Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Industriales

Departamento de Automática, Ingeniería Electrónica e Informática Industrial

Master on Industrial Electronics

FPGA-Based Wireless Sensor Node Architecture for High Performance Applications

Author: Juan Valverde Alcalá

Directors: Jorge Portilla Berrueco

Eduardo de la Torre Arnanz

April 2012

Master Thesis

Index

Chapter 1
INTRODUCTION

"Without craftsmanship, inspiration is a mere reed shaken in the wind"

Johannes Brahms

1. Introduction

During the last decade, Wireless Sensor Networks (WSNs) have been evolving towards more advanced applications using the latest technology breakthroughs. This evolution leads to an important increase of the resources required to cover the needs of these new scenarios. Even though the use of hardware-based solutions had been avoided by the majority of WSN designers, new approaches are being opened to this kind of devices due to their much higher powerful processing capabilities. In this Master Thesis, a novel hardware-based wireless sensor node architecture with specific energy saving methodologies is presented. This research work covers all the design steps beginning with the study of high performance WSN requirements, following with the design and development of a new node architecture and final prototype, and finishing with validation tests in order to illustrate the feasibility of the platform in terms of power consumption, size and performance.

I. Wireless Sensor Networks

Human endeavor for the understanding and control of the environment has led to the development of smart monitoring systems known as Wireless Sensor Networks (WSNs). These systems have been evolving during the last decade into more efficient and adaptable solutions to face a wide range of different scenarios [Buratti'09].

WSNs open a new way to understand the environment through the use of distributed intelligence. These networks consist of a group of electronic devices, called nodes or motes, deployed in different environments with the main goal of collecting information in a non-intrusive and unattended way. Depending on the tasks carried out by every one of these nodes, they must include different capabilities. Regarding their functionality, wireless sensor nodes are normally divided into three different groups:

- Sensor nodes: They must be capable of collecting information from the environment as well as processing this information to send it to other devices.
- Router nodes: They act as bridges in order to send the information through the wireless network via radio links.
- Sink nodes: They are in charge of collecting all the information from other nodes in order to share it with a computer or with another network.

These functions may change depending on each application. Besides, in some cases, a node can include some of these capabilities at the same time. For instance, a router node can work as a sensor node or, if the network is separated in different clusters, a sink node of one of these clusters can also work as a router.

Power consumption is one of the main concerns when dealing with this kind of systems since they must be both stand-alone and wireless. This fact implies a very important handicap in terms of power requirements. The use of low power processors together with power management strategies allows these systems to be powered by batteries that can last over a year depending on the applications and the platforms used. **These power management strategies combined with high performance capabilities are one of the main contributions of this Master Thesis**.

Even though the survey of standard WSN applications is out of the scope of this work, in order to understand their possibilities, some examples are listed. In the state of the art, it is possible to find quite a few different applications and scenarios where WSNs are being used. In [Corke'10], environmental and agricultural solutions are shown. Applications like cattle and water quality monitoring, virtual fencing, etc. take advantage of the use of these networks for continuous monitoring in a non-intrusive way. As well as in the previous case, the possibility of monitoring environmental parameters, but in food factories, is also detailed in [Valverde'11_1].

Another important contribution of WSNs is the capability of including intelligent energy management systems, [Nguyen'10]. In a world where energy efficiency has become one of the main concerns, the possibility of being aware of the power consumption in buildings, offices or factories and therefore being capable of acting

on them, opens a wide umbrella of power management opportunities. Those are some of the most popular applications related to WSNs. Apart from them, these networks are being used for health monitoring in humans [Yung-Cheng'09], structure monitoring for buildings and bridges [Xiao'11], fire detection in forests [Shixing'10], and myriad others.

As it happens with the applications, there are quite a few different WSN platforms in the state of the art. In Table 1, some of them are listed and compared (see: [Culler'05], [Nachman'05], [Smith'07], [Libelium'09], [Benbasat'05], [Limberopoulos'07], [Yamashita'06], [Portilla'06]). The table shows which processors and communication protocols are being used on every sensor platform.

Platform	Marketed	Nº of Layers	Processing	Communications
TelosB (Berkley)	YES	2	MSP430F1611 16 bit	CC2420 IEEE 802.15.4
Intel iMote	YES	2	XScale PXA271 32 bit	CC2420 IEEE 802.15.4
Sun SPOT	YES	2	ARM920T 32 bit	CC2420 IEEE 802.15.4
Wasp mote (Libelium)	YES	3	ATMega1281 8bit	XBee module, ZigBee compliant
MIT Platform	NO	4	C8051F206 8 bit	TDMA protocol
mPlatform (Microsoft)	NO	Not specified (>4)	2 processors in each layer and 1 CPLD XC2C512 CoolRunner	CC2420 IEEE 802.15.4

Table 1: WSN Platform Comparison.

In [Charoenpanyasak'11], a complete survey on WSNs nodes is done including both architectures for standard applications and a new generation of nodes to face more demanding tasks. In Figure 1, every one of these platforms is shown.

Figure 1: WSN Platforms. From left to right in the upper row: TelosB, Intel Mote, BTnode, mPlatform and on the lower row: Hitachi ZN1, MIT node, SunSpot, WaspMote.

In this Master Thesis, an adaptation of the *Cookie* WSN [Portilla'06] platform adding a high end FPGA to face very demanding computing requirements is presented. The main features of this architecture will be shown in chapter 2 in order to introduce the platform for future changes.

II. Introduction to Wireless Sensor Networks for High Performance Applications

WSNs are typically conceived as all-purpose designs used to cover a wide variety of applications. However, the requirements in some of these applications in terms of security, speed or energy efficiency are being increased while new demanding scenarios have turned up. In order to cope with this situation, new approaches must be taking into consideration.

Typically, WSN nodes have had some data processing capabilities, but since power consumption is always a crucial matter, the processors used are very limited. Accordingly, the availability of processing within the node also has a big limitation in terms of amount of data and speed of processing. These low profile processors offer enough processing capabilities to face standard applications while they keep ultra-low power consumption. The MSP430 microcontroller from Texas Instruments included in the TelosB platform [Culler'05] from the University of California at

Berkeley or the ATmega1281 used by Libelium in the Waspmote [Libelium'09], are some of these examples.

Apart from standard applications, new demands have appeared recently regarding two main aspects: first, more demanding scenarios are being introduced in the WSN field, and second, the requirements for traditional applications are increasing. In this context, traditional platforms reach a limitation so new solutions must be considered keeping in mind both low power consumption and the size constraints inherent to WSNs.

Normally, standard WSN applications share some common features such as low data rates, non-restrictive latency and low number of nodes (< 100 nodes) among others. However, only by increasing the area to be covered, the requirements can change sharply. For instance, in a deployment with a huge number of nodes, the amount of raw data can be too high to be processed by a low profile controller. As an example, 200 Kbits/s is a high data rate for a wireless sensor node. At the same time, synchronization and latency can also become a big problem so much more powerful processors are required.

Apart from the increasing demands in traditional applications, new ones have shown up related to surveillance, communication security, tracking algorithms, etc. All of these applications require much more powerful processing units to deal with huge amounts of data and complex calculations. Those high performance applications are normally related to the use of video cameras, data encryption and compression, audio applications, large deployments, tracking algorithms, latency restrictions and synchronization, big memory usage, etc. That is the reason why new WSN architectures must be studied to be able to face these new complex operations.

III. Motivation and Main Goals

The motivation to carry out this research work is the study of high performance WSNs in order to demonstrate the convenience of using hardware-based architectures instead of the traditional microcontroller-based ones. This Master Thesis shows the design and validation of a relatively high capacity state-of-the-art

FPGA-based platform together with the implementation of energy saving methodologies.

The main goals of this Master Thesis are listed below:

- Study and understand the needs of the new high performance applications for WSNs.
- Summarize all the requirements imposed by this kind of applications and translate them into real specifications.
- Adapt the *Cookie* WSN platform to comply with these specifications developing a new processing and power supply units.
- Build a final prototype to be used by the different partners of two European projects.
- Validate the design with real tests under real scenarios.
- Guarantee enough features to:
    - Carry out complex algorithms in a fast way keeping energy efficiency.
    - Have enough available memory to collect and work with application data.
    - Be capable of carrying out dynamic and partial reconfiguration.
    - Implement low power strategies to reduce power consumption as much as possible.

IV. Document Structure

This Master Thesis is organized as follows: chapter 2 is divided into three different subsections. The first one gives an overview about the Cookies Wireless Sensor Network platform. The second section describes the main requirements imposed by high performance applications, while the last one is focused on hardware reconfiguration.

Chapter 3 includes all the system requirements and specifications, details about the platform architecture and some interesting data about the real PCBs design.

In chapter 4, different methodologies to achieve the maximum energy efficiency are explained together with the different power save modes included in the design.

Chapter 5 explains two possible application scenarios where the performance of the platform proposed is shown. In addition to this, this chapter includes all the different steps needed to validate the hardware design as well as several tests to demonstrate the feasibility of the solution in terms of power of processing and energy efficiency compared to other solutions. To conclude, in chapter 6, some future lines and conclusions are discussed.

Chapter 2

GENERAL CONCEPTS AND STATE OF KNOWLEDGE

"Cuando creíamos que teníamos todas las respuestas, de pronto, cambiaron todas las preguntas"

Mario Benedetti

2. General Concepts and State of Knowledge

General concepts about WSNs were already discussed in the Introduction chapter. This second section is focused on describing the platform developed at the Technical University of Madrid (Universidad Politécnica de Madrid) within the Center of Industrial Electronics (CEI) as well as on going a bit deeper into the needs of high performance applications for WSNs.

I. Cookie Platform

The first step for the correct understanding of this Master Thesis, once WSNs have been introduced, is the explanation of the basic features of the WSN platform that is going to be adapted. This is the *Cookie* WSN platform [Portilla'06].

As a general matter, WSN nodes include at least four functionalities: processing, communication, sensing and/or acting and power supply. In order to have a flexible design, the *Cookie* platform is divided into four different PCB layers, each one of them covering one of these previous roles.

Every layer is connected to the following through vertical connectors. These connectors have two main goals: acting as a joint point between layers and connecting all the signals from one layer to the other. In this way, it is possible to exchange every layer separately if different sensors, communication modules, power supply sources, etc. are needed. This modularity is the main feature of this platform and it is very useful when adapting it to comply with different requirements and scenarios. The four layers mentioned are listed below:

- Sensing layer: it includes conditioning circuits for both digital and analog sensors. The output signals of these conditioning circuits go through the vertical connectors to the processing layer. All the different models are detailed in the next page.

- Power supply layer: it is the power source of the node. The node can be powered from an USB cable, lithium or AA batteries or directly from the mains, if it is necessary.
- Communication layer: it includes the radio module to communicate data between nodes. It can be either a ZigBee or a Bluetooth module. In the case of the ZigBee module, different frequencies are available (2.4 GHZ and 868 MHz).
- Processing layer: it is the brain of the platform. It is the layer in charge of processing all the information given by the sensors and the radio module. It includes an FPGA and a microcontroller that changes depending on each version. This Master Thesis is focused on the design of a new processing layer.

This architecture divided in layers can be seen in Figure 2.

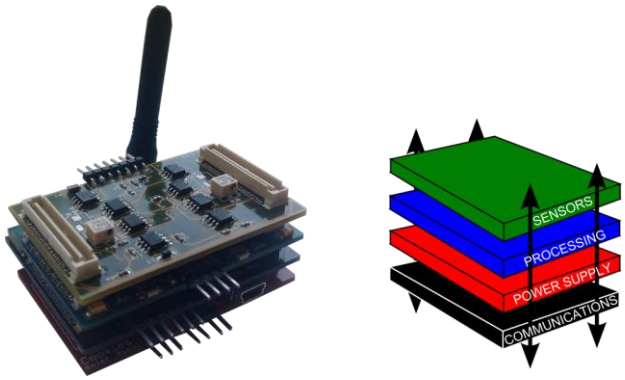


Figure 2: Cookies Architecture.

Even though every layer can be placed in any position, it is better for the communication one to be located in the external part of the node in order to avoid the noise from the rest of the layers. At the same time, it is important to have the sensing layer on the opposite side, since sensors and/or actuators usually need to be in contact with the environment. It is important to highlight that the position of the layers on the new adapted design will not be arbitrary but this will be explained in the following sections.

Depending on the tasks the node needs to carry out, it will include all the layers or only some of them. For instance, a node working only as a router will include the power supply layer and the communication module since only very simple processing tasks are required.

Thanks to its modularity, several versions of each layer have been already designed.

- Sensing layer:
  - Water temperature and pH.
  - Gas concentration: O_2, CO, NO_2, SO_2, CO_2.
  - Air temperature, accelometer and light.
  - Strain gauge and air temperature.
  - Video camera.
- Communication layer:
  - ZigBee 2.4 GHz Telegesis module.
  - ZigBee 900 and 868 MHz Atmel module.
  - Bluetooth.
- Power supply layer:
  - USB, Lithium battery, AA batteries.
  - USB, Lithium battery, TRACOPOWER. This version is capable of giving more current and it can be directly powered from de mains.
- Processing layer:
  - Spartan 3 FPGA and ADuC841.
  - Actel Igloo and MSP430 μC.

Since this Master Thesis main contribution is the design of a new processing layer, more details about the previous ones is needed. The first version of the processing layer included a low-cost and low-performance Spartan 3 FPGA together with a microprocessor (ADuC841). In this version, the FPGA was in charge of interfacing with different sensors and carrying out preprocessing tasks. In turn, the microprocessor executed the main node management functions. Despite of including an FPGA, the processing capabilities of the node were limited. In this Master Thesis its capabilities are extended while implementing more efficient energy management policies. Some reconfigurable capabilities were already included in this previous version of the processing layer, however, in the case of the Spartan 3, since the

processing board was not designed for reconfigurable purposes, reconfiguration needed to be done using the external microcontroller as a virtual JTAG port [Krasteva'11]. This feature is also enhanced in the new processing layer presented in this Master Thesis.

The *Cookie* platform has been already used in real scenarios such as tunnels, food factories, railways, food transport, mines or structure monitoring as part of other research projects. In all of those cases, the processing needs were mostly related to do some basic calculations and communication management in deployments with low number of nodes. The new design proposed in this work is not oriented to this kind of applications but for very demanding scenarios such as multimedia applications.

II. High Performance Wireless Sensor Networks

The evolution of WSNs towards more complex applications was already introduced in the first chapter. Standard applications are being improved in terms of latency, security or deployment size, while new application scenarios have turned up. The complexity imposed by these new requirements shows the limitation of the previous WSN platforms in terms of processing capabilities. In this context is where high performance platforms emerge as an enabling technology for WSNs.

These high performance applications are usually related to the use of complex algorithms such as video compression, data encryption, tracking algorithms, etc. The majority of them are multimedia applications that include the use of low-power video cameras and microphones. These applications are known in the state of the art as Wireless Multimedia Sensor Networks (WMSN). In [Harjito'11] and [Akyildiz'11], surveys about these applications are detailed. According to [Akyildiz'11], these applications can be classified in: surveillance, traffic monitoring and enforcement, personal and health care, gaming and environmental and industrial. Surveillance is probably the best example of the increase of complexity in traditional WSN applications. Traditionally, surveillance based on WSNs was limited to intruder detection or movement in target areas. Nevertheless, adding enhanced capabilities, like the use of cameras or node synchronization, permits the inclusion of advanced features such as location tracking or people identification, as seen in [He'06] [Tseng'06], [Wu'11] and [Kulkarni'06]. The use of these capabilities can also make a

breakthrough in terms of human health-monitoring applications. Applications related to telemedicine and complete patient monitoring can be achieved. For instance, in [Dilmaghani'11] a complete ECC monitoring scheme is shown. Regarding the same topic, the non-intrusive study of people behavior, mainly elder people suffering dementia, has been reported in the state of the art in [Avvenuti'09] and [Marzencki'09].

Applications related to environmental care and industrial monitoring can be also faced and improved by means of using these high performance networks. For instance, full manufacturing processes including quality control can be monitored relying on artificial vision techniques.

A good example of a new application field is Gaming, where 3D environmental virtualization gives the opportunity of introducing future approaches in videogames as seen in [Nedelcu'10] and [Chen'08].

Not only the inclusion of new sensors but also the toughening of traditional constraints such as maximum latency, bandwidth, the increase in the number of nodes or security requirements, causes architectural changes in WSN platforms. In [Grieco'09] and [Yong'06], a survey about this kind of applications is studied including encryption algorithms ([Stelte'06] [Portilla'10_1]) which at the beginning where considered unfeasible to be carried out by WSNs. As an example of complex data calculations, in [Cantoni'06], the authors face data-mining for WSNs while in [Rup'09] distributed multimedia source coding is addressed.

Traditional architectures have been based on ultra-low power microcontrollers with enough but limited computing power. However, the increase of complexity requires higher resources to cope with these new applications. This extra computing power can be achieved by means of either using more powerful microcontrollers or any other solution capable of accelerating task execution. Normally the increase of the node capabilities implies an increment of the energy consumption so that new energy saving policies must be taken into consideration.

The first solution could be the inclusion of more powerful microcontrollers, [Hammel'07]. This solution can be suitable for some applications, since they offer a good trade-off between price, size and programming flexibility. However, when

facing very intensive tasks, the computing time can be very high leading to a non-efficient solution in terms of energy consumption. A better solution can be achieved using Digital Signal Processors (DSPs), however, even though the computing time can be reduced compared to standard microcontrollers, it is still high so it is difficult to keep power consumption in acceptable levels for WSN standards. In Figure 3, a comparison between some microcontrollers in terms of power processing and power consumption is shown.

