodologías de diseño y procesos de producción; creación de una comunidad que utilice, mejore y aumente el conocimiento generado; hacer que el conocimiento generado en los pasos anteriores este disponible a todos los interesados; dar a conocer los procesos, productos y conocimientos creados a los generadores de políticas de estado.

- **Desarrollo:** Aumento de la demanda de productos, bienes y servicios relacionados; compra de maquinaria que permita la fabricación masiva de forma local; diseño de nuevos componentes (Circuitos Integrados, IPs, software CAD); creación de políticas gubernamentales que protejan la producción local; participación activa de la academia en la solución de problemas y en la formulación de políticas de gobierno relacionadas.

#### A. Elección y adquisición

1) *Niveles de complejidad de la tecnología:* Existen varias alternativas para la implementación de un sistema embebido: FPGA, sistema sobre silicio (SoC), SoC + FPGA y ASIC, la utilización de FPGAs o SoCs está determinada por el cumplimiento de restricciones temporales y funcionales, mientras que el uso de ASICs depende de el número de unidades producidas, se estima que a partir de 10 mil unidades se debe utilizar un ASIC para reducir los costos de producción. Debido a que los niveles de producción de los países en vía de desarrollo no son muy grandes, se abordará el problema de la transferencia utilizando FPGAs y SoCs comerciales, sin descuidar el estudio e implementación de ASICs. Adicionalmente, la inversión necesaria para construir un circuito integrado es muy alta, y puede ser considerada como un punto final.

Un proyecto reciente promovido por la unión Europea llamado Iberchip empezó desde hace 17 años un proceso de transferencia tecnológica en el diseño circuitos integrados de aplicación específica (ASICs) hacia los países iberoamericanos; a pesar de que Colombia era uno de los nodos principales no se logró fabricar ningún circuito integrado para aplicaciones comerciales; lo que era de esperarse ya que la mayoría de empresas en Colombia son importadores y distribuidores de productos tecnológicos y carecen de Departamentos de I+D; por otro lado, las pocas empresas de base tecnológica se encuentran muy atrasadas y dependen de productos comerciales para desarrollar sus productos, esto, unido a los bajos niveles de producción hace innecesaria la fabricación de un ASIC.

2) *Diagnóstico de la Industria Local:* Para determinar el estado de la industria electrónica en Colombia, se creó la empresa **emQbit LTDA.** en asociación con profesionales en ingeniería de sistemas, ingeniería eléctrica e ingeniería electrónica. Esta empresa desarrolló una serie de proyectos y actividades que ayudaron a entender e identificar los siguientes obstáculos para el desarrollo y comercialización de sistemas digitales: Falta de proveedores de bienes y servicios relacionados con la actividad (venta de dispositivos semiconductores, fabricación de placas de circuito impreso, montaje automático de componentes, etc); desconocimiento

de la tecnología (alcances y limitaciones) debido al uso de tecnologías y metodologías de diseño obsoletas; competencia con productos asiáticos de muy bajo costo; falta de confianza en los productos nacionales; desconexión de la academia con el sistema productivo; inexistencia de reglamentación de la industria de manufactura electrónica; profesionales con pocos conocimientos en procesos de diseño y fabricación. que coincide con los resultados de estudios consultados [10] [12] [13] [14] [15] [16].

3) *Diagnóstico de la Academia:* La tendencia moderna en los programas académicos a la utilización de herramientas de alto nivel para la enseñanza en áreas afines al desarrollo de dispositivos digitales [17] ocasiona que los profesionales no adquieran las habilidades necesarias para completar la cadena concepción - diseño - implementación y operación, en la mayoría de los casos se generan habilidades para la concepción y el diseño a alto nivel y dejan los otros pasos en manos de herramientas especializadas y/o a empresas asiáticas. Esta situación resulta la más atractiva desde el punto de vista económico, ya que no es necesario adquirir maquinaria costosa ni contratar personal calificado para operarlas; sin embargo, limita la generación de empleo local a personas con un nivel de formación alto [18] generando desempleo en las personas menos capacitadas. Según John Hall presidente y CEO de Linux International “ algunas facultades preparan a la gente en el uso de productos en vez de tecnologías de nivel básico” [17]. Esta situación unida al abandono de la implementación hace que la dependencia con las empresas manufactureras asiáticas aumente cada vez más.

Por otro lado, en muchas instituciones educativas se utilizan tecnologías y metodologías de diseño obsoletas (Familias 74XXX o 40XXX, lenguaje ensamblador, mapas de karnaugh), esto unido a programas académicos centrados en el análisis y no el diseño, donde el paso final es la simulación y el personal docente no tiene ninguna experiencia en el sector productivo; origina una deficiencia de habilidades necesarias para realizar el proceso completo para el diseño de dispositivos, lo que se traduce en profesionales que no disponen de las herramientas necesarias para resolver los problemas del país y al mismo tiempo competir con los productos asiáticos.

4) *Adquisición:* En la actualidad es muy fácil adquirir productos implementados con tecnología de punta, existe una gran variedad de plataformas de desarrollo y de dispositivos comerciales a los que se les puede aplicar ingeniería inversa para entender y modificar su funcionamiento. Al comienzo de este estudio se trabajó con la consola de juego GameBoy Advance, la cual posee un ASIC basado en un procesador ARM7.

#### B. Adopción

En el proceso de adopción se utilizaron productos desarrollados en otros países para suprir necesidades locales, en esta etapa se trabajó con las plataformas comerciales: Game Boy de Nintendo + Chamedlabs Xport (FPGA), la agenda Zaurus SL-5500, el dispositivo para redes sociales chumby y el portaretratos de Sungale ID800WT. Se utilizó la ingeniería inversa

para entender el funcionamiento y la arquitectura de un sistema embebido moderno, la metodología de diseño y como programar aplicaciones que cambien el funcionamiento de estos dispositivos. Se desarrollaron aplicaciones académicas como: Osciloscopio Digital utilizando FPAs [19], Automatización de un Puente grúa a escala [20], Control Adaptativo Embebido [21], Control de un horno de reflujo para componentes de montaje superficial [19], Dispositivo de visualización de variables que suministra el computador de un automóvil [19], herramienta para realizar reconfiguración parcial de FPGAs [22], evolución de un arreglo de células utilizando algoritmos genéticos [23] y comerciales como: Adquisición de datos para medición de calidad de energía; plataforma robótica didáctica; registro de aceleración de vehículos para compañías de seguros; sistema de seguimiento vehicular; monitor de signos vitales; diccionario basado en Wikipedia; menú electrónico y consola de juegos.

### C. Absorción y aplicación

La absorción es una actividad de aprendizaje que integra conocimiento que es nuevo para el país pero que no es nuevo para el mundo. En esta etapa se pasó de las plataformas comerciales al diseño de aplicaciones propias, para lo que se desarrollaron y/o adaptaron técnicas de fabricación al entorno local, se realizó la transferencia de los conocimientos adquiridos a la academia y se iniciaron contactos con empresas manufactureras locales y extranjeras. Se desarrollaron los productos académicos: Plataformas de desarrollo *hardware copyleft* para CPLDs, FPAs, Procesadores ARM, MIPS, Blackfin, LM32, Microblaze, Robótica, Linux Embebido, Codiseño HW/SW [24] [25] [26], [27], [28] así como la creación de un programa académico para la enseñanza de sistemas digitales utilizando tecnología de punta que crea las habilidades necesarias para concebir, diseñar, implementar y operar dispositivos digitales modernos y la definición del concepto *hardware copyleft* y su utilización como herramienta en la enseñanza de diseño de sistemas embebidos [29]. El proyecto SIE [8] proporciona la información recolectada durante la aplicación de este programa académico en las asignaturas del área de digitales en la Universidad Nacional de Colombia; una encuesta realizada a estos estudiantes muestra que ellos encuentran la metodología adecuada para generar habilidades necesarias para el desarrollo de aplicaciones comerciales, que se proporcionan las herramientas necesarias para lograr el objetivo final, y son conscientes de que se busca que ellos generen productos novedosos y de esta forma generar empleo; adicionalmente se muestra que los estudiantes no tienen ningún problema en utilizar herramientas licenciadas de forma ilegal y que prefieren el uso de las herramientas libres aunque su utilización sea mas compleja.

Adicionalmente, se desarrollaron los siguientes productos comerciales: Control de tornos industriales, plataforma robótica didáctica, monitoreo de Temperatura, sistema de seguimiento vehicular, sistema de medición de la calidad del suministro de energía eléctrica, monitor de signos vitales (UNAL), sistema de comunicación encriptada utilizando el

canal GSM (MICROENSAMBLE), switch de 4 canales de radio frecuencia (TESAMERICA).

### D. Difusión

Con las etapas previas se adquirieron los conocimientos necesarios para concebir, diseñar, implementar y operar dispositivos digitales, y la experiencia necesaria para realizar la producción a grandes escalas. La etapa de difusión (se está realizando en el momento de escribir este artículo) busca hacer llegar este conocimiento a todos los interesados, y de esta forma crear una comunidad que lo utilice como un recurso común; el cual proporciona a los centros de formación un programa académico actualizado que permite generar en los estudiantes las habilidades necesarias para innovar y generar empleo y a la industria le suministra herramientas que puede utilizar para desarrollo de nuevos productos comerciales y para la capacitación de su recurso humano. Ahorrando en ambos casos el tiempo necesario para la utilización efectiva de esta tecnología (valorada en 5 años) y el dinero asociado a este estudio. Esto es una consecuencia de la filosofía del movimiento FOSS y la iniciativa *hardware copyleft*.

El proceso de difusión se realiza en varios frentes, en el plano académico se realizaron conferencias y cursos de actualización donde se presentan las plataformas *hardware copyleft* desarrolladas y el programa académico de las asignaturas asociadas con el área de la electrónica digital, hasta el momento 4 de las principales universidades públicas del país están implementando este programa; en el plano industria se realizó una alianza entre el SENA, (la entidad de formación técnica más grande del país) la Universidad Nacional de Colombia, (la universidad más importante del país) Sharism at work LTDA, (empresa constituida en HONG KONG especializada en la manufactura de dispositivos digitales) empresas de base tecnológica y universidades de diferentes regiones del país; esta alianza busca difundir los conocimientos adquiridos durante 5 años de desarrollo e investigación, para ello se creará una plataforma *hardware flexible* que cumpla con las condiciones del *hardware copyleft* y que permita la implementación de aplicaciones comerciales en diferentes áreas. Durante el desarrollo de este proyecto las empresas participantes propondrán ideas para la realización de productos comerciales que den solución a problemas de su región, se vincularán estudiantes de los centros de formación que estén realizando su trabajo de grado para formar un equipo que trabaje en la construcción de un dispositivo que implemente estas ideas basado en la plataforma diseñada previamente.