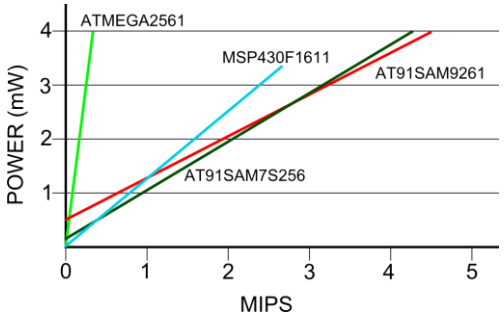


Figure 3: Power Consumption and Processing Capabilities.

On the other hand, better solutions can be achieved taking advantage of the specific features of WSN applications. The standard working profile of a WSN has a very low duty cycle (1% - 10%). Every node is awake only during the time concerning for carrying out measurements and doing some operations ([Dutta'05] [Pinto'10]). Then, the node is taken to a power-down or sleep mode where power consumption should be close to zero. Due to this reason, very quick processing engines are very suitable for these working profiles leading to very efficient solutions in terms of energy consumption by reducing processing time.

The best way to provide high processing speed is the use of hardware based systems. FPGAs and ASICs share the capability of doing tasks in parallel. This fact can decrease the computing time sharply so the node can remain in sleep mode for longer periods of time. However, the use of ASICs in WSNs is not always suitable due to the lack of flexibility and the huge design time. Due to these reasons, FPGAs are presented as a very suitable solution for high performance WSN applications.

In order to illustrate the convenience of using high-speed processing engines as a suitable method to save energy while being capable of performing very complex

tasks, a theoretical comparison of the working profile between an FPGA and a microcontroller in a WSN application is shown in Figure 4.

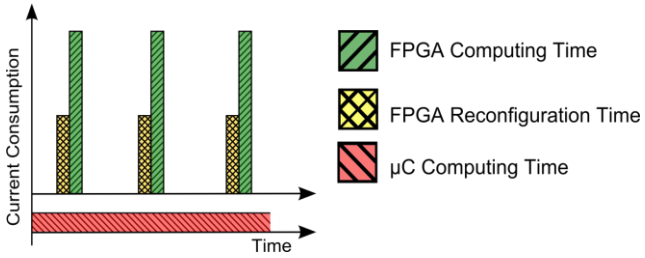


Figure 4: Consumption Profile Comparison.

While the time needed by the FPGA to carry out a certain complex task is usually very short, the time needed by a pure software-based solution is much longer. In this way, even though current consumption is much higher in the case of the FPGA, the energy is smaller due to the very short computing time. The main drawback is the fact that the FPGA configuration time is significant compared to the computing time. One of the main contributions of this work is proving that both configuration and processing times are still lower than the time the microcontroller requires to carry out the same task so very powerful energy saving methods can be implemented. It is important to highlight that this assertion is only true when dealing with complex algorithms where the computing time needed for the microcontroller is bigger than the configuration time the FPGA needs. Furthermore, the use of some FPGAs allows Partial and Dynamic Reconfiguration (DPR) which has many advantages when working with wireless sensor nodes as it will be discussed on the following section. In order to illustrate how fast hardware-based systems can be compared to software ones, in [Braun'09] a car-to-car communication system is proposed. In this kind of deployments, the reliability of communication is crucial, so both high security algorithms and low latency are required. In this work, the authors show an encryption algorithm based on elliptic curves running both in an ARM processor and a hardware-based system. Results show that for a short signature made of 160 bits, the encryption can take more than 90 ms in a 32-bit ARM processor or even more than 600 ms in a 16-bits M16C while the hardware-based solution, using 75 kgates, only takes 0.5 ms per signature verification.

It is important to highlight that it is not the aim of this work to be compared with ultra-low power WSN platforms since the application scope is completely different. This way, a trade-off between low energy consumption and high computing performance must be found.

Once the convenience of using FPGAs for high performance WSN applications has been introduced, a study of similar solutions in the state of the art is shown.

Using FPGAs in wireless sensor nodes to deal with high demanding scenarios is not a novel approach on the state of the art. In [Ji-gang'09], for instance, the authors introduce a Spartan 3E prototype board as a coprocessor attached to an external ZigBee transceiver for the implementation of a hyper-chaos encryption engine. Similar applications using off-the-shelf FPGA boards are shown in [Chalivendra'08] [Muralidhar'08] [Yan'11] and [Chao Hu'09], this last one oriented to visual sensors. These previous approaches prove that including hardware-based devices in WSNs offer certain benefits in terms of flexibility and performance. However, in spite of being valid for proof of concept, existing solutions are far from showing real WSN new architectures, leaving behind important aspects such as power consumption, power management, sensor integration, or even the inclusion of the FPGA itself on the node architecture. On the contrary, the development of a complete FPGA based node is provided in [Bellis'05], including a low-performance Spartan 2E. The complete node including the communication circuitry is integrated in a 25 mm × 25 mm board.

III. Hardware Reconfiguration: General Concepts

Reconfigurable hardware is presented as a feasible solution to reduce power consumption. In order to introduce the concept of reconfigurable hardware; first of all, it is sensible to make a brief review of those components that allow the implementation of this feature.

In 1984, the co-founders of Xilinx, Ross Freeman and Bernard Vonderschmitt, invented the Field Programmable Gate Array (FPGA) as an evolution of the Complex Programmable Logic Devices (CPLDs). An FPGA is an integrated circuit capable of the implementation of any logical function as a combination of several logic cells.

The most important elements in an FPGA are the Configurable Logic Blocks (CLBs), which are made of Look up Tables (LUTs) and interconnection matrixes to connect the inputs and outputs of each CLB. Apart from that, there are several Input/output elements to connect the internal logic with the external parts of the FPGA. This internal architecture can be seen in Figure 5.

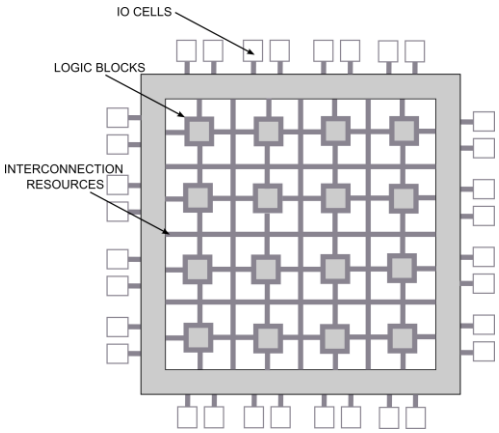


Figure 5: FPGA Architecture.

These features make FPGAs to be the most efficient way for digital circuit design in terms of implementation time and price when talking about low volumes. Therefore, they have become the best alternative to Application Specific Integrated Circuits (ASICs), even though ASICs usually have lower power consumption and sometimes can implement more complex custom-made designs. In this way, due to its flexibility and low price, FPGAs are used to test prototypes for ASIC design. It is important to take into account that the new generation of FPGAs are becoming more complex and are now the most suitable elements for digital signal processing. Besides, they can have embedded microprocessors to increase, even more, flexibility, making easy-to-use designs.

In order to understand how these devices work, it is crucial to talk about the configuration memory. The configuration memory can be seen as a parallel layer to the digital circuitry of the FPGA. This layer is the one in charge of defining the logic configuration of the circuitry, so it is where the designer must define what the circuit must do.

Once the architecture and behavior of FPGAs have been briefly explained, it is possible to talk about what hardware reconfiguration is. Hardware reconfiguration is the possibility of changing the functionality of an FPGA to reach the requirements of almost any hardware application. This reconfiguration can be done in a few different ways.

As it was explained before, the reconfiguration of an FPGA means writing in the configuration memory which is in charge of changing the functionality of the logic circuit. There are three main methods to carry out this task:

1) Full and Static Reconfiguration

It consists of changing the configuration of the whole area of the FPGA when the device is not working. This change is usually done by an external connection, normally the JTAG port (Join Test Action Group).

2) Partial and Static Reconfiguration

In this case, only a specific region of the FPGA is configured while the device is not working. In this way the configuration time can be sharply decreased as well as the power consumption. One of the main drawbacks in this method is the necessity of having advanced knowledge about the FPGA internal architecture.

3) Partial and Dynamic Reconfiguration

This is the most complex and the most interesting method. This option consists of changing the functionality of only certain region of the FPGA while the rest of the device is working. Normally, the internal area of the FPGA is divided into two different regions: static and reconfigurable area. The second one is in charge of allocating the changeable blocks while the first region is used for the embedded processor and peripheral controllers.

There are two main advantages when working with this reconfiguration method. The first one is the adaptation of the functionality of the FPGA in real time, and the second, is the possibility of multiplexing different tasks in time. In this way, the functionality can be changed at run time depending on different external parameters. This is translated in a huge increase of the efficiency of the system. Besides, task multiplexing allows the change of modules that have finished their tasks, replacing them for new ones with different functionalities.

On the contrary, the need of having a huge knowledge about the internal architecture of the device and the large design time, are the main drawbacks of this reconfiguration method. Besides, another aspect to take into consideration is the difficulty of porting the designs from one FPGA family to another, since everything depends on the internal architecture of the device.

Another important advantage offered by this reconfiguration method is Self-Reconfiguration. This is the possibility of changing the functionality without using any other device but the FPGA and, of course, some external memories to keep the configuration files. In this way, the system can be recovered if any problem occurs without any external help.

It is important to take into account that not all the FPGAs can be partially reconfigured. The different types of FPGAs that can be found in the market are listed below:

- RAM-based FPGAs: if the system is designed to be stand-alone, they need to have a non-volatile component to store the configuration file (bitstream hereafter), since once the FPGA is turned off, the information in the configuration memory is erased. They are quite power consuming especially considering the start-up time. Nevertheless, they are usually bigger, more evolved and they allow partial and dynamic reconfiguration.
- FLASH-based FPGAs: the power consumption in this case is lower taking into account that they are lived-at-powered-up, so no configuration is needed when the device is powered. The main drawback of this kind of FPGAs is that they cannot be reconfigured either partially or dynamically.
- Hybrid technologies: they are a mixture between the previous ones.

Now that the main features of both FPGA types and reconfiguration have been explained, the opportunities of partial and dynamic reconfiguration to face high demanding WSN applications can be studied.

IV. Opportunities of Reconfigurable Hardware in WSNs

The capability of carrying out hardware reconfiguration, partially and at run time, opens a wide variety of new opportunities for WSNs. The possibility of loading different hardware modules depending on the changeable needs of the network is a perfect feature for the optimization of task scheduling and power saving methodologies. Carrying out tasks in parallel and having the possibility of multiplexing them allow very fast processing engines that are very powerful for certain applications such as multimedia WSNs. Moreover, partial and dynamic reconfiguration allows the implementation of energy saving policies such as fast reconfiguration to decrease the time the FPGA is working, start-up sequences [Hübner'10] to make the initial configuration using only the modules that are strictly needed, etc. It is important to notice that, in WSN scenarios, this adaptation can be done both before and after the deployment. In this way the network can be easily adapted to new standards, new environmental conditions, or to the implementation of fault tolerance algorithms.

Partial configuration implies having smaller configuration files. In this way, it is easier and less power consuming for the system to send bitstream files through the wireless network via radio, while at the same time, loading these partial files is less consuming in terms of energy and storage size.

FPGAs have the possibility of being self-configurable; this means that they may include mechanisms to read the bitstream files from external memories automatically once they are powered. This feature allows the nodes to be autonomous which is crucial when talking about WSN deployments where most of the times the devices are switched off.

Hardware reconfiguration within wireless sensor nodes has been already faced in several applications on the state of the art. In [Glesner'11], a node that includes a Flash-based FPGA is used. Even though partial and dynamic reconfiguration is not possible with these devices, the authors overcome the reconfiguration problems by creating a virtual layer on the top of the logic one to enable different hardware blocks depending on the needs of the system. Even though the hardware blocks are always

implemented, they are not working unless the virtual layer enables them. In this case, dynamic reconfiguration is emulated.

A few applications including real reconfiguration have been already proposed. In [Nahapetian'07], the MicrelEye platform is presented. This platform uses an FPSLIC, a configurable device that includes a small FPGA (40 kgates) plus an ATMEL microcontroller. This device shares hardware flexibility and speed with the advantages of a microcontroller, when at the same time power consumption is smaller than in standard FPGAs. The main drawback of this kind of systems is that due to a smaller size than standard FPGAs, it is necessary to do a very precise task scheduling. In [Garcia'09], a Virtex 4 FPGA is used to implement a dynamically reconfigurable sensor node. In this work a Kalman filter is used to remove noise during data acquisition in a moving object tracking application. However, details about the node architecture or dynamic and partial reconfiguration impact on power management or start-up time are not included in the work. In the same way, the original *Cookie* platform was featured with dynamic and partial reconfiguration, using an external processor, as shown in [Krasteva'11].

In [Leligou'10], some applications where having reconfiguration properties can be very useful, are shown. For instance, urban operations can be very challenging, covering from applications where the movement inside and between buildings need to be checked, to military applications with rapid dissemination of combatants or shooter localization. The use of cameras in museums or supermarkets can permit face recognition and intruder tracking. Besides, by reconfiguring the platform according to the needs of video quality, power consumption can be reduced. Acoustic tracking of mobile targets in real time military applications is also an extremely challenging application where reconfigurable capabilities are very useful. Requirements such as high robustness, real time decision, high frequency sampling, multi-modality of sensing, complex signal processing and data fusion, distributed coordination in a wide area coverage, etc. can take advantage of hardware reconfiguration. In order to cover all of these requirements different hardware modules can be implemented and changed in real time. These modules can be:

- Data communication modules: changing communication parameters, reconfiguration according to the RSSI value, decreasing or increasing the data transfer rate according to the battery level.
- Data encryption and authentication modules.
- Data and video compression modules.

There is another important contribution to reconfiguration in WSNs related to data encryption with the *Cookie* platform. In [Portilla'10_2], a comparison between systems based on microcontrollers and FPGAs with embedded microcontrollers for encryption applications is done. In this case, since the Spartan 3 included in the *Cookie* platform is too small, the concept proof was done using the Spartan 3 plus a Virtex 2 plus the ADuC842 microcontroller of the *Cookies*. The idea was the adaptation of the sensor network security depending on different applications and available energy using the Elliptic curve cryptography (ECC). Apart from that, in [Otero'11] the authors have already provided a reconfiguration engine for Spartan 6, which exploits reconfiguration capabilities using the Internal Configuration Access Port (ICAP) of the FPGA. The main feature of this block is its relocation capability, which means, the possibility of changing at run time the position of the reconfigurable blocks within the device. Thanks to this block, it is possible to allocate each reconfiguration engine in any empty slot in the reconfigurable region as it will be explain in the next chapter.

Chapter 3

High Performance Node Architecture

"…just give me what I came for, then I'm out the door again…"

Perfect Circle

3. High-Performance Node Architecture

I. Design Requirements

The requirement list must answer the following questions: what needs the system meets and what features it must include. This requirement list has been evaluated following the needs of the high performance WSNs mentioned in the Introduction chapter. It is important to notice that these are the requirements of the system. The specifications and architecture will be explained within the following section in order to explain how these requirements are solved.

The main purpose of the system is acting as a wireless sensor node compliant with the *Cookie* platform to be used in very demanding applications. The main requirements for the platform design are listed below.

1. The node must be self-configurable so it can return to a known state once the system is powered on and off.
2. The node must be capable of including different functionalities and changing them depending on different internal or external variables, which means higher integration. All the changes must be done without stopping the rest of the system.
3. The configuration of the nodes must include the possibility of being done remotely and after deployment.
4. The platform must be capable of working with both digital and analog signals (i.e. from different sensors).
5. The system must be compliant with both the sensor and communication layers of the Cookie platform.
6. The design must be fully testable from a computer. It also has to have available check points for the initial development and validation.

7. The design must be compatible with future hardware upgrades.
8. The system must be aware of its own power consumption since it needs to implement energy saving policies.
9. The platform must be powerful enough to deal with multimedia applications, encryption algorithms, etc.
10. The physical size of the platform must be fully compatible with the Cookies architecture (60 mm x 40 mm) including the space for the vertical connectors.
11. The architecture of the node must include a mechanism to power different parts of the platform separately.
12. The design must be oriented to low power in order to be used with Lithium batteries.
13. The design must include hardware and software advantages to provide larger flexibility.

II. System Specifications and Architecture

Every one of the previous requirements must imply a specification for the node architecture. In the same way, every one of these specifications must be reflected in the node architecture. In order to make the structure of the document easier to understand, the architecture of the node is explained together with the specifications. Besides, to clarify, references to the previous requirements are done in order to explain how they are faced and solved.

This section is organized as follows: first, a top view of the system is explained in order to understand the communication between the existent layers and the new ones. Then, the new modules are explained in detail focusing on the new processing layer architecture and given some information about the new power supply layer.

II.1 Node Architecture

In Figure 6, the top view of the node architecture is shown. The node is divided into four different layers following the traditional design of the *Cookie* platform

(requirements 5, and 10). Both the communication and the sensor/actuator layers remain unchanged while new designs must be done for both the processing and power supply ones. The power supply layer must be redesigned in order to cover the new voltage supply values needed by the new processing layer. Besides, the power supply layer must be capable of giving much more current than the previous versions due to the current peaks required by the FPGA. Regarding the processing layer, this is the main design this Master Thesis is focused on, and it will be detailed in the following section.

As it was mentioned before, these four layers are connected through vertical buses. However, in the case of the power rails, the design has been changed in the work developed in this Master Thesis. In order to be able to power on and off the communication and sensor layers separately, the power layer must be placed under the processing one so that it is the processing layer the one in charge of controlling the voltage supply. Every module that needs to be powered permanently can be connected directly to the power supply layer in order to avoid the power rail control. Notice that in Figure 6, the communication layer is always powered while the sensor one can be powered on and off by the power management module included in the processing layer. Regarding the data transmission between layers, the signals are shared by all the layers like in the original design.

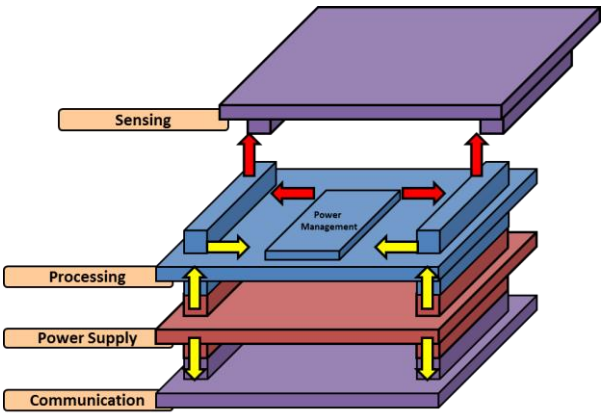


Figure 6: Power rails through the vertical buses.

II.2 Power Supply layer

In this new version of the power supply layer, the node can be powered either from a battery (Lithium or AA) or from a mini-USB port. Due to maximum current peaks, it is necessary for the power source to be able to provide up to 2 A in the worst case. In the case of a computer, the USB connection provides a maximum current of 500 mA, so a USB charger needs to be used. On the other hand, the battery used is a custom made one capable of handling these current values.

The USB connection has three main goals: powering the node, charging the battery and acting as an UART connection for debugging. The battery level and the current consumption of the whole node can be checked in order to send their values to the processing layer so that energy saving methodologies can be implemented (Requirement 8).

When both the battery and the USB cable are connected, the system is powered by the USB, while at the same time the battery can be charged. Both the battery and the USB outputs are connected to an integrated DC-to-DC converter with five outputs that provides the needed voltage and current values for the rest of the node. This DC-to-DC converter is an integrated circuit (IC) from Texas instruments that includes two power modes depending on the load connected to it. In this way, during the sleep periods, the load connected to the converter is low, so the power mode changes to power down mode with a power consumption of less than 1 mA. The diagram block of the power supply layer is shown in Figure 7. No further information about the development of this layer is detailed since this Master Thesis is focused on the processing layer design.

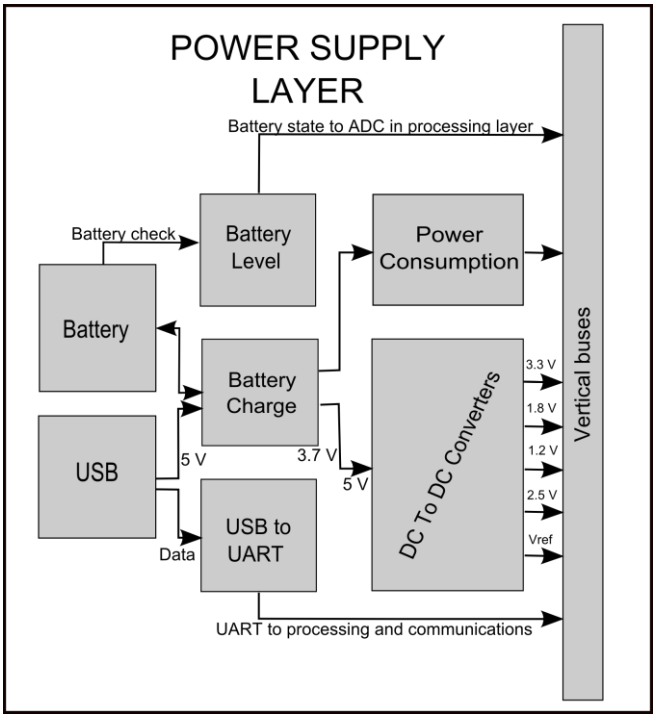


Figure 7: Power Supply Layer Diagram Block.

II.3 Processing Layer: HiReCookie

This section is divided into two different subsections. First, the physical architecture of the High Performance Reconfigurable Cookie (HiReCookie) including all main components and their interconnections is detailed. Second, the internal architecture of the FPGA, called Virtual Architecture (VA), which includes both the peripheral controllers and reconfigurable areas, is explained.

When designing this kind of systems, the design of the VA is usually the first step so that the placement of all the external components can be done according to the internal architecture. However, in order to make the information clearer, the external architecture is going to be explained first.

II.3.1 Physical Architecture

In Figure 8, the block diagram with the main components of the processing layer is shown. The figure is divided into two different areas separated by the vertical bus. All components included on the right side belong to the processing layer while the blocks placed on the left side correspond with the rest of the layers of the *Cookies* architecture. The architecture of the processing layer is divided into seven different power islands that can be powered on and off separately (Requirement 12). Every one of these islands together with the power management methodologies will be explained in the next section and are now represented by colored stars.

The FPGA is the brain of the platform. It will be in charge of all the processing tasks and management decisions. The external microcontroller is the only device that remains always powered. It works as a sentry to wake the rest of the system up according to the orders given by the FPGA. Since the FPGA is a RAM based device, once it is switched off, all the information is lost. In order for the system to be self-configurable, an initial bitstream is stored in the Flash memory so that it is automatically loaded into the FPGA once both devices are powered. The Flash memory also works as a storage device for program data and other bitstreams.

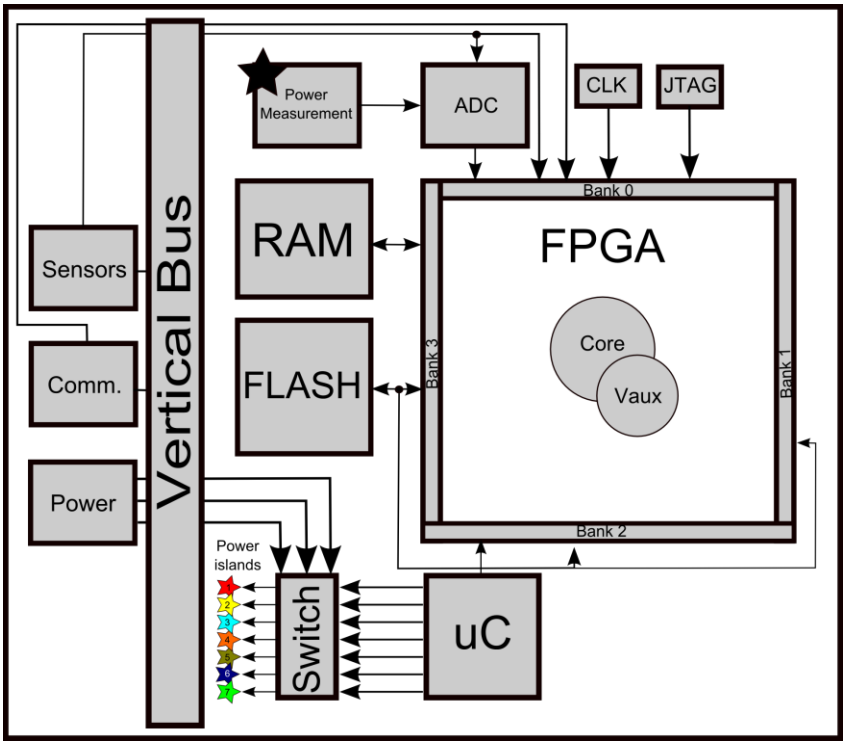


Figure 8: HiReCookie physical architecture.

The RAM memory is mainly used as an extension program memory to provide fast access to the program data. This is very useful when using fast partial and dynamic reconfiguration.

Not all the signals handled by the node are digital. In order for the system to be able to process data from analog sensors and measuring the instant power consumption in every island, an ADC converter is required.

All the previous blocks are detailed below.

FPGA

The FPGA that better matches the requirements of the system is the Spartan 6 XC6SLX150-2 from Xilinx. This is a RAM-based FPGA with a FGG484 package, large enough to allocate all the hardware blocks and peripherals needed by the high performance applications. Actually, it is the largest Spartan 6. The Spartan 6 family

has very competitive power consumption. This feature together with the power management strategies allows the system to be very suitable for WSN applications. Besides, a low-power version of the Spartan 6 with a speed grade of (-1) is already available in the market. According to the datasheet given by the manufacturer, this version can save up to 40% of energy by reducing the speed only in a 20%. The package of this low power version is compatible with the one used in the prototype presented in this work so it can be implemented in future work (Requirement 7).

This FPGA family offers a dual-register 6-input look-up table (LUT) logic and a wide selection of built-in system-level blocks including 18 Kb block RAMs, second generation DSP48A1 slices, SDRAM memory controllers, enhanced mixed-mode clock management blocks, etc. (See the component datasheet for more information). The main features of the chosen device can be seen in Figure 9.

Device	Logic Cells[1]	Configurable Logic Blocks (CLBs)			DSP48A1 Slices[3]	Block RAM Blocks		CMTs[5]	Memory Controller Blocks (Max)[6]	Endpoint Blocks for PCI Express	Maximum GTP Transceivers	Total I/O Banks	Max User I/O
		Slices[2]	Flip-Flops	Max Distributed RAM (Kb)		18 Kb[4]	Max (Kb)						
XC6SLX4	3,840	600	4,800	75	8	12	216	2	0	0	0	4	132
XC6SLX9	9,152	1,430	11,440	90	16	32	576	2	2	0	0	4	200
XC6SLX16	14,579	2,278	18,224	136	32	32	576	2	2	0	0	4	232
XC6SLX25	24,051	3,758	30,064	229	38	52	936	2	2	0	0	4	266
XC6SLX45	43,661	6,822	54,576	401	58	116	2,088	4	2	0	0	4	358
XC6SLX75	74,637	11,662	93,296	692	132	172	3,096	6	4	0	0	6	408
XC6SLX100	101,261	15,822	126,576	976	180	268	4,824	6	4	0	0	6	480
XC6SLX150	147,443	23,038	184,304	1,355	180	268	4,824	6	4	0	0	6	576
XC6SLX25T	24,051	3,758	30,064	229	38	52	936	2	2	1	2	4	250
XC6SLX45T	43,661	6,822	54,576	401	58	116	2,088	4	2	1	4	4	296
XC6SLX75T	74,637	11,662	93,296	692	132	172	3,096	6	4	1	8	6	348
XC6SLX100T	101,261	15,822	126,576	976	180	268	4,824	6	4	1	8	6	498
XC6SLX150T	147,443	23,038	184,304	1,355	180	268	4,824	6	4	1	8	6	540

Figure 9: FPGA Features. This image is taken from the Spartan 6 Datasheet.

The FPGA is the brain of the platform. It is the main component and the one in charge of taking all the decisions and implementing the energy saving methodologies.

The reasons why FPGAs are the best solution to face the requirements imposed by high performance sensor networks have been already discussed in the introduction chapter. The need of having very powerful processing capabilities working at very high speed is the perfect scenario for this kind of devices (Requirement 9). Besides,

the capability of loading different hardware modules with different sizes at run time, allows the network to be configured remotely and in a very efficient way.

An overview of the FPGA architecture, as well as the most important aspects about its dedicated I/O pins, is explained below.

1. Power Supply Rails

The FPGA is powered by six power rails: four input/output banks, the FPGA core and the auxiliary logic. The I/O banks need to be powered according to the components connected to them. The voltage range for these I/O resources varies from 1.2 V to 3.3 V. This range, together with the possibility of powering the banks at different voltages, allows maximum flexibility for the system architecture. The FPGA core is powered at 1.2 V, except in the case of the low power version that is 1 V to reach minimum power consumption. This is the primary power supply of the FPGA so it needs to be very stable. In the case of the auxiliary logic, it includes some configuration logic and some internal and I/O resources. The voltages permitted for this rail are both 2.5 V and 3.3 V. The selection of this voltage depends on the power consumption restrictions (a voltage of 2.5 V can reduce the power consumption in this rail in up to 40%) and on the voltage standard used by the I/O banks. Then, the different power rails are set to the following voltage values:

- Vccint: FPGA core at 1.2 V.
- Vccaux: Internal logic with configuration and I/O resources at 2.5 V.
- Vcco_0: Voltage supply for bank 0 powered at 3.3 V. As it can be seen in Figure 8, this bank is connected to the ADC, the external oscillator of 100 MHz, the communication layer (UART interface) and digital sensors.
- Vcco_1: Voltage supply for bank 1 powered at 1.8 V. This bank is placed in the reconfigurable side of the architecture. Some of the dedicated configuration pins are placed in this bank. They are connected to the external Flash memory.
- Vcco_2: Voltage supply for bank 2 powered at 1.8 V. As well as with bank 1, some of the dedicated configuration pins are placed in this bank. They are also connected to the external Flash memory. Apart from that, the external microcontroller for power management tasks is also connected to this bank.

- Vcco_3: Voltage supply for bank 3 powered at 1.8 V. This bank is connected to the external memories.
- GND pins: All of them will be tight to the digital ground of the PCB.
- VFS: Decryptor key EFUSE power supply pin for programming. The design of the PCB allows using it or not. In this way, if the bitstream encryption is not used, this pin can be tight to GND for a lower consumption. If the encryption is to be used, then it is powered through the 3.3 V rail.
- VBATT: Decryptor key memory backup supply. The same as the previous one. It can be used or not. If it is used it may be tight to 3.3 V using a 0 Ω resistor.

2. *Configuration pins*

The configuration of the FPGA can be carried out through two different paths: the JTAG port and the Flash memory device.

The JTAG connection will be used for debugging and configuration issues. Thanks to the Chip Scope Tool, it will be possible to monitor all the FPGA signals in order to check the correct behavior of the modules (Requirement 6). Besides, through the JTAG port, it is possible to download total and partial bitstreams which will be a key aspect when debugging and testing the new designs in the lab.

The FPGA has several initial configuration modes. The one chosen for this architecture is the Master Byte-wide Peripheral Interface (BPI) mode. This mode allows the FPGA to be configured directly and automatically from an external Flash memory device (Requirements 1 and 3). Regarding the Flash programming, the parallel interface permits a fast initial configuration. In the Master Configuration modes, it is the FPGA the one in charge of mastering the configuration clock (CCLK) while in the Slave modes the CCLK is an input. Therefore, this method follows the main concept of this platform: the FPGA is the brain of the system. The connection needed by this configuration mode is shown in Figure 10.

The pins needed for this configuration mode are mostly placed between banks 1 and 2:

- INIT_B: Before the mode pins are sampled, INIT_B is an input that can be connected low to delay configuration. After, INIT_B is an output indicating

whether an error occurred during configuration. It will be connected with a pull-up resistor since Program_B (hardware reset) is the pin used to delay configuration in this design.

- CCLK: Configuration clock output. This pin is not directly connected to the flash but is used internally to generate the addresses and sample read data.
- FCS: Chip selects Output. It selects the memory device. Connected to CE#.
- FOE: Output Enable. Connected to OE#.
- FWE: Write Enable. Connected to WE#.
- Address Output: A [1:23]. The FPGA will begin automatically in 0 and will increment by 1 until the bitstream is completely downloaded. Then, the pin DONE will be asserted.
- Data Input: D [0:15] It will be sample by the rising edge of CCLK, even though is not connected to the memory.
- CSO_B: Used for more than one memory in chain. Not connected.
- Mode pins: M [0:1]. They will be tight to GND for Master BPI configuration mode.

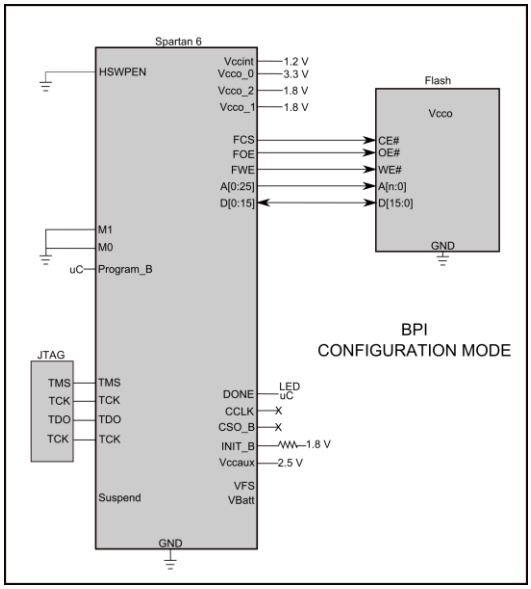


Figure 10: BPI Master Configuration Mode Connection.

Notice that even though the mode pins are selecting the BPI mode, the JTAG configuration can be used in any case since it is the one with the biggest priority. When the FPGA is powered, the address port starts providing addresses to the Flash memory. In this way, if the memory device is not correctly powered the configuration can be wrong. In order to ensure the correct configuration of the FPGA, it is possible to delay the addresses generation by holding the Program_B until the Flash memory reaches the correct voltage value. This can be done by the external microcontroller in charge of the power management.

All the above design considerations, together with a pre-placement analysis to identify possible problems in critical signals due to crossover noise or to excessive toggle rate or power consumption, were done. This analysis was performed with the assistance of the pin planning tools provided by Xilinx as shown in Figure 11 for all the components connected to the FPGA.