El primer paso de este proceso consiste en la capacitación de las empresas y los centros de formación en la utilización de software abierto y *hardware copyleft* como herramientas de desarrollo de sistemas embebidos, para esto se utilizará el material generado en los cursos de la Universidad Nacional; a continuación se diseñará la plataforma de desarrollo flexible, este proceso será guiado por la empresa Sharism y por la Universidad Nacional, una vez finalizado el diseño, se realizarán los primeros prototipos y las pruebas correspondientes, una vez corregidos los posibles errores se procederá a la realización

de 100 unidades de la plataforma base, esto con el fin de simular una producción a *gran escala*. Se entregará a cada grupo de trabajo varias de estas unidades para que desarrollen su aplicación con la ayuda de todos los grupos de trabajo.

Todo el proceso se documentará en un servidor de libre acceso, esto con el fin de hacerlo accesible a cualquier interesado, se suministrarán los archivos necesarios para reproducir la capacitación y la plataforma de desarrollo. Cada grupo de trabajo está encargado de llevar una *bitácora* donde se detalla el proceso de diseño y de suministrar los archivos necesarios para reproducir el producto final.

### III. CONCLUSIONES Y TRABAJO FUTURO

Este artículo presenta los resultados de la aplicación de una metodología que permite una transferencia tecnológica exitosa en el área de diseño de sistemas embebidos, obteniendo como resultados de este proceso la capacitación a empresas de base tecnológica en el uso de herramientas y metodologías de diseño modernas basadas en software libre y hardware copyleft; la actualización del contenido de las asignaturas relacionadas en los centros de formación participantes; la posibilidad de generación de empleo a diferentes niveles (de servicios, técnico, profesional, comercial, administrativo); la adopción y apropiación de esta tecnología por parte del sector productivo.

A futuro se debe monitorear el cluster semilla para observar la evolución de los productos comerciales generados, así como observar los efectos que generan el cambio introducido en los programas académicos, los participantes deben estar encargados de cuidar, mejorar y aumentar el recurso común, procurando difundirlo para aumentar el número de usuarios/beneficiarios.

Una vez alcance un nivel de producción adecuado, se debe dar el siguiente paso: la construcción de un ASIC que permita reducir los costos de producción, para esto se está comenzando a trabajar con el diseño de un SoC abierto, que permita su modificación, en el momento de escribir este artículo se está trabajando con el softcore Mico 32 de lattice (lm32), se están generando IPs y se está estudiando su arquitectura con el fin de mejorarlala e implementarla, la siguiente etapa de este proceso es el *desarrollo* de la tecnología, proporcionando productos nuevos que no tengan ningún tipo de patente o restricción de uso. Por lo que en un futuro no muy lejano estaremos anunciando la fabricación y disponibilidad de el primer ASIC abierto diseñado en Colombia.

### IV. BIBLIOGRAFÍA

#### REFERENCES

- [1] F. Bar, F. Pisani, and M. Weber. Mobile technology appropriation in a distant mirror: baroque infiltration, creolization and cannibalism. *Seminario sobre Desarrollo Económico, Desarrollo Social y Comunicaciones Móviles en América Latina*. Buenos Aires, April 2007.
- [2] Goel Cohen. *Technology transfer: strategic management in developing countries*. Sage Publications inc, 2004.
- [3] K. Goel and Sayers B. Modelling Global-Oriented Energy Technology Transfer to DCs. *Sixth Global Warning International Conference, San Francisco*, 1995.
- [4] M. odedra-straub. The Myths and Illusions of Technology Transfer. *IFIP World Congress Proceedings*, August 1994.
- [5] James A. Jolly. The Technology Transfer Process: Concepts, Framework and Methodology. *The Journal of Technology Transfer*. Springer, 1977.
- [6] E. Mansfield. East-West technological transfer issues and problems, international technology transfer: Forms, resource requirements, and policies. *American Economic Review*, 65(2), 1975.
- [7] C. Grimpe and K. Hussinger. Formal and Informal Technology Transfer from Academia to Industry: Complementarity Effects and Innovation Performance. *Centre for european economical research*, 2008.
- [8] C. Camargo. Proyecto SAKC. URL:<http://en.qi-hardware.com/wiki/SAKC>.
- [9] C. Camargo. ECB\_AT91 y ECBOT Plataformas Abiertas para el desarrollo de Sistemas Embebidos. URL: <http://wiki.emqbit.com/free-ecb-at91>.
- [10] M. Odedra. *Information Technology Transfer to Developing Countries: Case studies from Kenya, Zambia and Zimbabwe*. PhD thesis, London School of Economics, 1990.
- [11] M. Odedra. Information Technology Transfer to Developing Countries Is it really taking place? *The 4th IFIFT9 International Conference on Human Choice and Computers, North Holland, Amsterdam, Netherlands, HCC 4 held jointly with the CEC FAST Programme*., 1991.
- [12] Innovation Associates Inc. Technology Transfer and Commercialization Partnerships Executive Summary.
- [13] M. Duque and A. Gauthier. Formación de Inegniers para la Innovación y el Desarrollo Tecnológico en Colombia. *Revista de la Facultad de Minas - Universidad Nacional de Colombia*, December 1999.
- [14] D Zuluaga, S Campos, M Tovar, R Rodríguez, J Sánchez, A Aguilera, L Landínez, and J Medina. Informe de Vigilancia Tecnológica: Aplicaciones de la Electrónica en el Sector Agrícola. Technical report, COLCIENCIAS, 2007.
- [15] M. Tovar and R. Rodríguez. PROSPECTIVA Y VIGILANCIA TECNOLÓGICA DE LA ELECTRÓNICA EN COLOMBIA. Master's thesis, Universidad Nacional de Colombia, 2007.
- [16] Héctor Martínez. Apropiación de conocimiento en Colombia. El caso de los contratos de importación de tecnología. *Revista Cuadernos de Economía*, 2004.
- [17] Jon Hall. POR GRANDES QUE SEAN...: ASEGURE EL FUTURO DE SU NEGOCIO. *Linux magazine*, ISSN 1576-4079(58):92, 2009.
- [18] A. Grove. How America Can Create Jobs. [http://www.businessweek.com/magazine/content/10\\_28/b4186048358596.htm](http://www.businessweek.com/magazine/content/10_28/b4186048358596.htm), May 2010.
- [19] C. Camargo. Implementación de Sistemas Digitales Complejos Utilizando Sistemas Embebidos. *Memorias del XI Workshop de Iberchip ISBN 959-261-105-X*, 2005.
- [20] I. Castillo, C. Camargo, and C. Perez. Automatización de un puente grúa a escala, mediante una plataforma embebida la cual soporta multiprogramación. *XII Workshop Iberchip*, 2006.
- [21] F. Pedraza, F. Segura, C. Camargo, and A. Gauthier. Control Adaptativo Embebido. *Memorias del XI workshop de Iberchip ISBN 959-261-105-X*, 2005.
- [22] C. Camargo and O. Sanchez. Linux embebido como herramienta para realizar reconfiguración parcial. *XII Workshop Iberchip*, 2006.
- [23] J. Espinosa, F. Segura, and C. Camargo. Evolución de un Arreglo de Celulas Utilizando Algoritmos Genéticos. *Memorias del XI Workshop de Iberchip ISBN 959-261-105-X*, 2005.
- [24] C. Camargo. ECBOT y ECB\_AT91 Plataformas Abiertas para el Diseño de Sistemas Embebidos y Co-Diseño HW/SW. *VIII Jornadas de Computación Reconfigurable y Aplicaciones*, Madrid España, September 2008.
- [25] C. Camargo. ECBOT: Arquitectura Abierta para Robots Móviles. *VII conferencia Iberoamericana en Sistemas, Cibernética e Informática*, 2008.
- [26] C. Camargo. ECBOT y ECB\_AT91 Plataformas Abiertas para el Diseño de Sistemas Embebidos y Co-diseño HW-SW. *VIII Jornadas de Computación Reconfigurable y Aplicaciones*, 2008.
- [27] C. Camargo. ECBOT: Arquitectura Abierta para Robots Móviles. *IEEE Colombian Workshop on Circuits and Systems*, 2007.
- [28] C. Camargo. First Colombian Linux SBC runs Debian. URL: <http://www.linuxfordevices.com/c/a/News/First-Colombian-Linux-SBC-runs-Debian/>, 2006.
- [29] C. Camargo. Hardware copyleft como Herramienta para la Enseñanza de Sistemas Embebidos. *Simposio Argentino de Sistemas Embebidos*, 2011.

**SIE: Plataforma Hardware copyleft para la Enseñanza de Sistemas Digitales XVII**  
Workshop de Iberchip 2011.

# SIE: Plataforma Hardware *copyleft* para la Enseñanza de Sistemas Digitales

Carlos Iván Camargo – Universidad Nacional de Colombia  
cicamargoba@unal.edu.co

**Abstract**—El gran avance de las técnicas de fabricación de Circuitos Integrados ha permitido que los sistemas embebidos sean parte fundamental de nuestra vida, aún sin darnos cuenta diariamente interactuamos con decenas de ellos; esto unido a la disponibilidad de herramientas software de desarrollo gratuitas (compiladores, simuladores, bibliotecas, Sistemas Operativos, herramientas CAD) abre grandes posibilidades comerciales para países en vía de desarrollo ya que no son necesarias grandes inversiones de capital para la concepción, diseño, y fabricación de estos sistemas.

Los dispositivos *hardware copyleft* suministran la información necesaria para reproducir, modificar y construir el dispositivo físico, son el complemento ideal del software libre y del código abierto; la unión software libre y hardware *copyleft* constituye una herramienta muy poderosa para la transferencia tecnológica y de conocimientos en el diseño de sistemas digitales ya que pueden ser utilizadas como referencia para nuevos productos o como material de estudio. En este artículo se presentan las características de una plataforma *hardware copyleft* y su aplicación en la enseñanza del diseño digital.