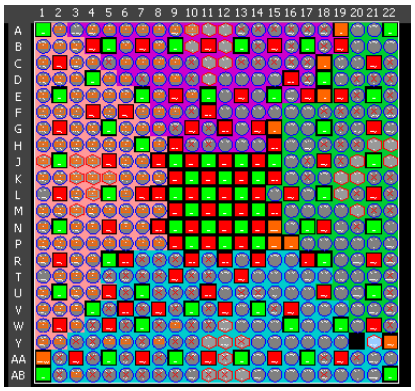


Figure 11: FPGA Pin-Out.

External Microcontroller

The external microcontroller chosen to execute the power management tasks implemented by the FPGA is the ATMEL AVR ATtiny2313V. This is a tiny microcontroller with 2k bytes programmable flash memory that can be powered at 1.8 V working in a frequency range of 0-4 MHz in a Micro-Lead-Frame (MFL) package of 4 x 4 mm. The power consumption in active mode running at 32 KHz is 20 µA while working in power down mode it can goes below 0.1 µA.

The microcontroller will be in charge enabling the seven different power islands and the interruptions caused by the external elements. The connection diagram is shown in Figure 12. The connection with the communication layer is done through an UART interface in order to be aware of the messages coming from the radio. Even though it is the microcontroller the component which actually executes the power management strategies, all the decisions of what to do and when to do it come from the FPGA. The communication between the FPGA and the microcontroller is made through the Universal Serial Interface (USI) that can be configured to work as a two-wire or three-wire connection. This is a pseudo I^2C or SPI protocols that can be easily configured as hardware modules inside the FPGA. The microcontroller also includes an analog comparator very useful to create interrupts if certain threshold voltage is overtaken by an analog sensor. This module compares two signals, the sensor response and a reference voltage that can be generated either by the microcontroller itself or by the power supply layer. Besides, a reset button is also included to reboot

the system manually and the chip reset of the FPGA, the Program_B pin, is connected to the microcontroller in case the delay of the initial configuration is necessary for any reason.

This external microcontroller will be the only component that is always powered in the platform. In this way, it can work as a sentry to wake the system up if necessary. The microcontroller works in power down mode with all the islands powered off until an external interrupt occurs. These external interrupts can be caused by: analog sensors, UART interface of the radio module, the FPGA or periodically using the internal clocks. The external microcontroller includes a JTAG connection for both programming and real time debugging (Requirement 6).

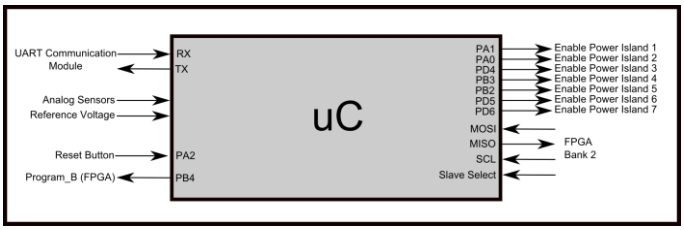


Figure 12: External Microcontroller Connection.

Flash Memory

The memory device selected is a Numonyx Strata flash Embedded Memory (P30) model JS28F128P30B85 with a 56-Lead TSOP package. This is a NOR flash memory recommended by Xilinx with 128 Mbits of capacity. It can be powered at 1.8 V which matches perfectly well with the rest of the power rails. The initial access time is 85 ns working at 52 MHz (more data can be found in the manufacturer datasheet).

The inclusion of a non-volatile memory device on the platform is due to four main reasons:

- Storing sensor data to be processed and sent through the network.
- Storing program data to be used by the embedded processor including in the FPGA.
- Working as the configuration bitstream source for the FPGA, after a switch-off period. Using the BPI mode explained before.

- Storing partial bitstreams of different hardware modules (Requirement 2). These modules are configured in the device at run-time only when they are required.

According to the proposed power management strategies that will be described in the following chapter, all the components included in the node will be switched off during sleeping periods. One of the drawbacks of using RAM based FPGAs is the fact that they do not keep their configuration when the power supply is cut off. Therefore, it is necessary to store an initial configuration file in a non-volatile device. In this way, using the automatic configuration mode (Master BPI Mode), the bitstream is automatically downloaded into the FPGA once both devices are powered on.

As it was mentioned before, in order to work as the initial bitstream source for the FPGA, the memory must be connected to the dedicated configuration pins of the FPGA. Nevertheless, the memory will also work as a storage device for other kind of information, so it needs to be accessible through another controller as well. Due to the restrictions imposed by the virtual architecture (the internal architecture within the FPGA), which will be explained in the following section, the memory controller cannot be placed close to the I/O banks 1 and 2 since they are within the reconfigurable area. Due to this reason, two different connections will be required for this memory. As it can be seen in Figure 8, the memory device is connected to the dedicated configuration pins in banks 1 and 2 and to bank 3 where the peripheral controller will be placed.

In Figure 13, the pin out of the Flash memory is shown. Notice that all pins, except the configuration clock (CCLK), are connected both to banks 1 and 2 and to bank 3. Even though the configuration mode is synchronous, the CCLK is not connected to the memory device although flash data is still sampled in every rising edge of the CCLK. The Write Protection pin (WP) is not used so it is asserted high in order to permit writing at all times.

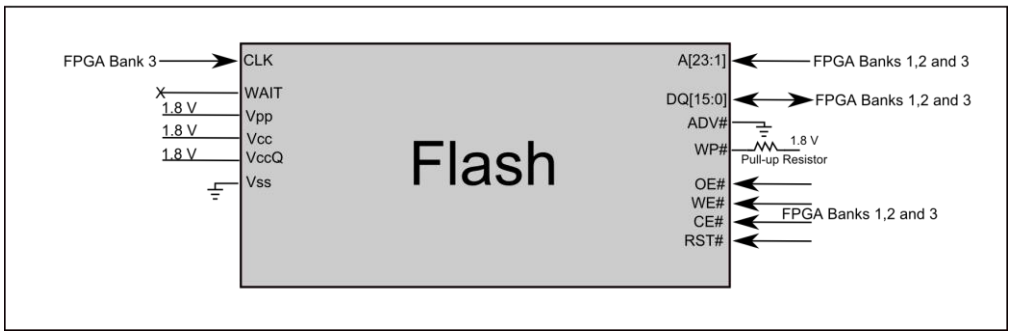


Figure 13: Flash Memory Connection.

It is important to notice that in order for the Flash to act as the initial bitstream source of the FPGA; this initial configuration file must be previously stored in the memory. Since the Flash memory is only connected to the FPGA, this storage must be done through the JTAG port. This storage can be done using the IMPACT tool provided by Xilinx. This is an indirect programming method where a small piece of IP is needed to connect the IMPACT software to the Flash memory through the Spartan 6.

RAM Memory

The RAM memory chosen is the Mobile Low-Power SDR SDRAM MT48H16M16LF from Micron with a 54-Ball VFBGA package. This memory is a high-speed 256 Mbits, CMOS dynamic random-access memory. The speed grade chosen is (-75) which corresponds to a 133 MHz with an access time of 5.4 ns. As well as in the case of the flash memory, it can be powered at 1.8 V. Read and write accesses to the memory are burst oriented. Accesses start at a selected location and continue for a programmed number of locations following a programmed sequence. Accesses begin with an ACTIVE command, followed by a READ or WRITE. The address bits registered in the ACTIVE command are used to select the bank and the row. The address bits registered in the READ or WRITE command are used to select the starting column location for the burst access. The device uses an internal pipelined architecture which is able to change the column address on every clock cycle in order to achieve high-speed and fully random accesses.

According to the methodology proposed in the introduction chapter, energy savings are increased by speeding up the processing. Due to this fact, a RAM memory has been also included in the node to allow fast information accesses by the embedded processor. Partial and dynamic reconfiguration speed can also have benefits by using an external RAM memory to store partial bitstreams, compared to the use of the external flash.

As it can be seen in Figure 8, the memory device is connected to the third bank of the FPGA. This is because the controller module that is normally used to master the memory, the MPMC provided by Xilinx, is a hard IP known as Memory Controller Block (MCB) which is placed both in bank 3 and bank 1. Since bank 1 is located in the side of the FPGA used for the reconfigurable blocks, the MCB used is the one in bank 3.

Analog to Digital Converter

Analog to digital conversion is needed to process the signals from the sensors (Requirement 4). Besides, the system has the ability of measuring its own power consumption in every one of the power islands, so the ADC is also required to convert these consumption values. The ADC chosen is the AD7928 from Analog devices. The AD7928 is an eight-input device with a SPI interface to communicate with the FPGA. It is an 8-bit resolution converter with a throughput rate of 1 MSPS that can be powered at 3.3 V which matches the voltage range imposed by the sensors and the power measurement circuitry.

The connection diagram of the ADC is shown in Figure 14. As it can be seen, half of the ADC inputs are dedicated to convert the power consumption of the power islands (Requirement 8). The most important ones, such as the island containing the FPGA core or the one with the RAM memory, have their own input, while the less used have been combined with other important islands like the Flash memory. The selection of which islands must be monitored is done externally by placing 0 Ω resistors. The fourth input is used to monitor the global consumption of the node that is measured in the power supply board. The rest of the ADC inputs are used to convert the signals coming from the analog sensors. As it was mentioned before, the

communication between the ADC and the FPGA is made through a SPI connection that will be detailed in the VA architecture of the FPGA.

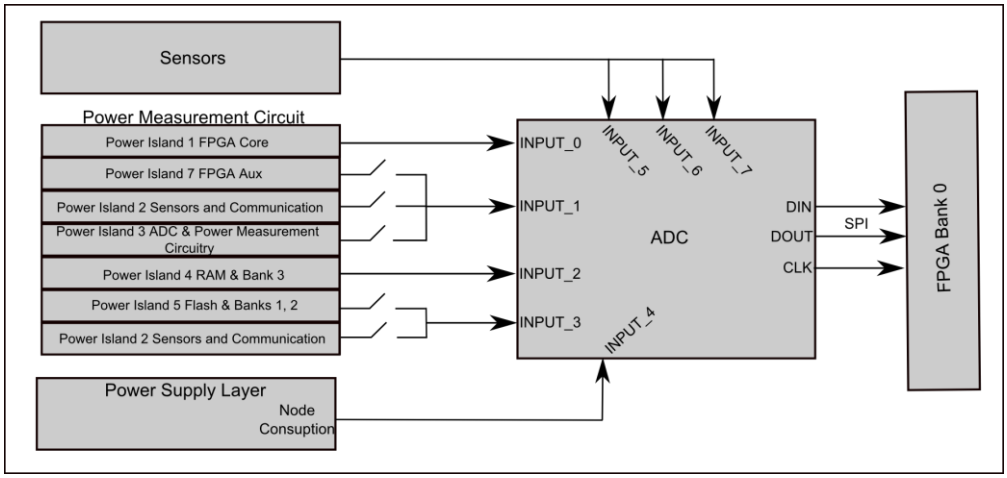


Figure 14: ADC Connection.

Power Management and Measurement Circuit

The inclusion of high performance components in the platform leads to very high energy consumption. Even though some of the components have their own power save modes, the static consumption is still too high (in the case of the FPGA: 60-80 mA) so the components need to be powered off when they are not needed.

The architecture proposed in this work is divided into seven different power islands that can be switched on and off separately depending on the system needs. Therefore, all the components can be switched off during sleeping time so that global power consumption is sharply decreased. In order to control these islands, it is necessary to include the ATtiny microcontroller that was explained above.

In Figure 15, every power island is represented in a different color. The islands are listed below (Requirement 11):

- Island 1: FPGA core. Powered at 1.2 V.
- Island 2: Sensor and communication boards. Powered at 3.3 V.
- Island 3: ADC and power consumption circuitry. Powered at 3.3 V.

- Island 4: RAM memory and bank 3 of the FPGA. Powered at 1.8 V.
- Island 5: Flash memory and Banks 1, 2 of the FPGA. Powered at 1.8 V.
- Island 6: External clock and Bank 0 of the FPGA. Powered at 3.3 V.
- Island 7: Auxiliary logic of the FPGA. Powered at 2.5 V.

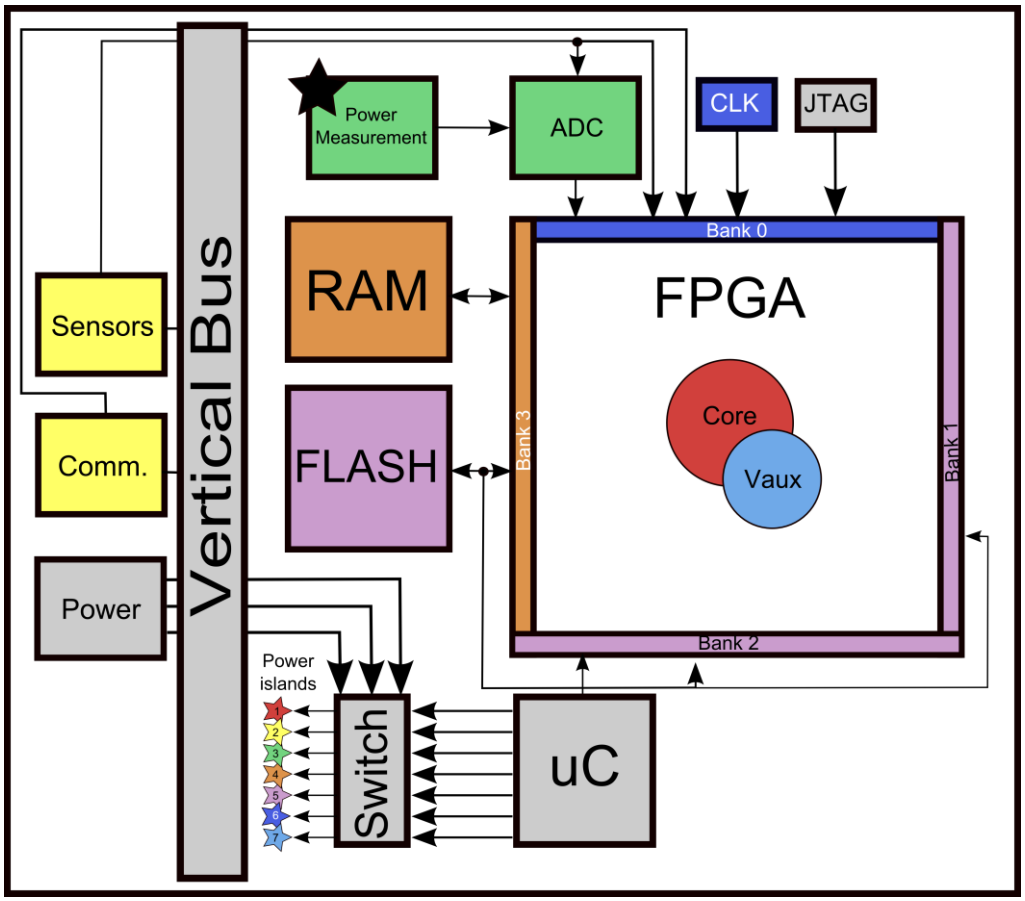


Figure 15: HiReCookie Architecture: Power Islands.

As it can be seen, four different power supply voltages are used: 1.2 V, 1.8 V, 2.5 V and 3.3 V. The FPGA core is the only one powered at 1.2 V and, therefore, it needs to be in an independent island. The 1.8 V supply is used to power banks 1, 2 and 3, the external memories and the external microcontroller. The external microcontroller is not included in the islands division since it needs to be powered at all times. During the initial configuration, the bitstream is downloaded from the flash memory

through the dedicated configuration pins which are within banks 1 and 2. Due to this reason both the flash memory and banks 1 and 2 belongs to the same island. The RAM memory and bank 3 are placed together in a different island, because the RAM memory controller, which is a hard IP of the FPGA, is located on the left side where this bank is located. In the case of the 2.5 V supply, it needs to be turned on at every time the core is powered. This auxiliary logic is not included in the island of the core because the power supply value is different and because it is necessary to have separated power consumption measurements for both rails. Regarding the 3.3 V rail, independent islands have been included to allow switching the sensor and communication layers separately from the power consumption circuitry.

The architecture of the power management circuitry is shown in Figure 16. As it can be seen, the external microcontroller acts as the enable source for seven different power switches. The switches chosen to work as the gates of the power islands are the TPS22907 switches from Texas Instruments. These are ultra-small switches with a R_{ON} Value of 44 mΩ and a maximum current of 1 A. In order to be able to measure the power consumption of every power island, shunt resistors (R_{shunt}= 0.01 Ω) after the power switches are included (Requirement 8). The voltage drop in every one of these resistors is conditioned by an instrumentation amplifier so that the value is converted by the ADC in order to be understood for the FPGA. The instrumentation amplifier used is the rail-to-rail micro-power INA333 from Texas Instruments.

The gain of the instrumentation amplifier is set by an external resistor. In this way, the FPGA is aware of the power consumption of every island so it can control the power supply of the node to implement energy saving methodologies. As it was mentioned before, the value of the global current consumption of the node is also measured by the power supply layer and converted by the ADC in order to have more information to implement better power saving policies.

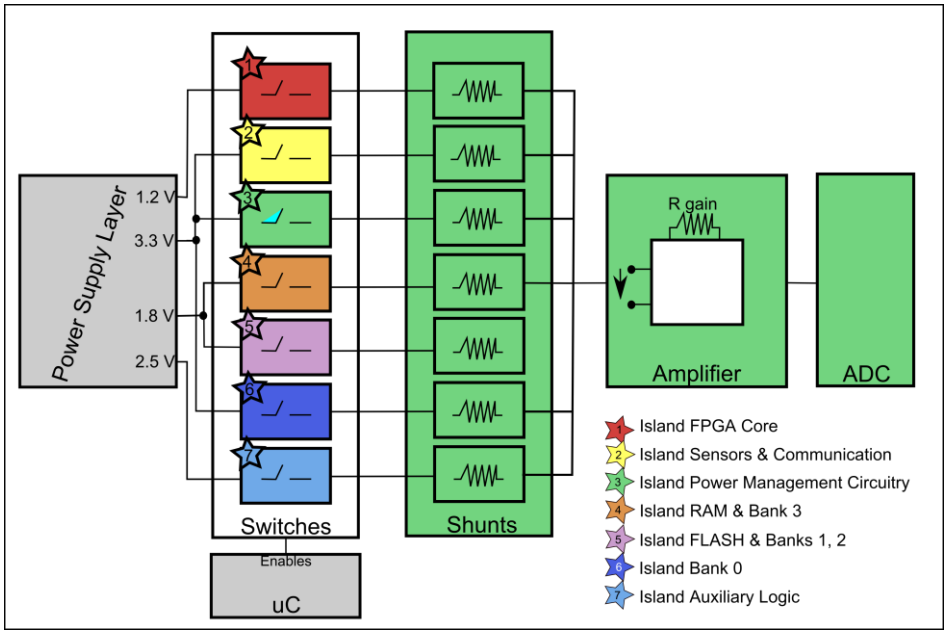


Figure 16: Power management circuitry.

II.3.2 Virtual Architecture

The Virtual Architecture (VA) refers to the internal module mapping of the FPGA defined by the designer. It describes the resources needed by the peripheral controllers and the reconfigurable hardware modules. Then, the VA refers to the size and position of the reconfigurable areas, the communication among them, as well as between reconfigurable and static parts of the system. The term Virtual Architecture comes from the virtual memory concept, widely used in operating systems [Fornaciari'98]. In dynamically reconfigurable system design, hardware virtualization allows the execution of complex applications on hardware platforms with insufficient resources. It also allows hardware module relocation between different virtual reconfigurable modules, and even between different reconfigurable systems with similar features.

The inner FPGA resources have been divided into two different regions: on the right side, the reconfigurable area is used to allocate changeable blocks at run-time (red and yellow blocks), and on the left side, the static area where the modules that

remain unchanged during the system life-time are placed. This division can be seen in Figure 17.

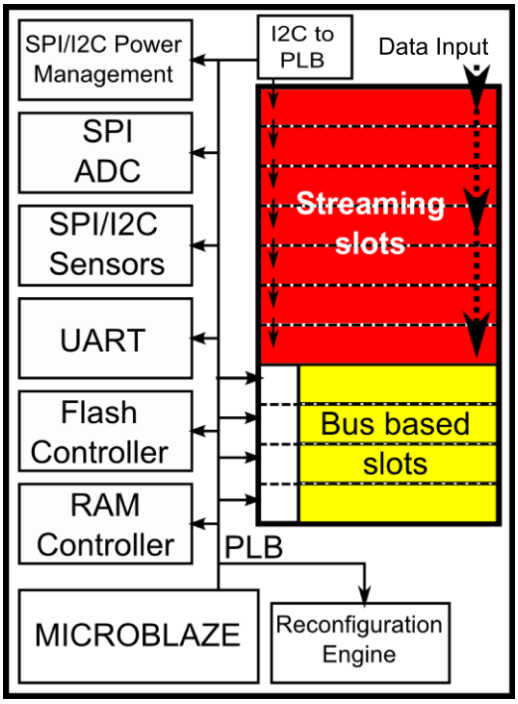


Figure 17: FPGA Internal Regions.

According to the results obtained with the Xilinx ISE tool, the static part occupies approximately 60% of the total area of the FPGA, leaving the other 40% for the reconfigurable blocks. Taking into account that the Spartan 3 XC3S200 used in the previous version of the processing layer has 4,320 logic cells and the Spartan 6 included in the HiReCookie has 147,443 logic cells, this 40 % dedicated for the reconfigurable region is almost 15 times larger than the whole resources of the Spartan 3 FPGA. Details about the static and reconfigurable regions are explained below.

Static Region

All the controllers required to interface with the external node components have been located in the static region. In addition, an embedded Microblaze processor

working as the system controller has been implemented in this part. The processor will be in charge of executing software tasks within the node, since unlike the previous *Cookie* platform, a general purpose external microprocessor has not been included in the mote. The rest of the hardware controllers and the reconfigurable accelerators are connected to this processor using a Processor Local Bus (PLB), following a System on Chip (SoC) approach. Together with the Microblaze processor, the main blocks are detailed below (Requirement 13):

- A reconfiguration module which makes use of the internal programming port (ICAP) to place reconfigurable blocks by writing into the associated areas of the configuration memory of the FPGA.
- Memory controllers, for both RAM and flash. These are the MPMC and EMC_MCH modules provided by Xilinx.
- UART interface to communicate with the ZigBee communication module.
- SPI interfaces to communicate with both, the external microcontroller and the ADC.
- Several I^2C and SPI peripheral modules for smart digital sensors interfacing.
- Chip-Scope peripheral for debugging issues.

Notice that this 60% area usage of the FPGA can be reduced by erasing the debug modules and some of the sensor controllers that may not be needed in all applications. The architecture of these hardware modules is detailed below:

1. Embedded Microprocessor: Microblaze

   The Microblaze embedded in the FPGA is a soft core with a RISC architecture optimized to work with Xilinx FPGAs. This is a flexible architecture that allows the selection of multiple features depending on the requirements of the design. Some of the fixed features of the microcontroller are: thirty-two 32-bit general purpose registers, 32-bit instruction word with three operands and two addressing modes, 32-bit address bus, Single issue pipeline, etc. All the configurable options can be seen on the part datasheet provided by Xilinx. The needed features can be selected using the ISE Xilinx Tool. The block diagram of the microcontroller is shown in Figure 18.

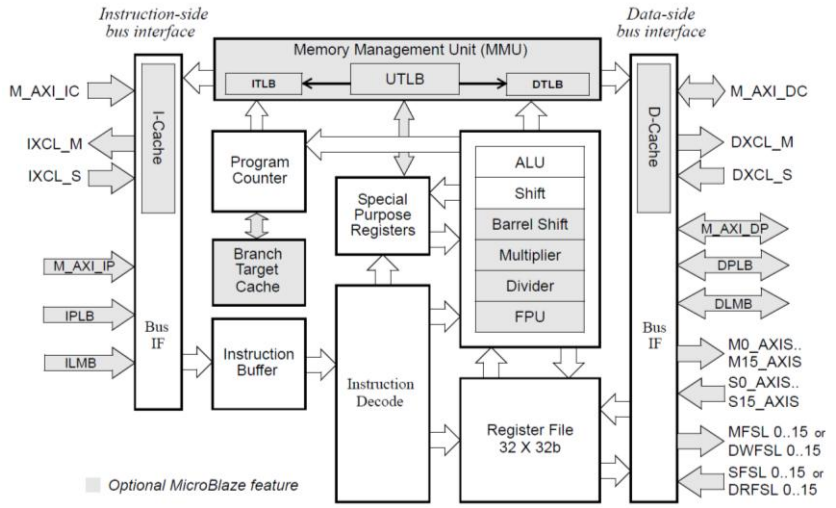


Figure 18: Microblaze diagram block. This image is taken from the Microblaze Datasheet.

The Microblaze processor is in charge of mastering all the information through the PLB bus. Since all the hardware modules are connected to this bus, the Microblaze will control all the external peripherals as well as the communication with the reconfigurable modules. In this way, the platform combines the speed of hardware with software flexibility (Requirement 13).

2. Reconfiguration Engine: Hardware ICAP Module

   The main feature of this block is its relocation capability, which means, the possibility of changing at run time the position of the reconfigurable blocks within the device (Requirements 2, 3 and 12). Thus, it is possible to allocate each reconfiguration engine in any empty slot in the reconfigurable region. Details about the operation of this module are explained in [Otero'11].

3. RAM Controller: MPMC

   The IP chosen to control the RAM memory is the Multi-port Memory Controller (MPMC). This is a hard IP block located in banks 1 and 3 of the FPGA. This is a parameterizable memory controller provided by Xilinx that supports quite a few memory types (DDR/DDR2/DDR3/LPDDR) and FPGAs (Spartan 3, Virtex 4, Virtex 5, Spartan 6 and Virtex 6). The IP contains

up to six ports that permit the communication with the internal Microblaze processor and other blocks in the device as well as the PLB bus interface. It also allows the configuration of a big number of options including the memory data width or the maximum frequency of operation. The internal architecture of the MPMC block is shown in Figure 19.

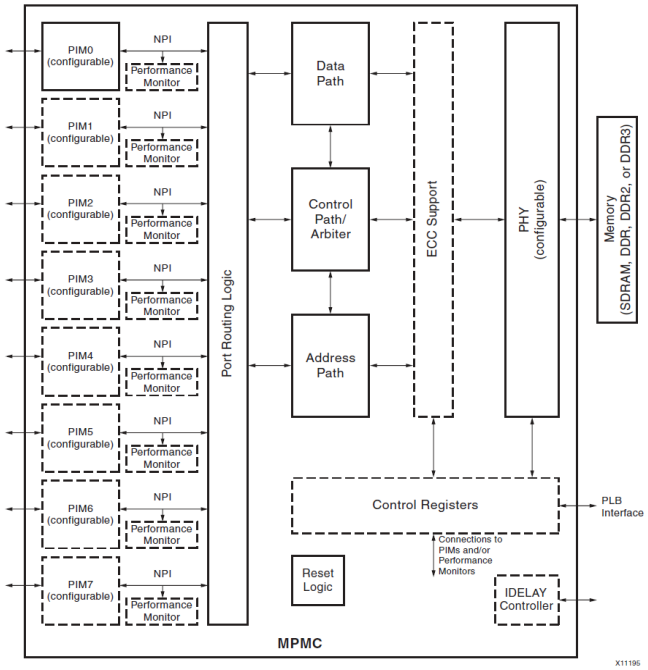


Figure 19: MPMC block diagram. This image is taken from the MPMC Datasheet.

As it is shown in the Figure, the connection ports can be chosen from a set of Personality Interface Modules (PIMs). These ports permit the connectivity with the Microblaze processor. The controller also includes an Error Correcting Code block and Debug Register support. In the case of the HiReCookie architecture the memory controller block that is used is the one located in bank 3.

The problem when working with this memory controller is that the RAM memory selected for the node is not directly supported by the software provided by Xilinx. The MPMC gives the opportunity of configuring the

interface with the memory but, in the case of the Spartan 6, a custom design is not supported. There are two different solutions to solve this problem. The first one implies the complete design of a new custom memory controller. Even though the manufacturer provides enough information for the development as well as a VHDL model of the memory, this task can be quite time consuming and it is not trivial. The second possibility is the edition of the libraries given by Xilinx in order to add a new memory model with the necessary configuration parameters. These tasks have not been solved yet, so they are included in the future work.

4. Flash Controller: EMC_MCH

In the case of the Flash memory, the controller is the Multi-channel External Memory Controller (EMC_MCH) which is also given by Xilinx. This multichannel external memory controller provides the control interface between different types of memories (synchronous and asynchronous SRAMs and Flash) and both the PLB and MCH protocols (PLB in this case). This IP block is connected to the PLB bus as a 32-bits slave and supports up to 4 external memory banks with single beat or burst transactions. This is a full configurable interface where parameters such as the data bus width and the access time, among others, can be configured to match with the memory requirements. The internal architecture of the controller block is shown in Figure 20. This block will be placed near bank 3.

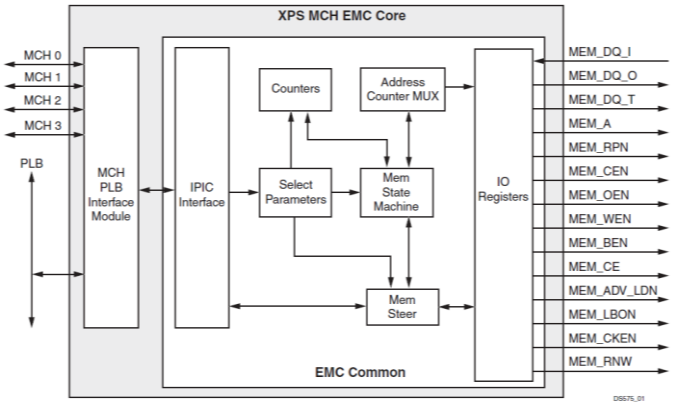


Figure 20: EMC_MCH Block Diagram. This image is taken from the EMC Datasheet.

5. UART Interface

The Universal Asynchronous Receiver-Transmitter interface (UART) is a low-speed communication protocol. The UART protocol consists mainly in a shift register to transform parallel communication into serial. This register normally includes start and stop bits to indicate that a new character is coming and finishing and an optional parity bit. The communication can be full duplex or half duplex. This protocol will be used to link the FPGA with the communication module in order to send data from one node to another, as well as to take general control of the network configuration process by issuing AT commands through this same interface. In this way, the network will be able to send either partial bitstreams to reconfigure the device or wake-up signals to implement the power management policies of the node. This module will be placed near bank 0 of the FPGA. As it happened with other protocols, Xilinx also provides an IP block that can be configured matching the needs of the system. The name of the peripheral controller is XPS 16550 UART which includes both the software and hardware tools needed for all the communication issues. Some of its features are: 32-bit Slave on PLB bus, hardware and software register compatible with all standard 16450 and 16550 UARTs, odd, even or no parity detection and generation, 1, 1.5 or 2 stop bit detection and generation, prioritized transmit, receive, line status and modem control interrupts, etc.

The XPS 16550 UART performs parallel to serial conversion on characters received from the CPU and serial to parallel conversion on characters received from a modem or microprocessor peripheral. The block diagram can be seen in Figure 21.

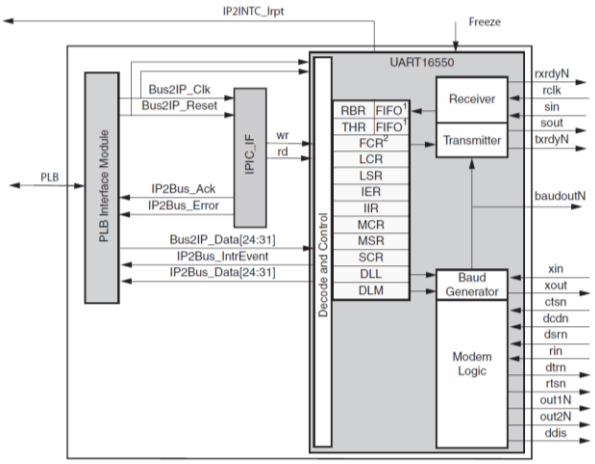


Figure 21: UART controller diagram block. This image is taken from the UART controller datasheet.

As it can be seen in Figure 21, there is a PLB interface module to provide bidirectional communication between the module and the PLB bus. The IPIC module is used to create acknowledges of the write and read transactions with the PLB bus. The UART 16550 module is fully configurable with different operation options such as baud rate, number of bits, slave acknowledges, etc. through different registers such as Line Control Register (LCR), the FIFO Control Register (FCR), etc. It also gives the opportunity of working with an external modem. In this design, only a 2-wire connection for the receiver and transmitter lines is going to be used.

6. SPI Interface

The Serial Peripheral Interface (SPI) is a low-speed communication protocol created by Motorola in 1979. One of the first uses of this protocol was in the Motorola 68000 microprocessor for the Apple Macintosh computer in 1984.

This protocol is normally used with one master device but multiple slaves. The SPI protocol usually uses four wires as it is listed below:

- A clock signal (SCLK) sent from the bus master to all the slaves. All the SPI signals are synchronized with this clock.
- A slave select signal (SS) for each slave. This is used to select the slave the master is communicating with.
- A line to send the data from the master to the slaves (MOSI).
- A line to send the data from the slaves to the master (MISO).

The structure of the hardware is shown in Figure 22. This protocol is also used trough a peripheral controller given by Xilinx (XPS_SPI) to communicate with some digital sensors and the external microcontroller.

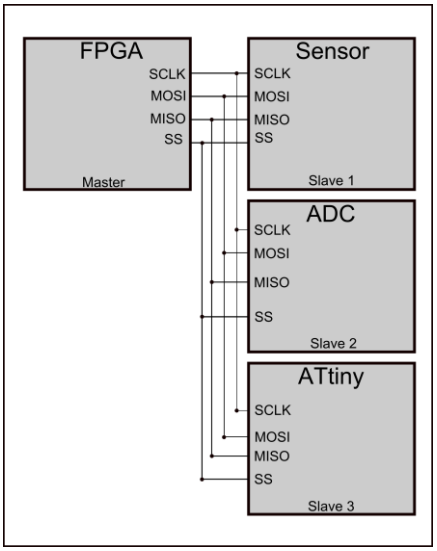


Figure 22: SPI connection with multiple slaves.

The interface provided by Xilinx is the LogiCORE IP AXI Serial Peripheral Interface. This is a standard SPI controller with configurable features that connects any SPI external device to the PLB bus. It has the option of being configured to act either as a master or as a slave. Since it is the FPGA the one in charge of controlling the system, the module will be configured as the

master device. The module supports Full-Duplex operation, programmable clock phase and polarity, continuous transfer mode for automatic scanning of a peripheral, etc. As it is shown in Figure 22, this protocol is used to interface with digital sensors, with the ADC and with the external microcontroller. Both the connection with the sensors and the ADC are done through bank 0 but the connection with external microcontroller is in bank 2. Due to this fact, the module should be placed close to bank 0 but at least four cables will be crossing the FPGA to connect with the ATtiny in bank 2.