**Index Terms**—Educación en Ingeniería, Sistemas Embebidos, Hardware *copyleft*, Co-diseño HW/SW.

## I. INTRODUCCIÓN

En la actualidad estamos presenciando una tendencia global a delegar las tareas de manufactura de sistemas digitales a países asiáticos, donde la mano de obra calificada es abundante y barata; se presentan casos donde los creadores de una determinada tecnología no la desarrollan y dejan que estos países se beneficien de sus descubrimientos [1]. Esta situación se agrava a medida que las grandes empresas manufactureras asiáticas como Foxconn capturan la producción de los grandes diseñadores como Apple, Nokia, DELL, HP y Microsoft, lo que genera el cierre de empresas manufactureras a lo largo del mundo, con la consecuencia de pérdida de empleo masivo. En la actualidad según *Bureau of Labor Statistics* y *Thomson Financial Extel Company Report* Foxconn emplea a más personas que Apple, Dell, Microsoft, HP, Intel y Sony combinados; esta situación es más grave en países en vía de desarrollo, donde no existe la plataforma tecnológica para fabricar dispositivos digitales. Lo anterior lleva a preguntarse por la función y la situación de un profesional en áreas afines a la ingeniería electrónica en países donde no existe la capacidad de concepción y diseño.

La tendencia moderna en los programas académicos a la utilización de herramientas de alto nivel para la enseñanza en áreas afines al desarrollo de dispositivos digitales ocasiona que los profesionales no adquieran las habilidades necesarias para completar la cadena concepción - diseño - implementación y

operación, en la mayoría de los casos se generan habilidades para la concepción y el diseño a alto nivel y dejan los otros pasos en manos de herramientas especializadas y/o a empresas asiáticas. Esta situación resulta la más atractiva desde el punto de vista económico, ya que no es necesario adquirir maquinaria costosa ni contratar personal calificado para operarlas; sin embargo, limita la generación de empleo local a personas con un nivel de formación alto [1] generando desempleo en las personas menos capacitadas. Según Jon Hall presidente y CEO de Linux International “ algunas facultades preparan a la gente en el uso de productos en vez de tecnologías de nivel básico” [2]. Esta situación unida al abandono de la implementación hace que la dependencia con las empresas manufactureras asiáticas aumente cada vez más.

Según el ex-director ejecutivo de Intel Andy Grove [1] la solución está en hacer de la creación de empleo la política económica gubernamental más importante y hacer que las demás giren en torno a ella; además, es necesario volver a la producción interna con el fin de generar nuevos empleos, y volver a adoptar medidas que protejan la producción interna de los productos asiáticos. Sin embargo, para lograrlo es necesario crear en los profesionales las habilidades necesarias para implementar productos comercializables. Aunque la situación latinoamericana es diferente, podemos aprender de estas experiencias y seguir el camino trazado por Brasil, Argentina y Chile en la búsqueda de independencia tecnológica y generación de empleo.

### A. Flujo de diseño de sistemas embebidos

Los Sistemas Embebidos son sistemas heterogéneos que contienen componentes Software (microcontroladores, microprocesadores o DSPs) y Hardware (funciones implementadas en Dispositivos lógicos programables PLDs); por este motivo, es necesario adquirir habilidades en la utilización de lenguajes de programación como C o C++ para implementar las funciones software y Verilog o VHDL para la implementación de las tareas hardware; adicionalmente, deben conocer las diferentes formas de comunicación entre estos dos tipos de funciones.

En la figura 1 se muestran los conceptos que deben dominar los diseñadores de sistemas embebidos, y las tareas que deben realizarse para la concepción, diseño e implementación de un sistema embebido. En una gran parte de los programas académicos se estudian únicamente los temas relacionados con la concepción y diseño centrándose en las especificaciones funcionales del sistema, utilizando herramientas comerciales

o COTS (Commercial off-the-shelf) para su implementación. Nuestra propuesta se basa en la utilización de herramientas abiertas tanto hardware como software que permitan recorrer todo el proceso de concepción, diseño e implementación y de esta forma obtener un entendimiento integral del proceso sin generar dependencia a productos comerciales.

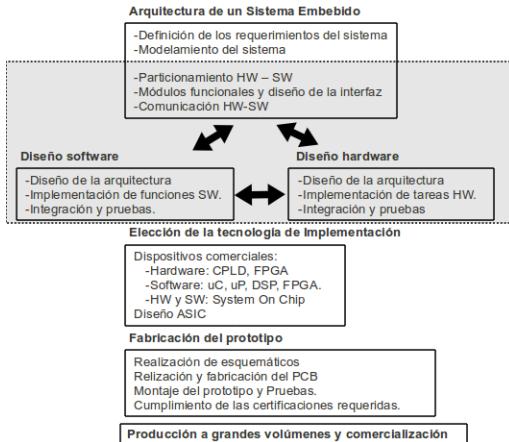


Fig. 1. Educación de sistemas embebidos. Tomada de: [3] y modificada

**1) Dominios de Diseño y Niveles de abstracción:** Existen tres dominios en los que se puede describir un sistema digital. **Funcional:** Describe el comportamiento funcional y temporal del sistema. **Estructural:** Describe su composición a partir de bloques básicos y **Físico** relacionado con la estructura física del sistema a nivel de circuito integrado o placa de circuito impreso [4]. Cada uno de estos dominios puede ser descrito utilizando diferentes niveles de abstracción. En el mercado existen herramientas que permiten entradas de diseño funcional utilizando especificaciones y algoritmos y de forma automática y optimizada generan representaciones en diferentes niveles de abstracción en los dominios estructural y físico. Desde el punto de vista comercial esto es muy útil ya que permite reducir el tiempo de diseño y los costos asociados. Sin embargo, no son muy adecuadas en la enseñanza de técnicas de diseño ya que proporcionan capas de abstracción que realizan ajustes a plataformas o a dispositivos comerciales ocultando el proceso completo al usuario; adicionalmente, su costo es muy elevado y en algunas ocasiones no se pueden adquirir de forma directa.

**2) Plataformas de Desarrollo Comerciales:** Existen en el mercado una gran variedad de plataformas de desarrollo que pueden ser utilizadas en la enseñanza de sistemas digitales, muchas de ellas proporcionan una gran variedad de recursos hardware y notas de aplicación con código fuente para ilustrar su programación y forma de uso. Sin embargo, presentan algunos de los siguientes inconvenientes:

- No es posible modificarlas: Lo que dificulta su uso en productos comercializables ya que es necesario construir placas sencillas adicionales para poder adaptarlas a un determinado problema.
- Los archivos de diseño de los esquemáticos y de la placa

de circuito impreso no están disponibles, o se utilizan herramientas CAD muy costosas.

- Algunos módulos son propietarios y no se suministra ningún tipo de información (esquemático, referencia de los componentes)
- No se suministra el código fuente: En el caso de los dispositivos lógicos programables se proporciona un netlist en formato propietario, en el caso de los procesadores se proporcionan archivos compilados.
- Utilizan componentes difíciles de conseguir o sujetos a acuerdos de confidencialidad.
- No se proporcionan contactos con las empresas manufactureras para fabricación de plataformas modificadas.

**3) Importancia del diseño e implementación de placas de circuito impreso:** El uso de plataformas comerciales para la implementación de dispositivos digitales en la academia hace que el estudiante no se preocupe por conocer, entender y realizar el proceso de concepción, diseño e implementación de una placa de circuito impreso, lo cual resulta muy importante para la aplicación de una serie de conceptos (capacidad de corriente, interferencia electromagnética, aislamiento, ruido, señales mixtas, velocidad de operación, niveles lógicos, etc); otro inconveniente que se presenta es que el estudiante se convierte en un usuario de tecnologías; una de estas habilidades es la "lectura" de esquemáticos, entender y comprender la función que realiza cada componente, como debe conectarse y si es compatible con otros componentes.

Observando la forma en que los estudiantes utilizan estas plataformas de desarrollo notamos que existe un desconocimiento parcial de su funcionamiento, cuando se ven obligados a realizar conexiones de dispositivos externos se cometen errores originados por el desconocimiento de parámetros de funcionamiento como velocidad o capacidad de manejo de corriente; incorrecta arquitectura o conexión; falta de comprensión de las hojas de especificaciones suministradas por los fabricantes.

Por lo anterior podemos decir que es muy importante para un estudiante realizar el ejercicio de concebir, diseñar e implementar una placa de circuito impreso, ya que esto ayuda a la creación de habilidades necesarias en su formación y ayudan a formar profesionales con un perfil de diseñador.

## II. PLATAFORMA HARDWARE COPYLEFT SIE

Antes de entrar a describir la estructura de la plataforma SIE se realizará una breve descripción del concepto *hardware copyleft*, que nace como una proyección al hardware del proyecto FOSS (Free and Open Source Software) [5], el cual constituye hoy en día la organización auto-gobernada más grande del mundo y es la responsable de la creación de una gran cantidad de herramientas software como el servidor web *Apache*, la suite para oficina *Openoffice*, el navegador *Mozilla* y el sistema operativo *Linux*.

En desarrollo hardware no existe un proyecto de estas dimensiones que permita utilizar desarrollos ya existentes para entender su principio de funcionamiento y realizar modificaciones para adaptarlas a la solución de problemas locales. En

la actualidad existen proyectos aislados como *Arduino*, *Beagle Board*, *chumby* y *NanoNote* [6] que son presentados como *open-source*, todos ellos proporcionan: Esquemáticos detallados en formato PDF; código fuente del software necesario para su operación; listas de discusión para soporte; tutoriales sobre su funcionamiento y programación; software de desarrollo; archivos de diseño: esquemático, layout; archivos para la fabricación de las placas de circuito impreso.

Salvo la plataforma *Arduino*, los costos necesarios para reproducir una de estas plataformas es muy elevado, esto se debe a las herramientas propietarias utilizadas en el diseño (10000 USD); las características de la placa de circuito impreso, 6 u 8 capas, vías de .2mm, (400 USD), el costo de los componentes (400 USD), y el montaje de encapsulados BGA, TQFP, SMD, y QFN (100). Esto ocasiona que los proyectos *open-source hardware* no presenten una gran dinámica en desarrollo hardware y las contribuciones se limitan a la fabricación de tarjetas de expansión para propósitos específicos. Desde el punto de vista comercial, estos proyectos son muy atractivos ya que permiten modificar un diseño validado, ahorrando mucho tiempo y dinero, desde el punto de vista académico, tener acceso a los archivos de diseño, permite estudiar las técnicas utilizadas en su diseño para poder aplicarlas en proyectos propios.

Del análisis anterior podemos notar que el mayor costo se produce al utilizar herramientas comerciales para el diseño, por esta razón una de las características del *hardware copyleft* es la utilización de software libre o gratuito durante todo el proceso de diseño; esto permite que muchas universidades, empresas de todo tipo y personas interesadas puedan beneficiarse del proyecto. Para que un dispositivo sea considerado *hardware copyleft* es necesario que permita su distribución, modificación y fabricación bajo un esquema de licencias que conserve la propiedad intelectual y obligue a sus productos derivados a utilizar el mismo tipo de licenciamiento.

La principal diferencia entre el hardware y el software es la implementación física, por este motivo, todo proyecto *hardware copyleft* debe proporcionar servicios de manufactura y suministro de componentes, por lo que debe estar asociado con empresas manufactureras que permitan la producción a diferentes niveles.

#### A. Proyecto SIE

El proyecto SIE [7] fué creado para satisfacer las necesidades de los desarrolladores de hardware permitiendo la creación de aplicaciones bajo la licencia Creative Commons BY - SA [8] la que permite la distribución y modificación del diseño (incluso para aplicaciones comerciales), con el único requisito de que los productos derivados deben tener la misma licencia y deben dar crédito al autor del trabajo original. Lo que constituye la base de los productos *hardware copyleft*.

Al ser inspirado en el movimiento FOSS, los dispositivos *hardware copyleft* comparten la misma filosofía [5], y son el complemento perfecto del software libre. Para que un dispositivo HW sea reproducible y modificable es necesario: suministrar los archivos necesarios para la fabricación, es decir, los esquemáticos y los archivos de la placa de circuito

impreso (preferiblemente para herramientas abiertas como Kicad o geda); la cadena de herramientas de compilación y depuración para desarrollo de aplicaciones; el código fuente de: el programa que inicializa la plataforma (*bootloader*), la herramienta que carga dicho programa en la memoria no volátil (*usbboot*), el sistema de archivos y aplicaciones (*openwrt*); documentación completa que indique como fué diseñada, construida, como utilizarla, desarrollar aplicaciones y tutoriales que expliquen el funcionamiento de los diferentes componentes. (Todo esto puede ser descargado del sitio del proyecto: <http://projects.qi-hardware.com/index.php/p/nn-usb-fpga/>). Adicionalmente, se debe contar con la posibilidad de fabricación y montaje, lo que constituye la principal diferencia entre el software y el hardware libre. Esto contrasta fuertemente con el movimiento de software libre, en donde no se requiere inversión de capital para modificar un proyecto existente. Por esta razón, deben existir varios niveles de libertad, un proyecto que utilice componentes costosos y de difícil consecución limitará su alcance a un sector determinado. El nivel de libertad más alto requiere bajas sumas de dinero para replicar y/o modificar el hardware, mientras los niveles de libertad bajos requieren grandes sumas de dinero; SIE fue diseñado para que pueda ser fabricado fácilmente por el que este interesado.

1) *Arquitectura*: La Figura 2 muestra el diagrama de bloques de la plataforma SIE, en ella podemos encontrar un procesador que posee periféricos para comunicación serial (UART), memorias micro-SD, un puerto I2C, un LCD a color de 3 pulgadas, 2 entradas y salidas de audio stereo, 2 entradas analógicas; una FPGA que proporciona 25 señales de entrada/salida digitales de propósito general (GPIOs) y controla un conversor analógico digital de 8 canales. Existen dos canales de comunicación entre la FPGA y el procesador: uno para controlar el puerto JTAG, lo que permite la configuración de la FPGA desde el procesador (lo que elimina la necesidad de cables de programación); y otro que proporciona el bus de datos, dirección y control para comunicarse con las tareas HW o periféricos implementadas en la FPGA. El procesador utilizado es un Ingenic JZ4725 (MIPS) corriendo a 400MHz, se dispone de una memoria NAND de 2GB para almacenamiento de datos y programas, así como de una memoria SDRAM de 32 MB, lo que permite la ejecución de una gran variedad de aplicaciones Linux.

2) *Comunicación con el usuario y programación*: SIE proporciona un canal de comunicación y alimentación a través del puerto USB-device, y es configurado para ser utilizado como una interfaz de red (*usb0*), permitiendo la transferencia de archivos y ejecución de una consola remota utilizando el protocolo *ssh*; este canal de comunicación también se utiliza para programar la memoria NAND no volátil, por lo que para realizar la programación completa de los componentes de la plataforma solo es necesario un cable USB. SIE posee un sistema de archivos basado en el proyecto *openwrt* y dispone de una gran cantidad de aplicaciones y librerías que pueden ser compiladas en un computador tradicional, siguiendo las instrucciones contenidas en la wiki del proyecto. Las aplica-

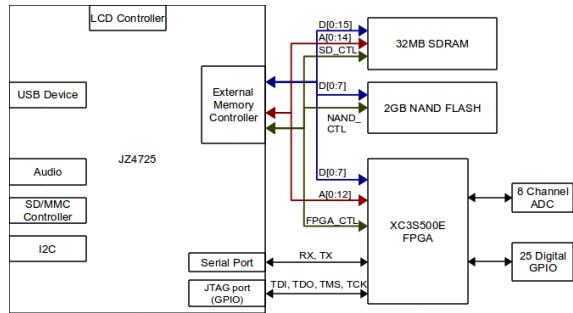


Fig. 2. Estructura de la plataforma de desarrollo SIE

ciones software se cross-compilan utilizando las herramientas suministradas por el proyecto *GNU-Linux*; para el componente hardware se utilizan las herramientas gratuitas suministradas por Xilinx. La figura 3 muestra el diagrama de flujo y las herramientas utilizadas para la implementación de las tareas software.

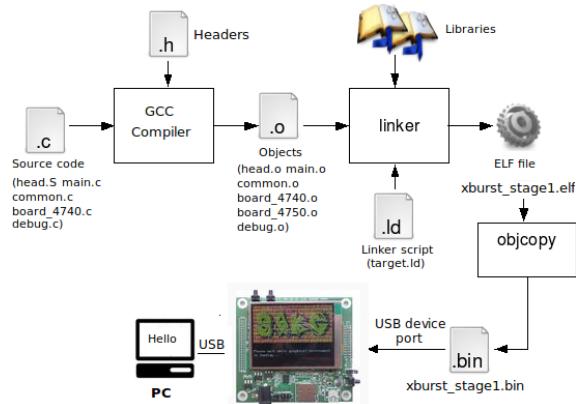


Fig. 3. Flujo de diseño software

3) *Especificaciones físicas*: Las dimensiones de SIE son 8cm de largo, 8 cm de ancho, 1cm de altura; su placa de circuito impreso es de dos capas, utiliza líneas de 0.2m, vías de 0.3mm de diámetro, los componentes se encuentran en una sola capa y son TQFP o SMD, no posee componentes BGA o QFN lo que facilita el montaje manual. Todo esto hace que sea posible la reproducción y modificación de esta plataforma a un precio muy bajo. El costo de fabricación de esta tarjeta se estima en 70 usd para 50 unidades. La figura 4 muestra la apariencia de la plataforma de desarrollo SIE.

#### B. Comunidad hardware copyleft

Una parte importante de este método de enseñanza es la filosofía del proyecto hardware copyleft, por esta razón, cada grupo debe hacer un aporte, suministrando la información completa del proceso de desarrollo y suministrando los archivos necesarios para replicar y/o modificarlo, esto es una consecuencia de la licencia CC-BY-SA.



Fig. 4. Plataforma de desarrollo SIE

La experiencia del proyecto FOSS indica muchos miembros de estas comunidades ingresan para suplir necesidades, pero muchos de ellos continúan creando código y prestando servicios a la comunidad porque disfrutan programar. Estos *aficionados* realizan un papel muy importante dentro de la comunidad encargándose de tareas como mejora de la plataforma tecnológica, re-escribiendo secciones de código, documentándolo, respondiendo preguntas, preservando o mejorando la arquitectura [9]. Las actividades de documentación además de contribuir a mejorar las habilidades de escritura de reportes técnicos ayudan a formar una comunidad que contribuye al crecimiento del proyecto copyleft hardware, los estudiantes ingresan a las listas de desarrolladores aprendiendo a utilizar una herramienta muy poderosa en la que pueden compartir sus inquietudes con miembros más experimentados y mientras participan ayudan a crear un banco de preguntas que pueden ser útiles para futuros miembros. Adicionalmente se obliga a expresarse en un idioma diferente.

Crear estos hábitos ayuda a que los jóvenes sean conscientes de su papel dentro de una comunidad y piensen que sus acciones pueden ayudarla o perjudicarla, los proyectos realizados por ellos podrán ser parte de los recursos de la comunidad (si la calidad del trabajo lo amerita) y pueden ser la continuación de un esfuerzo prolongado o el punto de partida de un nuevo conocimiento; la licencia CC-BY-SA garantiza que todos los trabajos derivados de este recurso serán parte del mismo, lo que permite su crecimiento, la labor de los estudiantes es vital para el uso del recurso común y puede crear miembros que en un futuro formularán políticas y reglas de uso del recurso. Por otro lado, participar en este tipo de proyectos permite crear reputación, la cual puede ser útil para establecer relaciones profesionales, de negocios o personales. El entorno académico es ideal para atraer nuevos miembros a la comunidad hardware copyleft, ya que se trabaja con jóvenes con deseos de ser parte de un grupo y de adquirir conocimientos. Desde el punto de vista comercial este recurso es muy atractivo ya que permite ahorrar mucho tiempo, esfuerzo y dinero para la creación de

nuevos productos. Por otro lado, el concepto de hardware copyleft es una herramienta poderosa para transferir tecnología y conocimientos a los países en vía de desarrollo donde la plataforma tecnológica no está lo suficientemente desarrollada.