7. I²C Interface

The inter-integrated circuit protocol (I²C) was developed by Philips in 1982 with the main goal of making the PCBs simpler by connecting the microcontrollers with their peripherals in a serial way. This protocol consists of only two wires, one for data interchange (SDA) and the other for clock synchronization (SCL). The original specification defined the bus speed as 100 kb/s but this specification has had several reviews, such as 400 kb/s in 1995 and, since 1998, 3.4 Mb/s for even faster peripherals. Some of the main features of this protocol are listed below:

- Resistant to glitches and noise.
- Supported by a large and diverse range of peripheral devices.
- A well-known robust protocol.
- A long track record in the field.
- A respectable communication distance which can be extended to longer distances with bus extenders.
- Easily emulated in software by any microcontroller.
- Available from an important number of component manufacturers.

The hardware structure of the I²C protocol can be seen in Figure 23. The system works using a master and slave devices. The master device is the one in charge of generating the clock signal and terminates the transfer. The master device can either receive data or transmit it to the slave. The slave

device is the one that is addressed by the master either for transmitting or receiving data.

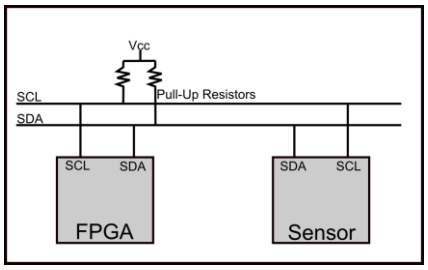


Figure 23: I^2C Connection.

Since the I²C is a standard serial communication protocol, Xilinx provides an I²C controller (XPS I²C) that can be configured to match the requirements of the system. This controller works as an interface between the peripheral and the PLB bus. The controller can configure the FPGA to work as a master or slave device or it can even work in a multi-master environment. Other features of the controller are: 7 bit or 10 bit addressing, fast mode 400 KHz operation or standard mode 100 KHz, bus busy detection, acknowledges bit generation/detection, filtering on the SCL and SDA signals to eliminate spurious pulses, interrupt generation, etc.

The internal architecture of the I²C block is shown in Figure 24. This module will be used to interface with digital sensors and, in the case the SPI protocol is not used, to control the external microcontroller.

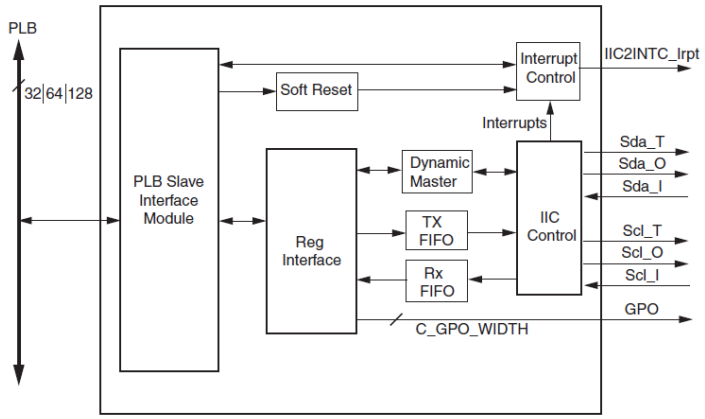


Figure 24: Internal architecture of the I^2C block.

Reconfigurable Region

The resources needed by the reconfigurable area have been selected considering the forecast space requirements of the changeable hardware modules.

Regarding the internal architecture of the FPGA, a frame is the minimum unit that can be reconfigured inside the device internal memory. However, it is still possible to come up with a VA including blocks with finer granularity. This will incur into higher reconfiguration cost, so this strategy would be only worthy if very small blocks have to be included in the system in order to reduce the internal area fragmentation. However, this is out of the scope of this work.

In Table 3, a few examples of the resource utilization are shown. These hardware modules correspond with two authentication algorithms that will be implemented later for the platform validation. Notice that the area requirements are in the order of the existing resources within each clock region of the FPGA.

The number of resources included in each slot is shown in Table 2.

Slices	Block RAMs	DSP48Es
960	12	8

Table 2: Resources available in every slot.

According to these values, the area division is shown in Figure 25, where the reconfigurable slots, the static region and the I/O Banks (IOBs) that will be used within the system are shown. As it can be seen, each slot will expand a full clock region. Available IOBs will only be those located in the static region where the external peripherals will be placed. In addition, no slots have been included both at the top and the bottom of the right side of the FPGA since the ICAP and BSCAN ports, which might be accessible from the static region, are placed in these areas.

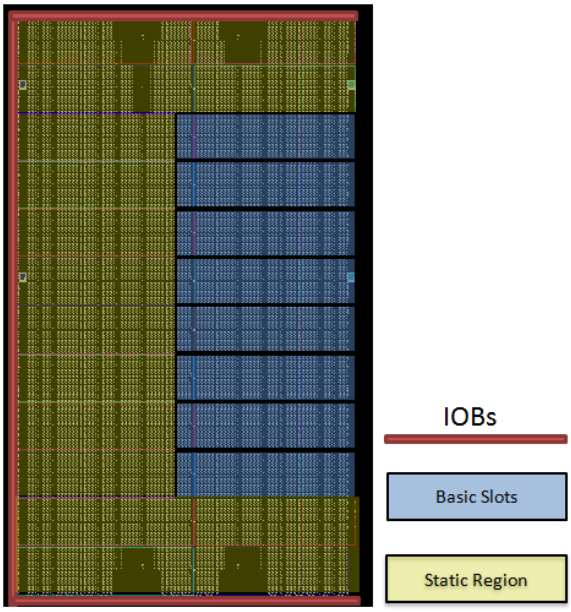


Figure 25: Area Division of the Internal Resources.

Resources	SHA1	MD5
DSP48Es	3 out of 180 (1%)	3 out of 180 (1%)
Block RAMs	4 out of 268 (1%)	8 out of 268 (2%)
Number of occupied slices	1,404 out of 23,038 (6%)	985 out of 23,038 (4%)

Table 3: Resource utilization summary corresponding with the Authentication Algorithms included within the node.

Comparing the available resource values in each slot with the requirements of the reconfigurable modules, it is clear that each one of these modules will fit into one or two slots. Therefore, it will be possible to merge these slots in a single reconfigurable area. The number of available slots for reconfiguration is eight.

Regarding the communication among modules, they can be classified in two groups: modules with bus-based communication dependence with the rest of the system, and modules with a streaming nature. The bus-based ones require a direct communication with the processor, which will be done using a shared bus. On the other hand, the streaming blocks require a direct point-to-point communication between them, without the intervention of the embedded processor. Typical Bus-based modules will be those in charge of the security and communication, while video coding and data compression will have a streaming nature.

These two communication possibilities have been included in the reconfigurable system design as it can be seen in Figure 17. Regarding the stream based nature; direct bus-macro based interfaces are included in fixed positions both at the top and the bottom of each slot, as shown in Figure 26. In addition of the point-to-point pipelined interfaces, which will be in charge of data transfer, a low-speed communication bus interface has been included, to allow the transfer of configuration information from/to the processor. Streaming interfaces has been also included in the border between the upper slot and the reconfigurable region. This interface will feed the system with data coming from the external sensors, mainly, the input camera. The lower interface of the last streaming based module will be connected to the rest of the system using a specific ending module, acting like a lid.

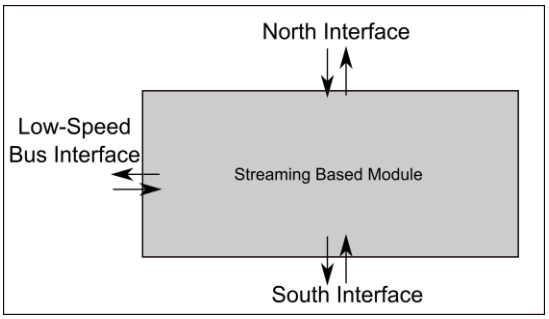


Figure 26: Streaming based modules.

Regarding the Bus-based slots, as shown in Figure 26, a bus interface has been included in the left side of each module. This bus will be also reconfigurable, since two versions of these bus slices have been considered, one including an access point, to exchange data with the attached reconfigurable module, and another version

without the access point, to allow slot merging. Hence, when a module occupies more than one slot, only one access point needs to be included. This sliced-bus will have a unique access point to communicate with the embedded processor as shown in Figure 17.

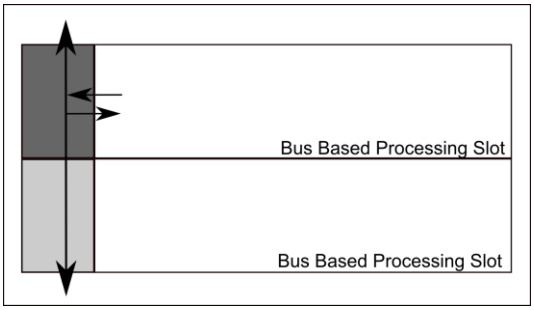


Figure 27: Bus based modules

III. PCB Design

Following the architecture shown in the previous section, a Printed Circuit Board (PCB) compliant with the *Cookies* platform has been developed. The PCB is a 60 x 40 mm board consisting of ten different layers as shown in Figure 28.

Layers (a) and (b) are the top and bottom faces of the PCB where the components are placed. Layer (c) includes both digital and analog ground planes. Layers (d), (e), (f) and (g) are used both for routing and placing the different power supply planes. Layers (h), (j) and (k) are routing layers.

Some other interesting features of the PCB are listed below:

- Size: 60 x 40 mm.
- Design divided in ten layers.
- Number of components: 215.
- Ball Grid Array (BGA) components such as the FPGA and RAM memory.
- Footprints: 0201, 0402, 0603, 1206.
- Total price of components: 209 € (FPGA is 115 €).

- Manufacturing price per unit when ordering three units: 60 €.

Soldering price for one unit: 700 €. The price is reduced if a big amount of PCBs are ordered.

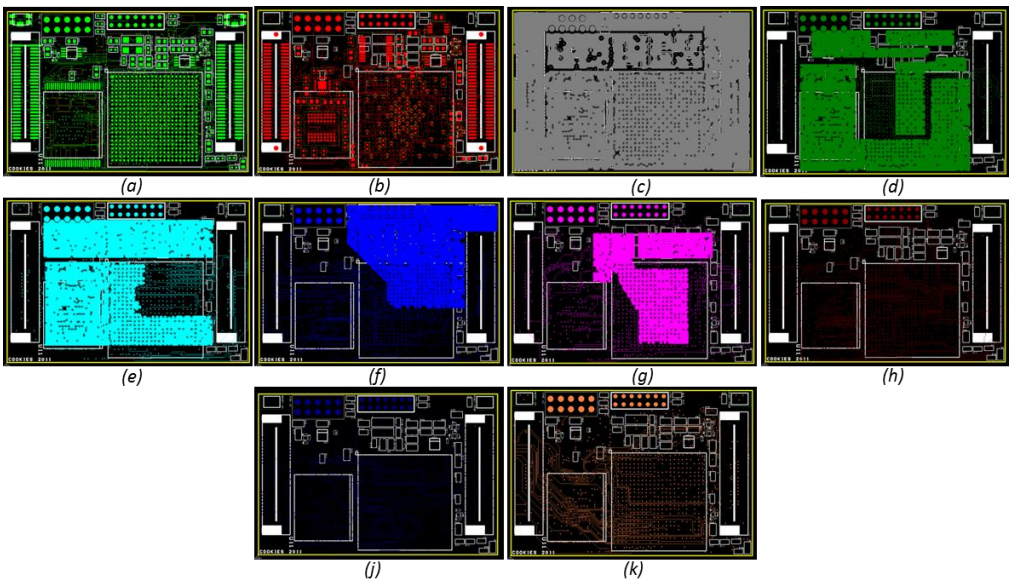


Figure 28: (a) Top Layer, (b) Bottom Layer, (c) Ground Layer, (d) First Power Plane Layer, (e) Second Power Plane Layer, (f) Third Power Plane Layer, (g) Fourth Power Plane Layer, (h) Routing Layer, (i) Routing Layer, (j) Routing Layer, (k) Routing Layer.

Due to the complexity of the board, the initial debugging tasks were difficult to accomplish. Due to this reason, an expansion board, including all the necessary check points has been designed to be connected to this processing layer. This board will be also used to power the processing layer in the previous lab tests since in this way it is easier to be aware of the instant power consumption in case any problem occurs. In Figure 29, both *Cookie* layers can be seen.

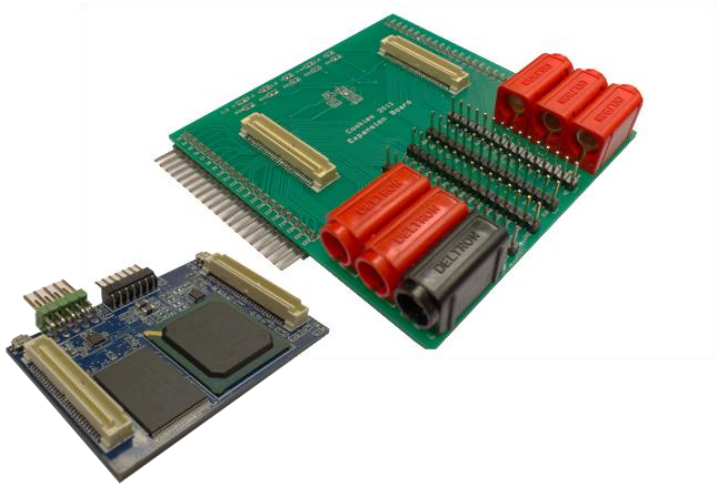


Figure 29: From left to right: HiReCookie in blue and Expansion board in Green.

Chapter 4

Energy Saving Methodologies

"Power is not sufficient evidence of truth".

Samuel Johnson.

4. Energy Saving Methodologies

WSNs are specifically designed to be autonomous and usually powered by batteries. Therefore, the way energy is handled within the node is one of the main concerns for WSN designers. In order for the batteries to last as long as possible, the nodes need to work most of the time in a sleep mode (current consumption in the order of μA or nA). Moreover, in a WSN that is specially designed to face very demanding applications, the inclusion of high performance devices, such as the FPGA or the memories, leads to higher power demands, so that, this concern becomes even more critical. Even though most of these components have their own power saving modes, their static consumption is still too high to be considered appropriate for WSN standards. Due to all these reasons, energy saving methodologies need to be included in the architecture design.

The standard working profile of a wireless sensor node is shown in Figure 30.

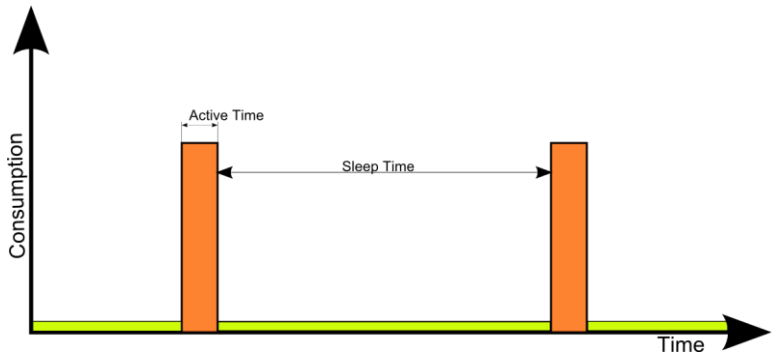


Figure 30: Working Profile.

According to the figure, the nodes are awake only during the time needed to carry out certain task. During the rest of the time the nodes are working in a sleep mode. Following this working approach, the global energy consumption can be optimized by making the duty cycle as low as possible while keeping the current consumption during sleep time negligible. In order to achieve this critical goal, the active time should be as short as possible, so that very quick processing engines are very helpful.

Taking into account that the working profile is divided into these two regions, different energy saving methodologies can be implemented.

Sleep Time

The main feature of this period is that it should be as long as possible while it should have a negligible current consumption. This issue is achieved by powering all components off except of the external microcontroller and the component that triggers the wake-up event. Both the microcontroller and the component that acts as a trigger will be working in power down mode when possible. Depending on each application, there will be different available sleep modes based on the control of the power islands together with the management of the power saving modes of the components that remain powered.

In order for the system to be capable of waking up when an external event occurs, the external microcontroller, the power supply board and the source that triggers the external event must be powered at all times. The ATtiny controller includes three different power modes: Idle, Power Down and Standby. The Standby mode will not be used since it requires an external oscillator that is not included in this platform. The microcontroller can wake the system up using different sources of interruption. The selection of these sources depends on the power mode that is being used so that it defines how deeply the system is sleeping. The different possibilities are listed below:

- ATtiny working in **Idle mode** (10 μA at 0.1 MHz): In this mode, an interruption coming from the radio (UART), any internal timer, a threshold in a sensor value or an interruption caused by the FPGA (if it is awake), can wake the microcontroller up.
- ATtiny working in **Power down mode** (< 0.1 μA): In this mode, only an interruption caused by the FPGA and the watchdog timer can wake up the microcontroller. When the node is working inside the Sleep region, the FPGA is not powered so the only way to wake up the microcontroller is the watchdog timer.

Therefore the sources of interruption to wake the system up can be: the radio module, analog sensors or internal timers.