### III. INTEGRACIÓN CON LA EDUCACIÓN

En la actualidad SIE está siendo utilizada en los cursos de arquitectura de computadores y sistemas embebidos de la Universidad Nacional de Colombia sede Bogotá (cerca de 120 estudiantes utilizando 45 placas), la metodología utilizada se centra en la realización de un proyecto durante el período académico, los estudiantes deben realizar el proceso de concepción, diseño e implementación de una aplicación que da solución a un determinado problema, cada grupo integrado por tres estudiantes está encargado de llevar una bitácora del proyecto en la wiki servidor del proyecto QI-hardware<sup>1</sup>, incluyendo toda la información generada durante el proceso de diseño (diagramas de flujo, diagramas de bloque, simulaciones) el único requisito es la construcción de una placa hija (diseñada en Kicad) con la funcionalidad que se le desea dar a la plataforma SIE.

#### A. Máquinas de Estados Algorítmicas

En el primer curso del área de diseño digital en la Universidad Nacional de Colombia, el objetivo es el estudio, diseño e implementación de Máquinas de Estado Algorítmicas utilizando metodologías de diseño como la presentada en [10]. SIE proporciona un canal de comunicación entre el procesador y el puerto JTAG de la FPGA, este canal es utilizado para su configuración y para realizar pruebas a baja frecuencia del circuito implementado en la FPGA. El proyecto SIE proporciona una herramienta basada en *urjtag* que recibe como entradas un archivo (con extensión *pad.csv* generado por la herramienta de Xilinx) con la función de cada uno de los pines de la FPGA y el resultado de la simulación en formato *vcd* (value change dump); utilizando la instrucción INTEST de la cadena JTAG, aplica vectores de prueba al circuito implementado en la FPGA, captura y grafica su respuesta en el LCD de la plataforma; esta herramienta puede verse como una combinación de un analizador lógico y un generador de vectores de prueba de bajo costo (ver figura 5).

#### B. Arquitectura de Computadores

En la página de descargas del proyecto SIE<sup>2</sup> se pueden obtener implementaciones en VHDL para el procesador MIPS MLITE y en verilog para el procesador de lattice LM32, se suministran los archivos necesarios para el desarrollo de aplicaciones software utilizando la cadena de herramientas GNU y los archivos para realizar la simulación del softcore utilizando *ghdl* e *icarus verilog*; A diferencia de Microblaze de Xilinx (suministra un formato tipo netlist que utiliza los bloques básicos de Xilinx) *plasma* y *lm32* permiten realizar modificaciones en la arquitectura del procesador, permitiendo

<sup>1</sup><http://en.qi-hardware.com/wiki/2010-II/es>

<sup>2</sup><http://projects.qi-hardware.com/index.php/p/nn-usb-fpga/source/tree/master/>

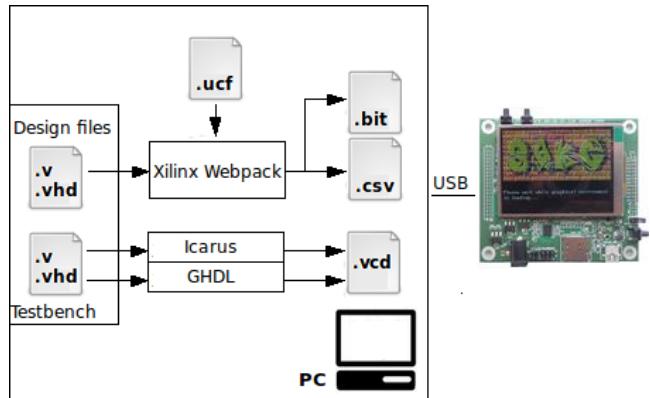


Fig. 5. Flujo de diseño hardware

la creación de nuevas instrucciones y co-procesadores. Adicionalmente, se proporcionan herramientas que inicializan las memorias internas de los procesadores softcore con el código generado por la cadena de compilación; lo que permite realizar un estudio completo del funcionamiento del procesador. En este curso se estudia la arquitectura de un sistema sobre silicio (SoC) y los estudiantes deben concebir, diseñar e implementar un periférico dedicado, conectarlo al SoC y escribir el código en C que lo controla e implementa la funcionalidad. La FPGA de SIE posee recursos limitados, lo que obliga a los estudiantes a optimizarlos y a buscar en el mercado circuitos integrados que les permita cumplir con la funcionalidad requerida.

En este curso el procesador es utilizado como herramienta de configuración de la FPGA, los archivos *.bit* son transferidos al sistema de archivos de SIE utilizando el protocolo *ssh* y desde allí son transferidos a la FPGA utilizando *xc3sprog* o *urjtag*. Ejecutando la aplicación *minicom* en el procesador de SIE, es posible utilizar el puerto serie del procesador implementado en la FPGA como canal de depuración; por lo que no es necesario utilizar programadores o aplicaciones externas.

#### C. Sistemas Embebidos

QI hardware, proporciona las herramientas software necesarias para el desarrollo de aplicaciones software en los procesadores MIPS de Ingenic JZ4720/25/40; en la actualidad existen aplicaciones abiertas para: programar las memorias no volátiles *usbboot*; inicializar la plataforma y cargar la imagen de linux. *u-boot* crear la imagen del kernel de Linux; crear un sistema de archivos *openwrt*; crear instaladores de un gran número de aplicaciones *opkg-openwrt*; desarrollo software utilizando librerías gráficas como QT o GTK. *openwrt*.

SIE utiliza un procesador JZ4725 y tiene una arquitectura compatible con el producto de QI-hardware *BEN NANONOTE*, por esta razón, todo el desarrollo realizado para BEN es aplicable a SIE, esto garantiza el mantenimiento, actualización y disponibilidad del software necesario para el desarrollo de aplicaciones.

En este curso se utiliza el procesador MIPS para ejecutar tareas de visualización, comunicación, control e interfaz con el usuario. Se utilizan librerías gráficas de alto nivel como QT (de Nokia) para realizar la interfaz, se desarrollan módulos del kernel y programas en espacio de usuario para el control de periféricos dedicados (implementados en la FPGA). Con esto se proporciona a los estudiantes herramientas que están siendo utilizadas en la actualidad por los grandes fabricantes de dispositivos digitales como Nokia, Dell, Hewlett Packard.

La figura 6 muestra la arquitectura de la etapa de comunicación entre el procesador y la FPGA, las señales *DATA*, *NCS2*, *we0\_n*, *address* provenientes del procesador no se encuentran en fase con la señal de reloj de la FPGA, por lo que deben ser sincronizadas con este; (el módulo SYNC se encarga de esta tarea). Debido a que no se pueden implementar buses tri-estado en el interior de la FPGA (sólo se pueden utilizar en los pines de la FPGA), por esta razón, los periféricos deben tener dos buses de datos uno de entrada (Color Naranja) y otro de salida (color verde). Se debe proporcionar un circuito a la salida de los buses de los periféricos que permita el paso de la información del periférico seleccionado (en la figura se representa este elemento como un Multiplexor). Debido a que la FPGA es mucho más rápida que el procesador, es necesario generar un pulso con la duración de un ciclo de reloj de la FPGA para la señal de escritura *we0\_n*; esto se hace para evitar que se realice más de una operación de escritura (esta tarea es realizada por el módulo Write Pulse Generator, en la figura puede verse el diagrama de tiempos que representa su funcionamiento). Un buffer tri-estado debe controlar el acceso al bus de Datos, este buffer debe presentar un estado de alta impedancia cuando no se selecciona ningún periférico o cuando se realiza una operación de escritura y debe permitir el paso de información desde el periférico seleccionado en las operaciones de lectura.

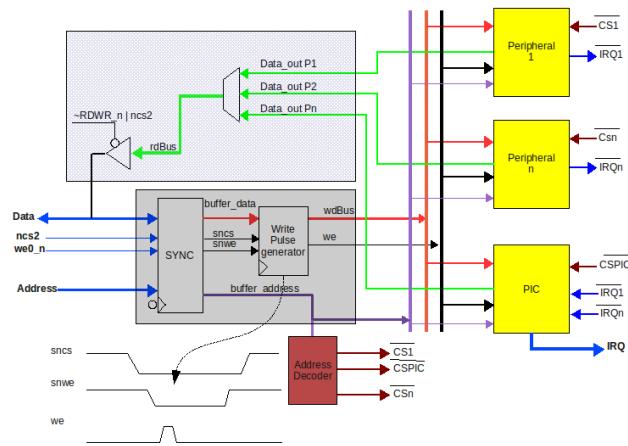


Fig. 6. Interfaz HW-SW en SIE

#### IV. CONCLUSIONES

SIE es una plataforma única en el mercado de muy bajo costo que proporciona todo lo necesario para desarrollar un

curso completo de diseño digital, reduciendo de forma considerable el equipo de medida necesario para la realización de las prácticas. Las actividades propuestas en las tres asignaturas del área tienen como objetivo generar en el estudiante las habilidades necesarias que le permitan diseñar sistemas digitales de diferente grado de complejidad, hasta llegar a un sistema que puede ser comercializable y satisfacer una necesidad de una determinada comunidad, con esto, se evita que el último paso en el proceso de enseñanza sea la simulación.

En este artículo se presentan los pasos necesarios para crear dispositivos comercializables (desde el punto de vista técnico), pero no se presentan los pasos necesarios para su comercialización como estudio de mercado; plan de negocios, financiación; esto será tratado en trabajos posteriores.

La utilización de herramientas abiertas permite que el estudiante conozca y controle los diferentes pasos de la metodología de diseño y sea capaz de ajustarlas para diferentes situaciones, esto hace que se adquiera un conocimiento sobre la tecnología sin crear dependencia hacia las herramientas comerciales.

La utilización de estas herramientas abiertas en los cursos del área de electrónica digital en la Universidad Nacional de Colombia han aumentado la calidad de los trabajos de grado de los estudiantes, pasando de la utilización de procesadores de 8 bits, implementación de prototipos en placas universales (perfboard) o en placas de pruebas (protoboard), lenguaje ensamblador, a aplicaciones que utilizan procesadores de 32 bits, tareas HW implementadas en FPGAs, sistemas operativos, implementadas en placas de circuito impreso con componentes de montaje superficial. Adicionalmente, se logró dar continuidad a los contenidos de las tres asignaturas haciendo que los estudiantes utilicen las herramientas adecuadas para dar solución a un determinado problema (en el pasado todo se resolvía con procesadores de 8 bits, o con FPGAs).

#### REFERENCES

- [1] A. Grove. How America Can Create Jobs. [http://www.businessweek.com/magazine/content/10\\_28/b4186048358596.htm](http://www.businessweek.com/magazine/content/10_28/b4186048358596.htm), May 2010.
- [2] Jon Hall. POR GRANDES QUE SEAN...: ASEGURO EL FUTURO DE SU NEGOCIO. *Linux magazine*, ISSN 1576-4079(58):92, 2009.
- [3] H. Mitsui, H. Kambe, and H. Koizumi. Use of Student Experiments for Teaching Embedded Software Development Including HW/SW Co-Design. *IEEE TRANSACTIONS ON EDUCATION*, 52(3), August 2009.
- [4] Gajski D.D., Abdi S., Gerstlauer A., and Schirner G. *Embedded System Design: Modeling, Synthesis, Verification*. Springer, 2009.
- [5] R. Stallman. Philosophy of the GNU project. URL: <http://www.gnu.org/philosophy/>, 2007.
- [6] Qi Hardware. Qi Hardware Copyleft Hardware Project. URL: <http://en.qi-hardware.com/>.
- [7] C. Camargo. Proyecto SAKC. URL:<http://en.qi-hardware.com/wiki/SAKC>.
- [8] Creative Commons. Licencias Creative Commons. URL: <http://creativecommons.org/licenses/>, 2004.
- [9] S. Shah. Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. *Management Science*, July 2006.
- [10] Luis Alejandro Cortés. *Verification and Scheduling Techniques for Real-Time Embedded Systems*. PhD thesis, Linköpings universitet Institute of Technology, 2005.

**Metodología para la Transferencia de Conocimientos en el Diseño de Sistemas Digitales**  
A publicar en la reunión nacional ACOFI 2011.

# METODOLOGÍA PARA LA TRANSFERENCIA DE CONOCIMIENTOS EN EL DISEÑO DE SISTEMAS DIGITALES

Carlos I. Camargo Bareño  
Universidad Nacional de Colombia  
Bogotá Colombia

## RESUMEN

Los canales tradicionales para la transferencia tecnológica en el área del diseño de sistemas embebidos no han sido exitosos en los países en vía de desarrollo donde la plataforma tecnológica no está lo suficientemente desarrollada para absorber esta nueva tecnología, esto debido a la escasa transferencia de conocimiento que pueda ser utilizado para generación de productos locales. Este artículo presenta una metodología para la transferencia tecnológica en el diseño de sistemas basada en el conocimiento como bien común, el movimiento de software libre y un concepto nuevo desarrollado en conjunto con un grupo de desarrolladores hardware y software: el *hardware copyleft*.

Palabras claves: Sistemas Embebidos, educación en ingeniería, hardware copyleft, transferencia tecnológica.

## Abstract

Traditional channels for technology transfer in embedded system design have not been successful in developing countries where the technology platform is not sufficiently developed to absorb this technology, this due to poor knowledge transfer can be used for generation of local products. This article presents a methodology for technology transfer in digital system design based in the knowledge as a common, free software movement and a new concept developed in conjunction with a group of hardware and software developers: the hardware copyleft.

Keywords: Embedded systems, engineering education, copyleft hardware, technology transfer.

## 1. Introducción

La transferencia de tecnología ha introducido técnicas de alta productividad y en muchos casos cambios técnicos en países menos desarrollados. La adquisición de tecnología foránea contribuye a mejorar la competitividad en los mercados locales e internacionales en estos países, en los que debe ser considerada como un proceso vital. Este proceso presenta problemas cuando se pierde capacidad de absorción por parte del país receptor y la renuencia del país que transfiere a transferir tecnología real y el *know-how*. Por lo que es necesario que estos países promuevan sus capacidades tecnológicas con el fin de absorber las tecnologías foráneas de forma eficiente en función de sus necesidades locales y de esta forma generar un rápido proceso de industrialización. La transferencia de tecnología involucra la adquisición de "actividad Inventiva" por parte de usuarios secundarios. Es decir, la transferencia tecnológica no involucra necesariamente maquinaria o dispositivos físicos; el conocimiento puede ser transferido a través de entrenamiento y educación, y puede incluir temas como manejo efectivo de procesos y cambios tecnológicos (Bar, 2007). En este trabajo utilizaremos al conocimiento como canal para la transferencia; entre más personas puedan

soportar la nueva tecnología, adaptándose a sus cambios y utilizándola para generar productos que den solución a problemas locales; mayores posibilidades de alcanzar una transferencia exitosa.

**Transferencia tecnológica:** (Odedra, 1994) define la transferencia tecnológica como el problema de transferencia de conocimiento (o know-how) sobre un número de aspectos (que incluyen el conocimiento) sobre como funciona un determinado sistema, como operarlo y desarrollar sus aplicaciones, como mantenerlo y si es necesario, como producir sus componentes y implementar un sistema similar. La transferencia tecnológica se considera exitosa cuando los receptores de la tecnología asimilan los conceptos anteriores para suplir sus necesidades locales.

## 2. El Conocimiento como bien público

El conocimiento en esta propuesta se refiere a las ideas intangibles; información y datos en el que el conocimiento es expresado u obtenido; y como el entendimiento adquirido a través de la experiencia o estudio científico, académico o no académico. La adquisición y descubrimiento de conocimiento es un proceso social y personal.

Ostrom (Hess, Ostrom, 2006) propone dos variables generales para clasificar los diferentes tipos de bienes: *el nivel de exclusión* entendido como la dificultad para excluir el acceso a un recurso por parte de usuarios potenciales; y *el nivel de rivalidad*, el cual se refiere al impacto que tiene el uso de un recurso por un individuo (o grupo) sobre el uso de otros usuarios. Ostrom, identifica cuatro tipos de bienes y/o servicios: *bienes públicos* bajo nivel de rivalidad y bajo nivel de exclusión; *bienes comunes* alto nivel de rivalidad y bajo nivel de exclusión; *club goods* bajo nivel de rivalidad y alto nivel de exclusión y *bienes privados* alto nivel de rivalidad y alto nivel de exclusión. A nuestro modo de ver, el conocimiento es un recurso con un grado de rivalidad bajo, ya que no se afecta negativamente al aumentar el número de usuarios, lo que nos deja la posibilidad de clasificarlo como un bien público o un bien tipo club. Idealmente, el conocimiento debería ser un bien público, esto es, no deben existir restricciones para acceder a él; sin embargo, en la actualidad el acceso al conocimiento es restringido, ya sea por medio de patentes, derechos de propiedad intelectual (lo cual es muy común en el desarrollo de nuevos productos y tecnologías) o por que el acceso al conocimiento tiene un costo que no puede ser pagado por cualquier miembro de la sociedad. En Colombia el acceso a la educación técnica, tecnológica y superior es limitado, se tiene una cobertura del 37%; aunque la cobertura en educación media aumentó de forma considerable, análisis realizados por varios profesores de la facultad de ingeniería de la universidad nacional demuestran que los conocimientos de los recién ingresados ha disminuido de forma considerable llegando al punto de tener que crear nuevos cursos para suplir deficiencias en su formación. La escasez de cupos en los centros de formación públicos y los altos costos de las matrículas en las universidades privadas (con niveles de calidad similares) han convertido al conocimiento en un bien tipo club (en Colombia); afectando de forma considerable la cantidad de personal con las capacidades necesarias para absorber los conocimientos asociados a nuevas tecnologías; adicionalmente, según el ministerio de educación nacional, el 64% de los matriculados cursan programas universitarios y solo el 24% en programas relacionados con ingeniería. Otra fuente de capacitación la suministra el único centro de desarrollo tecnológico de la industria electrónica CIDEI, a través de tres cursos: Diseño digital empleando dispositivos lógicos programables; diseño de software embebido para microcontroladores ARM y diseño de circuitos impresos con normas internacionales IPC; sin embargo, el elevado costo de estos cursos genera una barrera en la obtención de este conocimiento por parte de los interesados; adicionalmente, estos conocimientos no son suficientes para generar cambios significativos en el país ya que su difusión es limitada.

A nuestro modo de ver, basados en el estado de la industria electrónica nacional y de la capacidad del país para la formación de personal calificado, el principal problema que presenta la industria electrónica nacional es la falta de conocimientos sobre procesos de diseño y fabricación, debido en parte a la fuerte exclusión que se tiene al acceso de la información relacionada con estos procesos, lo que se traduce en la incapacidad de producción local de productos que cumplan con los estándares internacionales. Este trabajo pretende crear un recurso público basado en el conocimiento necesario para concebir, diseñar, implementar y operar sistemas digitales que utilicen tecnología de punta y metodologías de diseño modernas, proporcionando un programa académico que actualice los contenidos y la metodología de las asignaturas del área de electrónica digital.

Tomando como fuente de inspiración el movimiento de software libre y de código abierto (FOSS) Hoy en día la estructura auto-gobernada más exitosa, millones de personas alrededor del mundo trabajan de forma conjunta y distribuida en busca de un bien común: generación y distribución de herramientas software, sistema operativo y todo tipo de aplicaciones incluyendo el código fuente bajo una licencia que permite su distribución y modificación. En este trabajo, se definió un concepto similar pero aplicado al desarrollo hardware, es decir, que permite la generación, distribución, estudio y modificación de plataformas físicas (placas de circuito impreso, dispositivos funcionales): el *hardware copyleft*, su principal objetivo es servir como canal para la transferencia del conocimiento necesario para diseñar y producir sistemas digitales a todo sector de la sociedad que esté interesado. Con esto pretendemos generar en el país los conocimientos necesarios para aumentar la oferta de productos tecnológicos producidos localmente, lo que se traducirá en un aumento de la oferta local de bienes y servicios relacionados con la manufactura de sistemas digitales, generando empleo para personas con diferentes niveles de formación.