The radio module includes four different power modes as seen in Table 4. Every one of these modes has different power consumption that depends on whether the node is working either as a router or as a coordinator, or if the node is working as an end-device.

Power Mode	Router or Coordinator Device	End Device
Awake	36 mA	9 mA
Idle	32 mA	4.5 mA
Asleep 1	0.7 mA	0.7 mA
Asleep 2	0.7 µA	0.7 µA

Table 4: Communication module. Power-down modes.

When the node is working as a router or as a coordinator, the communication module should be working in awake-state since the rest of the network depends on its behavior. When the module is working in idle mode, it can be woken up using commands sent from the external microcontroller or the FPGA. However, if a deepest sleeping mode is required, such as Asleep 1 or Asleep 2, the ATtiny has to trigger an external interruption to wake the module up, or in the case of Asleep 1, internal timers can be used.

In the case of the analog sensors, there is no power save mode available for them. When the system is working in sleep mode waiting for an analog sensor to cross a certain threshold value, the sensor layer needs to be powered but no power save mode is possible. Therefore, the current consumption will depend on the sensors used. The solution for that is powering the sensors island periodically in order to check if a certain value has been crossed.

If the only source of interruption is an internal timer, the watchdog can be used so that the system can be completely switched off with the exception of the microcontroller and the power supply layer, which can be both working in power down mode. In this way, the system can be woken up periodically while during the rest of the time it is working in a very deep sleep mode.

In the case of the power supply board, the power mode changes depending on the current consumed by the load. Therefore, during sleep time, when the majority of the islands are powered off, the current consumption is very low, so the DC-to-DC converter enters into power down mode (it happens when the current consumed by the load is less than 150 mA, which is the majority of the times).

Regarding the use of the Sensors/Communications power island, if both layers are connected on the top of the processing layer, they can be switched on and off but never separately. If it is necessary to have one of the layers always powered, it should be connected manually to the bottom of the processing layer as it was explained in Chapter 3. Since it is possible to take the communication module to a very deep sleep mode (0.7 μA), the communication layer will be usually placed under the power supply board, so it is powered at all times.

All the different sleep modes are detailed as follows:

Sleep Mode 1: This is the deepest sleep mode. The ATtiny can be woken up using the watchdog timer and then it can power the communication module and the sensor layer together in case a message from the radio or any sensor threshold is being crossed. In case none of these events occur, the ATtiny can go back to power down mode.

Islands	On- Off State	Power Mode	Wake up Possibilities	Current Consumption
FPGA core (1)	OFF	-	Watchdog.	0.1 μA (ATtiny) + 1 mA (Power Supply).
Sensor and communication boards (2)	Sensors OFF Comm. OFF	-		
ADC and power consumption circuitry (3)	OFF	-		
RAM memory and bank 3 (4)	OFF	-		
Flash memory and Banks 1, 2 (5)	OFF	-		
External clock and Bank 0 (6)	OFF	-		
Vccaux (7)	OFF	-		

ATtiny	ON	Power Down		
Power Supply Layer	ON	Power Down		

Table 5: Sleep Mode 1.

Sleep Mode 2: In this mode, since the ATtiny is working in power down mode, only the watchdog timer can wake it up. Nevertheless, it is possible to be working in a very deep sleep mode while the sensor layer is not powered. Once the ATtiny is woken up by its timer, it can send an interruption to the communication module to wake it up or it can power the sensors island to check if a threshold value has been crossed, in this case, the communication module can be placed under the power supply board so it is always powered.

Islands	On- Off State	Power Mode	Wake up Possibilities	Current Consumption
FPGA core (1)	OFF	-	Watchdog, Radio.	0.1 µA (ATtiny) + 1 mA (Power Supply) + 0.7µA (Radio module)
Sensor and communication boards (2)	Sensors OFF Comm. ON	- Asleep 2		
ADC and power consumption circuitry (3)	OFF	-		
RAM memory and bank 3 (4)	OFF	-		
Flash memory and Banks 1, 2 (5)	OFF	-		
External clock and Bank 0 (6)	OFF	-		
Vccaux (7)	OFF	-		
ATtiny	ON	Power Down		
Power Supply Layer	ON	Power Down		

Table 6: Sleep Mode 2.

Sleep Mode 3: In this mode, the ATtiny is also working in power down mode. When the timer wakes it up, it can change to idle mode in order to check if the sensors have crossed the threshold value. This mode only makes sense if the sensor response is critical due to a dangerous parameter where it is not possible to wake the sensor island up and wait until the sensor measurement is stable. A variation of this method

is the same configuration but the ATtiny working in idle state in order to have an instant response in case a problem on the measurement occurs.

Islands	On- Off State	Power Mode	Wake up Possibilities	Current Consumption
FPGA core (1)	OFF	-	Watchdog, Analog Comparator	0.1 µA/10 µA (ATtiny) + 1 mA (Power Supply) + Sensors layer
Sensor and communication boards (2)	Sensors ON Comm. OFF	-		
ADC and power consumption circuitry (3)	OFF	-		
RAM memory and bank 3 (4)	OFF	-		
Flash memory and Banks 1, 2 (5)	OFF	-		
External clock and Bank 0 (6)	OFF	-		
Vccaux (7)	OFF	-		
ATtiny	ON	Power Down/ Idle		
Power Supply Layer	ON	Power Down		

Table 7: Sleep Mode 3.

Sleep Mode 4: This case is a combination of the previous two modes. In this way, it is possible to wake the system up using the sensors response or a possible message coming from the radio.

Islands	On- Off State	Power Mode	Wake up Possibilities	Current Consumption
FPGA core (1)	OFF	-	Watchdog, Analog Comparator, Radio.	0.1 µA (ATtiny) + 1 mA (Power Supply) + Sensors layer + 0.7 µA (Radio).
Sensor and communication boards (2)	Sensors ON Comm. ON	- Asleep 2		
ADC and power consumption circuitry (3)	OFF	-		
RAM memory and bank 3 (4)	OFF	-		
Flash memory and Banks 1, 2 (5)	OFF	-		
External clock and Bank 0 (6)	OFF	-		
Vccaux (7)	OFF	-		
ATtiny	ON	Power Down		
Power Supply Layer	ON	Power Down		

Table 8: Sleep Mode 4.

Sleep Mode 5: This sleep mode can be useful if a faster response is required since it is not necessary to wake up the microcontroller.

Islands	On- Off State	Power Mode	Wake up Possibilities	Current Consumption
FPGA core (1)	OFF	-	Watchdog, Analog Comparator	10 µA (ATtiny) + 1 mA (Power Supply) + 0.7 µA (Radio).
Sensor and communication boards (2)	Sensors OFF Comm. ON	- Asleep 2		
ADC and power consumption circuitry (3)	OFF	-		
RAM memory and bank 3 (4)	OFF	-		
Flash memory and Banks 1, 2 (5)	OFF	-		
External clock and Bank 0 (6)	OFF	-		
Vccaux (7)	OFF	-		
ATtiny	ON	Idle		
Power Supply Layer	ON	Power Down		

Table 9: Sleep Mode 5.

Even though there are more possible combinations, these are the most useful ones.

Active Time

The active period can be also divided into two different regions as it is shown in Figure 31. As it was detailed in the previous chapter, the FPGA used is a RAM-based one. Due to this fact, once the device is switched off, all the information is deleted so that every time the FPGA is powered up, it must be configured again.

In this way, there are two regions where power saving methodologies can be implemented: Configuration and computing time.

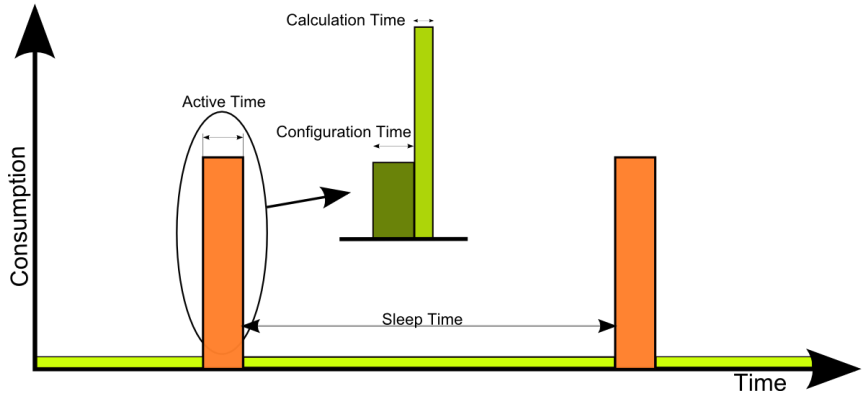


Figure 31: Active Time Profile.

During configuration time all the rails of the FPGA must be powered. This fact implies an instant current consumption of 80 mA (Islands 1, 4, 5, 6 y 7 powered). Even though this current consumption is too high, the main goal is to reduce this configuration time as much as possible in order to make the area (energy) as small as possible. The initial configuration file is stored in the Flash memory so that it is automatically downloaded in the FPGA once the system is powered up. The size of the bitstream is very significant in terms of power consumption since it is directly related to the time needed for configuration. The smaller the configuration file, the shorter the configuration time becomes. In this way, there are two strategies that can be combined to reduce the size of the configuration file. On the first hand, the method consists of loading a bitstream file with the minimum information required by the system while on the other hand it is possible to compress this bitstream file. The compression of this bitstream can be done through two different methods.

The first method consists of using the Xilinx ISE Tool. This method allows up to 65% of compression that reduces significantly the configuration time. Even though this is already an important reduction, the compression can be still optimized. The second method consists of creating a partial configuration file. This work follows the line reported in [Hübner'10] as part of another work that is being carried out in the Center of Industrial Electronics, but it is not finished yet. The followed approach is based on the fact that the FPGA is empty by default, that is, the bitstream is full of zeros. In this way, it is possible to erase them so that the size of the bitstream would

be reduced without losing any information. The start point for this optimization can be either a full or a compressed bitstream.

The basic configuration unit of the FPGA is called frame. Every frame consists of 65 words, so if the same frame needs to be written a fixed number of times, it can take up a lot of space. Nevertheless, in the case of the compressed one, when a frame is repeated a certain number of times, the frame is stored in a register called Multiple Write Frame Register (MWFR) followed by a command to specify the frame address (FAR address). The procedure is shown in Figure 32.

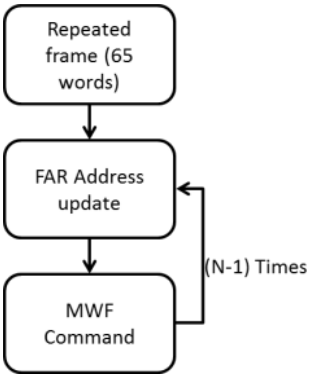


Figure 32: Compression Algorithm

Considering the structure of the compressed bitstream, the best way to reduce the configuration time is identifying these empty frames and erasing the MWF commands. This strategy deletes all the redundant zeros regarding the CLBs information. The MWF command is not used for the BRAMs configuration, so it will be necessary to check frame by frame to erase the empty ones. After checking different bitstreams and different times of configuration, it was observed that the results could be approximately represented by a straight line as shown in Figure 33.

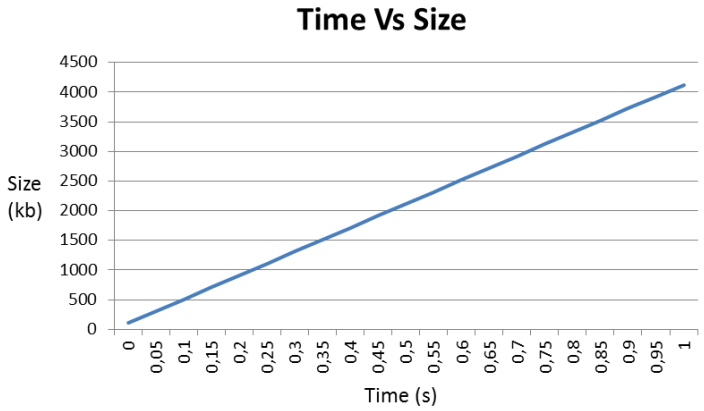


Figure 33: Theoretical evolution of the configuration time depending on the size of the bitstream.

Thus, even though the FPGA has not been configured with an initial-partial bitstream yet, it is possible to estimate the time that would be required. In Table 10, a comparison in terms of size and time between these different compression methods for the same bitstream is shown.

Initial Configuration	Size (Kb)	Time (s)	Consumption (mAh)
Total Bitstream	4,122	1.002	0.02220
Compressed Bitstream	1,416	0.353	0.00780
Compressed-Partial Bitstream (CEI)	800	0.172	0.00381

Table 10: Bitstream Comparison

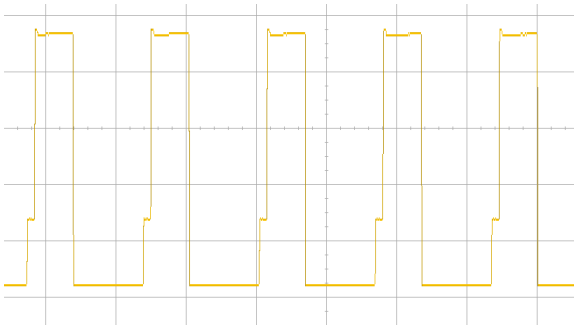
By compressing the size of the bitstream file, an 80% of time reduction can be achieved in this period, as it is shown in Table 10.

If in addition to the file compression, the initial bitstream only contains the minimum information required, the configuration file is even smaller so the reduction of time is very significant. This method only makes sense if the platform allows dynamic and partial reconfiguration. Thus, once the platform is working, different hardware blocks can be loaded at run time depending on the application needs. Regarding the region where the calculations are carried out, the capability of running different tasks in parallel together with the possibility of powering up only the components

that are strictly needed, lead to very short computing time with the minimum consumption possible. A representative example of how these methodologies are implemented is shown in the next chapter.

Chapter 5

Platform Tests



"The unexamined life is not worth living"

Socrates.

5. Platform Tests

I. Scenarios & Applications Examples

Before starting with the validation tests, some application scenarios will be shown in order to illustrate the usability of the platform. Even though these applications are still examples that have not been implemented yet, they are very useful for the correct understanding of the possibilities offered by this high performance sensor node. After the explanation of these possible scenarios, real tests have been run emulating these real conditions.

I.1 Example 1: Surveillance in Restricted Military Areas

Surveillance inside restricted military areas becomes a crucial matter since the environment changes as the camp moves through different positions. In this situation, wired systems are not an appropriate solution. The possibility of having wireless sensor nodes with the capability of dealing with multimedia applications seems to be a perfect solution for a system that needs to change its location very often.

Considering a deployment in a restricted military area where the nodes are divided into two different levels: several low-profile nodes including vibration sensors and a HiReCookie node including a video camera. The low-profile nodes are in charge of giving an alarm if any vibration is detected. Once the event occurs, this node must send a warning message via radio to the HiReCookie node. In the meanwhile, the HiReCookie node must be working in a very deep sleep mode so that it is not woken up until the use of the video camera is required. The behavior of the node is detailed in the next figure.

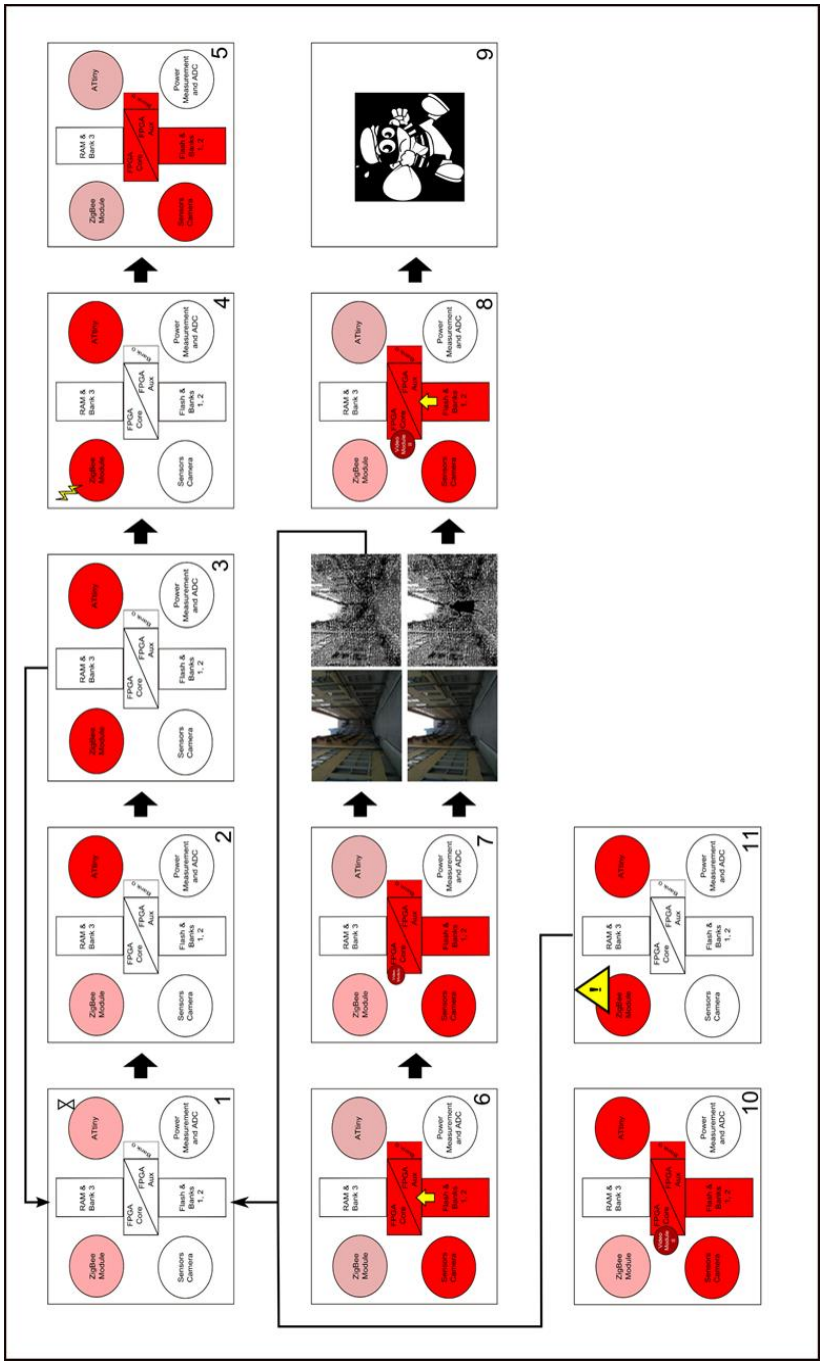


Figure 34: Example 1.