### **3. Metodología para la transferencia tecnológica en el área de diseño de sistemas embebidos.**

Para adquirir los conocimientos asociados a la tecnología utilizada en el diseño e implementación de sistemas digitales modernos, se aplicó una metodología para la transferencia tecnológica, que ha sido utilizada con éxito en otros países del mundo (Cohen, 2004,

Wood, 2005, Al-Mabrouk, y Soar, 2009) . A continuación se describen brevemente los pasos de esta metodología (para mayor información consultar Camargo, 2011c) :

**Elección:** Existen varias alternativas para la implementación de un sistema embebido: FPGA, sistema sobre silicio (SoC), SoC + FPGA y ASIC, la utilización de FPGAs o SoCs está determinada por el cumplimiento de restricciones temporales y funcionales, mientras que el uso de ASICs depende de el número de unidades producidas, se estima que a partir de 10 mil unidades se debe utilizar un ASIC para reducir los costos de producción. Debido a que los niveles de producción de los países en vía de desarrollo no son muy grandes, se abordará el problema de la transferencia utilizando FPGAs y SoCs comerciales, sin descuidar el estudio e implementación de ASICs. Adicionalmente, la inversión necesaria para construir un circuito integrado es muy alta, y puede ser considerada como un punto final.

**Adquisición:** En la actualidad es muy fácil adquirir productos implementados con tecnología de punta, existe una gran variedad de plataformas de desarrollo y de dispositivos comerciales a los que se les puede aplicar ingeniería inversa para entender y modificar su funcionamiento.

**Adopción** En esta etapa se utilizó la ingeniería inversa para: identificar las diferentes arquitecturas de los sistemas digitales adquiridos; identificar los mecanismos que permitan modificar su funcionamiento; identificar las herramientas que permiten crear nuevas funcionalidades; aprender la forma de utilizar los recursos tecnológicos existentes de forma adecuada; conocer la metodología de diseño. Se desarrollaron y/o adaptaron metodologías de

diseño que utilizan herramientas de libre distribución.

**Absorción** La absorción es una actividad de aprendizaje que integra conocimiento que es nuevo para el país pero que no es nuevo para el mundo; en esta etapa se pasó de las plataformas comerciales al diseño de aplicaciones propias, para lo que se desarrollaron y/o adaptaron técnicas de fabricación al entorno local, se realizó la transferencia de los conocimientos adquiridos a la academia y se iniciaron contactos con empresas manufactureras locales y extranjeras. Se liberaron 6 plataformas de desarrollo bajo la licencia CC-BY-SA Camargo, 2008; Camargo, 2007; Camargo 2006, los archivos necesarios para su reproducción, tutoriales sobre su funcionamiento, notas de aplicación y diseños de referencia se encuentran disponibles a todo interesado en el sitio web: <http://wiki.linuxencaja.net>

**Aplicación** En esta etapa se transfieren los conocimientos adquiridos en las fases anteriores a la industria creada (emQbit) como parte de la metodología. Adicionalmente, se creó, un programa académico para la enseñanza de sistemas digitales que crea las habilidades necesarias para concebir, diseñar, implementar y operar dispositivos digitales modernos y la definición del concepto *hardware copyleft* y su utilización como herramienta en la enseñanza de diseño de sistemas embebidos (Camargo, 2011b); este programa está siendo utilizado de forma oficial en el departamento de Ing. Eléctrica y electrónica de la Universidad Nacional de Colombia sede Bogotá.

**Difusión:** En esta etapa (se está realizando en el momento de escribir este artículo) se busca hacer llegar este conocimiento a todos los sectores de la sociedad interesados en él. Creando una comunidad que lo utilice como un recurso común; proporcionando a los centros de formación un programa académico actualizado que permita generar en los estudiantes las habilidades necesarias para innovar y generar empleo; y a la industria le suministra herramientas que puede utilizar para desarrollo de nuevos productos comerciales y para la capacitación de su recurso humano. El proceso de difusión se realiza en dos frentes, en el plano académico se realizaron conferencias y cursos de actualización donde se presentan las plataformas *hardware copyleft* desarrolladas; hasta el momento 4 de las principales universidades públicas del país están comenzando a implementar total o parcialmente el programa académico de las asignaturas del área de electrónica digital (Camargo 2011a, Camargo 2011b). Para ayudar en la difusión, se creó el portal público *linuxencaja* para almacenar todos los archivos, documentos, tutoriales, notas de aplicación y diseños de referencia de las plataformas diseñadas en este estudio; se dispone de una wiki que puede ser utilizada por cualquier persona para documentar nuevos proyectos y consultar los existentes; además se cuenta con una lista de correo que permite tener contacto con los desarrolladores y con otros interesados en el diseño hardware.

#### **4. Plan de estudios adaptado a la iniciativa CDIO.**

La iniciativa CDIO (<http://www.cdio.org>) ha sido desarrollada por el MIT con ayuda de académicos, industriales, ingenieros y estudiantes (CDIO, 2009) como respuesta a los diferentes caminos que están tomando la educación de la ingeniería y las demandas del mundo real. La iniciativa CDIO se basa en la suposición de que los egresados de los centros de formación en ingeniería deben ser capaces de: **Concebir, Diseñar, Implementar y Operar** sistemas funcionales en el mundo real. En Colombia, la mayoría de los centros de formación solo tienen en cuenta la concepción y el diseño, descuidando por completo la implementación y la operación; esto, impide que se generen las habilidades necesarias para establecer una estrecha relación con la industria, la cual, requiere productos que pueda comercializar o den soluciones a sus necesidades. La frase *en el mundo real* resalta la importancia de trabajar en la solución de problemas que pueden encontrarse en el ejercicio profesional, lo que es muy difícil de determinar cuando los docentes no han tenido contacto con la industria. La iniciativa CDIO se enfoca en preparar a los estudiantes con los conocimientos habilidades y aptitudes para ser ingenieros líder; y sus principales objetivos son (CDIO, 2009): educar a los

estudiantes para dominar un conocimiento más profundo de los fundamentos técnicos; educar a los ingenieros para liderar la creación y operación de nuevos productos y sistemas; educar futuros investigadores para entender la importancia estratégica y el valor de su trabajo.

Estos objetivos se adaptan a los requerimientos que se exige a la plataforma tecnológica de un país para que pueda realizar una adecuada absorción del conocimiento transferido para posteriormente transformar ese conocimiento en nuevos productos adaptados a las necesidades del país.

**Estructura del plan de estudios CDIO:** Todo individuo interesado en obtener habilidades técnicas posee *habilidades personales y profesionales*, las cuales son fundamentales para la práctica. Para ser capaces de desarrollar sistemas complejos en ingeniería, los estudiantes deben dominar los fundamentos del *razonamiento y conocimiento técnico*; para trabajar en un entorno moderno basado en grupos de trabajo, los estudiantes deben desarrollar *habilidades interpersonales* de comunicación y trabajo en equipo; finalmente, para ser capaz de crear y operar productos y sistemas, un estudiante debe entender el concepto de *concebir, diseñar, implementar y operar* sistemas en el Contexto social y empresarial (CDIO, 2009).

**Definición e Identificación de las Habilidades CDIO:** El primer paso en la implementación del plan de estudios CDIO es definir e identificar las habilidades requeridas en un área específica del plan de estudios. En el DICE de la Universidad Nacional de Colombia, el área de electrónica digital esta compuesta por tres asignaturas para la carrera de ingeniería electrónica: Electrónica Digital 1, Electrónica Digital 2 y Sistemas Embebidos. Para trasladar esta lista de habilidades a objetivos de aprendizaje es necesario determinar el grado de competencia que se espera que el profesional adquiera en cada una de las asignaturas; por supuesto, algunas de estas habilidades no pueden obtenerse solo en una asignatura y es necesario que todo el plan académico contribuya a generarla, lo que requiere un consenso del personal académico. Los niveles de competencia seleccionados para indicar el grado en que debe ser apropiada una determinada habilidad son: *Introducir*, (no evalúa); *enseñar* (enseña y evalúa) y *utilizar* :( puede ser evaluado o no).

**Competencias de las habilidades CDIO:** En la figura 1 se muestran las competencias de las habilidades para el plan de estudios CDIO aplicado a las asignaturas del área de electrónica digital. En ella podemos observar que existen habilidades comunes a las tres asignaturas en lo relacionado con el planteamiento y resolución de problemas, experimentación y descubrimiento de conocimiento y habilidades y actitudes personales; todas ellas buscan que el estudiante sea capaz de identificar un problema y con base en los conocimientos adquiridos formule hipótesis y modelos que permitan dar solución. Las habilidades interpersonales son tratadas de forma gradual, en el primer curso, se guía en la formación de estas habilidades utilizando ejemplos que ellos utilizarán en los cursos posteriores. Estas habilidades, a nuestro modo de ver ayudarán a la formación de personales capaces de absorber y asimilar los conocimientos necesarios para adaptarse a los cambios en la industria electrónica digital, al tiempo que los aplican en la solución de problemas locales.

Algo único en estas tres asignaturas es la implementación de sistemas físicos; por lo tanto, es muy importante que los estudiantes adquieran estas habilidades en estos tres cursos. El proceso de fabricación hardware se explica desde el primer curso, pero desde el segundo y tercer se realizan implementaciones por parte de los estudiantes. Las plataformas *hardware copyleft* son utilizadas como diseños de referencia y son utilizadas para copiar sus normas y reglas de diseño. En el primer semestre se enseñará como se pueden implementar tareas hardware utilizando lenguajes de descripción de hardware y dispositivos lógicos programables; en el segundo curso se utilizarán estos conocimientos para entender al

arquitectura de un procesador y para realizar periféricos, en este curso se estudiará el proceso de implementación de tareas software y la integración software - hardware; en el último curso se utilizarán estos conocimientos para desarrollar productos comercializables que utilizan dispositivos modernos.

Competencias de las habilidades CDIO nivel 2 y 3			
APTITUDES PERSONALES Y PROFESIONALES	Nivel 1		
	E. DigI	E. Dig2	Sist. Emb.
<i>Planteamiento y resolución de problemas de ingeniería</i>			E.U.
1 Identificación y formulación del problema			E.U.
2 Modelamiento			E.U.
3 Solución y recomendación			E.U.
<i>Experimentación y descubrimiento de conocimiento</i>			U
4 Formulación de hipótesis			U
5 Investigación experimental			U
<i>Planteamiento sistemático</i>			E.U.
6 Pensamiento global			U
7 Sangramiento e interacciones			U
<i>Habilidades y actitudes personales</i>			U
8 Pensamiento creativo			I.EU.
9 Pensamiento crítico			I.EU.
10 Toma de conciencia de conocimientos propios			I.EU.
11 Curiosidad y aprendizaje permanente <i>Habilidades y actitudes profesionales</i>			U
12 Ética profesional, integridad, responsabilidad			U
13 Comportamiento profesional			U
39 Confianza y lealtad			I.EU.
Nivel 1			
HABILIDADES INTERPERSONALES			
<i>Equipo de trabajo</i>	E. Digital1	E. Digital2	Sist. Emb.
			E.U.
14 Formar grupos efectivos	E.U.	U	U
15 Equipo de liderazgo	E.U.	U	U
40 Equipo Técnico y Multi-disciplinario	E.U.	U	U
<i>Comunicaciones estructuradas</i>			E.U.
16 Estrategia de comunicación	E.U.	U	U
17 Estructura de la comunicación	E.U.	U	U
18 Comunicación Escrita	E.U.	U	U
19 Comunicación Electrónica	E.U.	U	U
20 Presentación Oral	E.U.	U	U
<i>Comunicación en Idioma Extranjero</i>			U
21 Inglés			U
<i>Comunicaciones Informales: Relacionarse con los demás</i>			U
41 Preguntar, Escuchar y Dialogar	E.U.	U	U
42 Negociación, compromiso y resolución de conflictos	E.U.	U	U
43 Establecimiento de conexiones	I.EU.	U	U

HABILIDADES CDIO	Nivel 1		
	E. Digital1	E. Digital2	Sist. Emb.
<i>Contexto Externo, Social, Económico y Ambiental</i>			I.EU.
22 Rol y responsabilidad de los Ingenieros			I.EU.
23 Impacto sobre la sociedad y el medio ambiente			I.EU.
24 Cuestiones y valores actuales			I.EU.
44 Sostenibilidad y necesidad de un desarrollo sostenible	IE	IE	IE
<i>Empresa y contexto empresarial</i>			E.U.
25 Intereses en la empresa, metas y objetivos			I
26 Espíritu Empresarial Técnico			I
27 Trabajo exitoso en organizaciones			I
45 Finanzas y Economía de los Proyectos de Ingeniería	IE	IE	IE
<i>Concepción y Administración de Sistemas en Ingeniería.</i>			I.EU.
28 Entender las necesidades y establecer las metas	I.EU.	E.U.	U
29 Definir la función, concepto y arquitectura	I.EU.	E.U.	U
<i>Diseño</i>			I.EU.
30 Proceso de Diseño	I.EU.	E.U.	U
31 Fases del proceso de Diseño y enfoques	I.EU.	E.U.	U
32 Utilización de conocimiento científico en el diseño	I.EU.	E.U.	U
33 Diseño específico	I.EU.	E.U.	U
34 Diseño multi-disciplinario	I	E	U
<i>Implementación</i>			E.U.
35 Proceso de fabricación Hardware	I.EU.	E.U.	U
36 Proceso de Implementación de Software	I	E.U.	U
37 Integración Software - Hardware	I	E.U.	U
38 Pruebas, verificación, validación y certificación	IE	E.U.	U

## **Figura 1.** Competencias de las Habilidades CDIO.

**Metodología:** Los tres cursos tienen un carácter teórico-práctico y están articuladas alrededor de una única metodología de diseño; el componente teórico tratará los diferentes temas de forma general, con el fin de no crear dependencia con las herramientas utilizadas (lo que permitirá realizar actualizaciones de forma fácil). En el componente práctico, se tratarán temas específicos de manejo de las herramientas (lenguajes de descripción de hardware, lenguajes de programación, manejo de plataformas de desarrollo) y como estas se relacionan con la metodología de diseño utilizada. El estudiante debe estudiar, profundizar y comprobar algunos temas tratados en clase y debe leer previamente la documentación que se encuentra disponible en el sitio web de los cursos. Adicionalmente, debe formar grupos de trabajo para definir las especificaciones, diseñar e implementar un dispositivo que resuelva una determinada necesidad (con la complejidad adecuada para cada curso). En la sesión teórica se tratarán aspectos relacionados con la concepción, diseño, identificación y definición de las funciones de los componentes del sistema, mientras que en el laboratorio se tratarán temas relacionados con la implementación de dichos componentes sobre PLDs o SoC. Se deben realizar presentaciones del avance, indicando las razones que se tuvieron en cuenta en cada decisión y como se resolvieron los problemas encontrados, todo este proceso debe documentarse en la wiki del portal *linuxencaja*, esto último para formar un banco de proyectos que pueda ser utilizado como referencia por quien este interesado.

## **5. Conclusiones**

El problema de los canales tradicionales para la transferencia tecnológica es la poca cantidad de conocimiento transferido a la sociedad receptora; la metodología aquí presentada se centra en la adquisición de conocimiento y en su difusión a todo el que este interesado, lo cual es totalmente opuesto a iniciativas similares que terminan en cursos de actualización que limitan la difusión de conocimiento a quien pueda pagar por él.

La falta de cursos que enfrenten al estudiante con problemas reales es una consecuencia de la falta de conexión entre la empresa y la academia y el desconocimiento de los procesos industriales por parte de los docentes; un porcentaje considerable de la planta docente de las universidades mas prestigiosas del país siempre se han desempeñado en entornos académicos y no entienden las necesidades de la industria, lo que impide, por un lado la creación de habilidades requeridas por la industria y por otro lado dificulta la creación de empresa desde la universidad y la creación de vínculos con empresas constituidas.

Encuestas realizadas a los estudiantes durante los últimos dos años muestran que ellos perciben un grado de exigencia mucho mayor comparando con otras asignaturas; pero al mismo tiempo, que la experiencia en estos cursos es muy útil para su vida profesional y que es la única asignatura que los enfrenta a problemas reales de diseño e implementación de sistemas y trabajo en equipo; entienden que es necesario dedicar tiempo por fuera de aula si se desea asimilar la información, son conscientes de que la responsabilidad de adquirir este conocimiento es de ellos; manifiestan la importancia del uso de esta tecnología en la solución de problemas locales y entienden el estado de la industria digital en el país y su papel para dar solución a sus problemas.

Tanto los temas como la metodología del presente plan de estudio representan una forma novedosa de enseñanza enfocada a generar en los estudiantes habilidades necesarias para crear productos innovadores, proporcionando conocimientos actualizados y metodologías de diseño modernas basadas en herramientas abiertas que permiten conocer el flujo de diseño completo, trabajar en equipo y compartir los resultados con quien esté interesado. Se espera

que estas habilidades sean utilizadas por la industria para crear una oferta local de bienes y servicios relacionados con el diseño digital y que se genere el interés necesario en algunos estudiantes para crear empresas de diseño digital. De la experiencia obtenida al dictar cursos de capacitación en diferentes centros de formación a lo largo del país podemos afirmar que no se cuenta con los conocimientos necesarios para absorber y aplicar los conocimientos necesarios para el diseño e implementación de sistemas embebidos debido en parte al uso de metodologías de diseño obsoletas y al abandono de la implementación física por parte de los centros de formación; en las empresas, los profesionales no cuentan con la formación necesaria que les permita realizar un proceso de auto-aprendizaje y muestran deficiencias conceptuales en el diseño de sistemas digitales.

## 6. Referencias

- Al-Mabrouk, K y Soar, J. (2009) Identification of key issues for successful technology transfer in the Arab countries: a Delphi study . Journal: International Journal of Technology Transfer and Commercialisation
- Bar, F. Pisani, F., y Weber, M. (2007) Mobile technology appropriation in a distant mirror: baroque infiltration, creolization and cannibalism. *Seminario sobre Desarrollo Económico, Desarrollo Social y Comunicaciones Móviles en América Latina*. Buenos Aires.
- Camargo, C. (2006). *First Colombian Linux SBC runs Debian*. Recuperado el 13 de enero de 2011 de <http://www.linuxfordevices.com/c/a/News/First-Colombian-Linux-SBCruns-Debian/>.
- Camargo, C. (2007). ECBOT: Arquitectura Abierta para Robots Móviles. *IEEE Colombian Workshop on Circuits and Systems*.
- Camargo, C. (2008). ECBOT y ECB AT91 Plataformas Abiertas para el Diseño de Sistemas Embebidos y Co-Diseño HW/SW. VIII Jornadas de Computación Reconfigurable y Aplicaciones, Madrid España.
- Camargo, C. (2011a) SIE: Plataforma Hardware copyleft para la Enseñanza de Sistemas Digitales , *Memorias del XVII workshop de Iberchip*.
- Camargo, C. (2011b). Hardware copyleft como Herramienta para la Enseñanza de Sistemas Embebidos. *Simposio Argentino de Sistemas Embebidos*.
- Camargo, C. (2011c) Metodología Para la Transferencia Tecnológica en la Industria Electrónica Basada en Software Libre y Hardware Copyleft. VI Congreso Internacional de la Red de Investigación Y Docencia en Innovación Tecnológica RIDIT.
- Cohen, G. (2004). *Technology transfer: strategic management in developing countries*. Sage Publications inc.
- Hess, C. y Ostrom, E. (2006). *Understanding Knowledge as a Commons: From Theory to Practice*. The MIT Press.
- Odedra, M. (1994). The Myths and Illusions of Technology Transfer. *IFIP World Congress Proceedings*.

Wood, D. (2005) The Use of Satellite-Based Technology in Developing Countries

School: S.B., Aeronautics and Astronautics Massachusetts Institute of Technology

CDIO (2009). Benefits of CDIO consultado en Enero de 2009 en <http://www.cdio.org/benefits-cdio> on November, 2009

**Sobre los autores**

Carlos Iván Camargo Bareño: Ingeniero electricista, Máster en Ingeniería eléctrica de la Universidad de los Andes, Candidato a Doctor en Ingeniería Eléctrica de la Universidad Nacional de Colombia, Profesor asistente. [cicamargoba@unal.edu.co](mailto:cicamargoba@unal.edu.co)