1. The HiReCookie node is working in Sleep mode 1. Everything is switched off with the exception of the microcontroller and the communication module which are both working in a power down mode. The microcontroller is programmed to wake up periodically.
2. The Watchdog timer wakes the microcontroller up. The microcontroller sends an interruption signal to wake up the communication module.
3. Both the microcontroller and the ZigBee module are awake and waiting for a warning message to arrive from any of the low profile nodes. If no message arrives in a fixed period of time, the node goes back to point 1.
4. If the ZigBee module receives a warning message, this means that something has activated the vibration sensors, so there is the possibility of an intruder presence.
5. The ATtiny wakes up the next islands: island 1 (FPGA Core), Island 2 (Sensor Layer, Video camera), Island 5 (Flash memory and FPGA banks 1 and 2), Island 6 (FPGA bank 0 and external oscillator), Island 7 (Auxiliary Logic). After that, both the microcontroller and the ZigBee module change to sleep mode again.
6. Then, since the Flash memory was already loaded with an initial bitstream that includes:
    a. Static Region: SPI controller to master the communication between the FPGA and the external microcontroller, the embedded processor (Microblaze), Flash memory controller, Hardware ICAP module to relocate additional hardware modules and the UART interface for the communication between the FPGA and the ZigBee module.
    b. Reconfigurable Region: Simple video module to compare a reference image with the one taken by the video camera.

    This bitstream file is automatically downloaded to configure the FPGA.

7. Now the node includes all the hardware blocks mentioned before and it is ready to take the image and compare it with the reference one. If the image taken by the camera is equal to the reference, the node goes back to point one since it considers that no intruder has been detected. If the image is different, more resolution is needed in order to be sure that the image detected is a real intruder.
8. A video processing hardware block with bigger resolution than the previous one is taken by the Flash memory controller and relocated by the Hardware ICAP module to replace the old one.
9. Then, more image resolution is obtained so it is possible to be sure if the presence detected is an intruder or not.
10. If the presence is considered as an intruder, the FPGA wakes the microcontroller up, and so does the microcontroller with the ZigBee module.
11. Finally, an alarm is sent from the HiReCookie to the base station.

As it can be seen, the high performance node is only awake if a second level event occurs. During the time when no event is detected, the node is kept in a very deep sleep mode with only a few μA of current consumption. Besides, even the functionality of the HiReCookie node is divided into two different levels of precision in order to keep power consumption as low as possible.

I.2 Example 2: Data Logging and Authentication

Security in WSNs has become a crucial aspect nowadays. The possibility of discarding untrustworthy information produces an increase in the reliability of the system. There are different authentication algorithms in the state of the art. In this case, both for this example and for the tests running on the platform, a SHA1 (Secure Hash Algorithm) has been used. This is a well-known authentication algorithm used to check message integrity. The algorithm generates a unique signature for each data block.

Considering a scenario where a WSN deployment has been divided into different node clusters, the network architecture could be as it is shown in Figure 35.

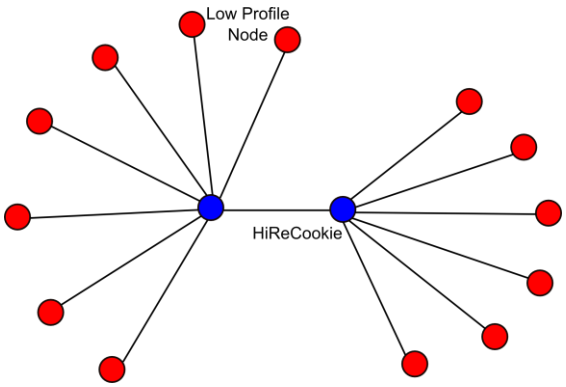


Figure 35: Network Architecture Divided into two different Clusters.

In Figure 35, two different types of nodes are shown. The low profile nodes are in charge of taking different measurements and sending them to the head clusters which, in this case, are the high performance nodes. The HiReCookie is able to store this data in the non-volatile memory. Once a sufficient amount of data blocks has arrived, the HiReCookie can use the SHA1 algorithm to obtain the signature of each one of these data blocks. It is important to highlight that, as it will be demonstrated within the platform validation tests, the use of a high performance system is only recommendable when a large amount of information needs to be processed. Thus, hardware acceleration together with a big memory capacity, are very convenient for this kind of applications.

As it was done in the previous example, the basic steps for that application are shown in the next figure.

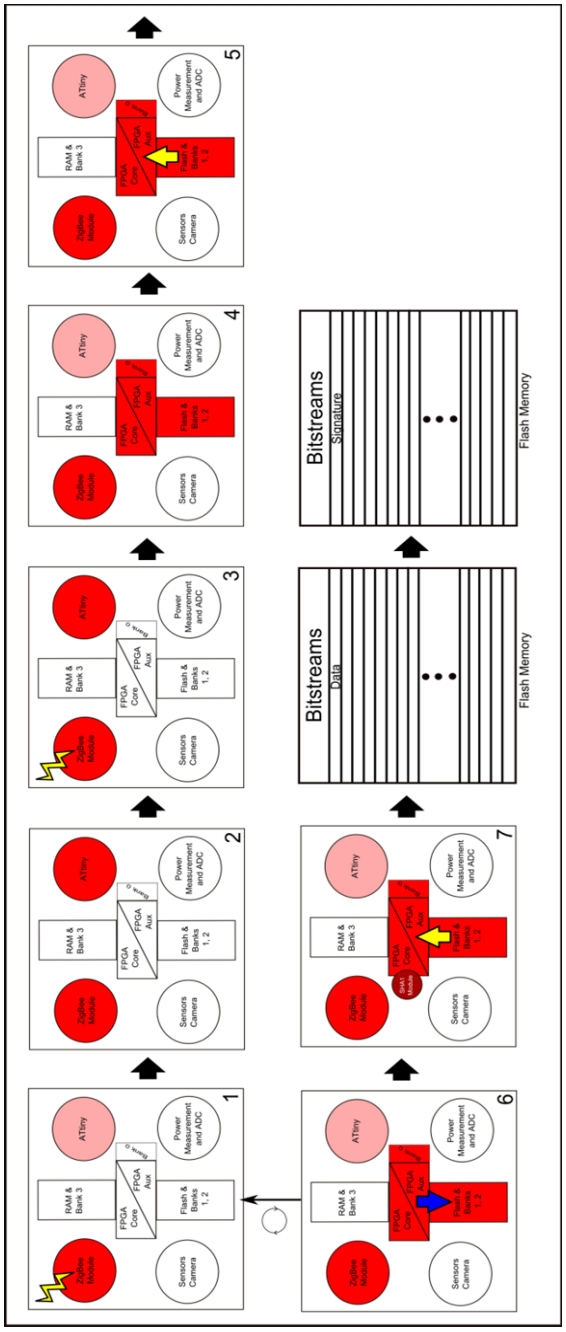


Figure 36: Example 2.

1. Since the HiReCookie node is now working as router node or coordinator of the cluster, the communication module cannot be in any sleep mode. The rest of the node will be switched off with exception of the ATtiny that will be in power down mode. The low profile nodes have to send an initial message to wake up the HiReCookie, and a second one containing the information.
2. If an initial message from any of the measuring nodes arrives, the communication module wakes the microcontroller up using an external interruption.
3. Then the node keeps waiting for the next message with the information.
4. When the message arrives, the ATtiny power the FPGA and Flash memories and change its mode to power down mode.
5. Automatically, an initial bitstream is downloaded from the Flash memory to configure the FPGA. This bitstream must contain: Hardware ICAP module, Flash controller, Microblaze and the UART interface.
6. Then the data is stored in the non-volatile memory. The node goes back to the initial state to wait for another message to arrive.
7. Once a sufficient number of data blocks are stored in the memory, a partial bitstream containing the SHA1 module is loaded to encrypt the information.

Thanks to this methodology, it is possible to discard those messages that have not arrived from a known device or those that are not complete. Besides, the possibility of having encrypted information stored in the flash memory, allows a faster access since the size of the encrypted data is smaller than the original one, while at the same time they take up less memory usage. Apart from these advantages, once the data stored in the HiReCookie is needed by the system, it can be delivered in a secure way since all the information is encrypted.

II. Validation Tests

The validation tests carried out on the HiReCookie platform have had three main goals: checking hardware design, showing the flexibility and performance of the

platform and demonstrate the higher energy efficiency and speed compared with software based solutions.

II.1 Hardware Validation

Hardware validation has been done through different basic tests in order to check the proper behavior of the platform. These validation tests must show the correct functionality of the components as well as the communication between them.

First of all, the external microcontroller was programmed via JTAG using the AVR Studio 4 software tool. The first code had the only purpose of powering the seven power islands at the same time in order to check if all the components were powered properly. Once every island was checked, the software code was changed to open and close the islands in different groups to check their correct behavior under different power configurations.

Once the external microcontroller was programmed and all the islands were tested, next step was detecting and configuring the FPGA using the JTAG port. Therefore, a software code to keep all the islands powered was loaded into the ATtiny microcontroller. In order to verify the correct connection of the FPGA, simple hardware modules were loaded. The output signals of these blocks were connected to the vertical connectors so that the wave forms could be seen using an oscilloscope connected to the expansion board.

Once the connection of the FPGA was checked, it was possible to verify the functionality of the Flash memory. Using the IMPACT software tool provided by Xilinx, the bitstream of one of the previous hardware blocks was loaded into the Flash device using the FPGA as a connection bridge between the JTAG port and the memory. Then, every one of the power islands was switched off and on again (by reprogramming the ATtiny) so that the initial bitstream was automatically downloaded into the FPGA, demonstrating that the Master BPI configuration mode worked.

In the case of the RAM memory, since the first model chosen was not supported by the software provided by Xilinx, a simple custom controller was developed to guarantee the functionality of the memory device. Even though the correct

connection of the RAM was verified, for usability reasons, the installation of a new memory model supported by Xilinx was decided. In this way, the MPMC microcontroller can be used giving the needed flexibility to the platform.

In order to check the behavior of the ADC together with the power consumption circuitry, the SPI interface block that controls the ADC was loaded into the FPGA. Then, the drop voltage measured in the shunt resistor included in the FPGA core power rail was converted by the ADC and sent via SPI to the FPGA. The value obtained was displayed using the ChipScope debugging module and compared to the values shown by the external power supply source used to power the node. The results were correct. Thus, the ADC, the power measuring circuit and the communication between the FPGA and the ADC were verified. Besides, the convenience of using the ChipScope module for the hardware module debugging was shown.

Once the proper behavior of both the external microcontroller and the FPGA were demonstrated, it was necessary to verify the SPI communication between them. Several commands were developed in order to send different commands from the microblaze embedded in the FPGA to the ATtiny processor. These commands configured the interrupts of the microcontroller, the power modes or the control of the power islands. In this way, it was the FPGA which took every decision**. This is the key aspect of all the energy management of the platform.**

Apart from the communication between the external microcontroller and the FPGA, the UART communication between the ATtiny and the ZigBee module was also checked. Besides, the use of an interrupt given by the microcontroller to wake the ZigBee module up was also proved. The UART communication between the FPGA and the ZigBee module was also verified.

II.2 Application Tests and Power Measurements

Once all these basic analysis were finished, the platform was tested under certain conditions in order to emulate a real application. **Through these tests, the convenience of using hardware based systems instead of microcontrollers in terms of energy efficiency and speed of processing is shown**.

The main setup used for these test is shown in Figure 37.

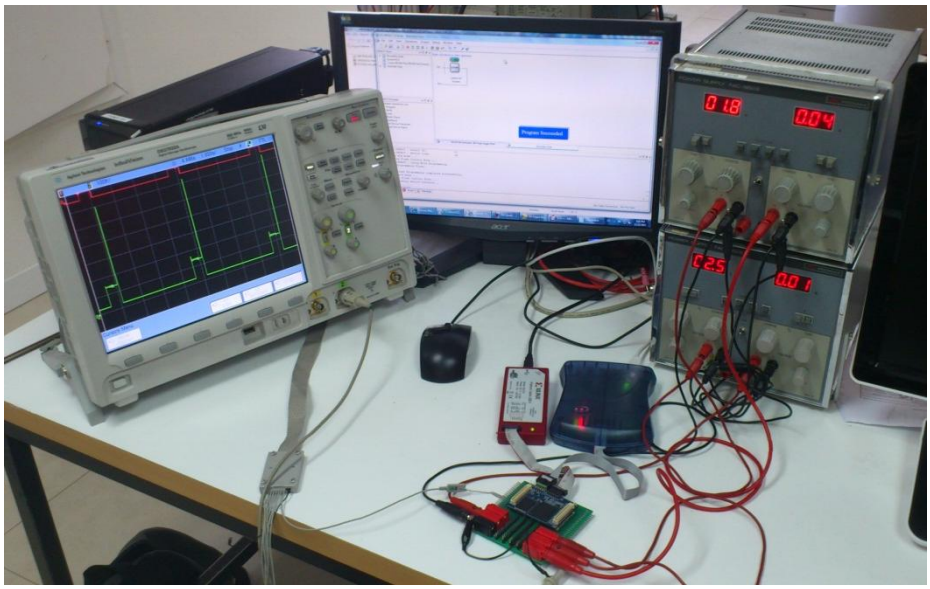


Figure 37: Tests Setup.

It is important to highlight that the energy consumption tested and compared in these analysis corresponds to the FPGA, since it is the use of the FPGA what is to be compared with the use of a microcontroller. It is understood the all the peripherals needed by the platform are equally needed either if the processor is an FPGA or if it is a microcontroller. Then, the consumption of the power supply source, the ZigBee module, the external memories, the ADC, etc. would be the same. Besides, in this platform, all the components that are not being used can be switched off feature that is not possible in most of the state-of-the-art WSN platforms.

In order to obtain the consumption profile of the application, an oscilloscope was used to monitor the current consumption of the FPGA core using the output signal of the instrumentation amplifier of the first power island. The curves shown in this chapter represent only the power consumption of the FPGA core, so both the consumption of the auxiliary logic and the IOBs must be added. The values to be added are 30 mA during configuration time and 40 mA during computing time since they remain constant even when the application changes. The consumption of the Flash memory is included within the consumption of banks 1 and 2 since they belong

to the same power island. During sleeping-time, all the islands are powered off, so the power consumption is 0 A (see chapter 4 for more details about the sleep modes) in all cases. As it will be shown on the figures, the value of the current consumption when the system is not powered has an offset. This offset appears because all the power measurements are done using an instrumentation amplifier. The offset of this amplifier is 21 mV (the value in mA is obtained by dividing this mV by 3.22 which is the gain of the instrumentation amplifier) when the current consumed by the island is 0 A. Every one of these considerations is shown in the tables included in every test.

The selected module used to run on the FPGA is the SHA1 encryption algorithm, the same shown in the second application example. This algorithm is used in real applications for secure WSNs. In this way, real needs can be tested to demonstrate the feasibility of the theoretical proposals.

The SHA1 algorithm has been implemented both as an independent hardware block and as software code running on the embedded processor. In both cases, the number of data blocks to be encrypted was configurable. In this way, it was possible to evaluate the minimum number of data blocks from which hardware starts being worthy compared to software. The time needed to configure the FPGA with the initial bitstream is crucial to minimize power consumption, since this time is huge compared with the one needed for the computational tasks. In order to show this importance, different sizes of bitstreams were also tested.

Apart from different data blocks and bitstreams, these tests also combined different wake-up policies such as the ones seen on the power management methodologies section.

These validation tests show the high flexibility of the platform. The possibility of running different algorithms in hardware and software at the same time depending on each application scenario gives the chance of optimizing task scheduling and power efficiency. **Besides, the debugging capabilities compared with previous versions of the *Cookie* node are drastically increased**. The use of the JTAG port, both for hardware and software debugging, the use of the ChipScope module, or even the UART interface, have been essential for the correct development of the applications and lab tests.

Different combinations were compared and the same algorithm was also tested in a low power version of the *Cookie* node that includes an MSP430 microcontroller.

The name of the different tests is coded as follows (1, 2, 3 and 4):

1. **HSW** if the encryption algorithm is running on the platform Microblaze (100 MHz). **HW** if the encryption algorithm is running as a hardware block on the FPGA. **SW** if it is running on the MSP430 processor, in the low power consumption version of the *Cookies*.

2. **Number** of data **blocks** to be encrypted. Each block corresponds to 16 x 32-bit row data words (512 bits).

3. **COMP** if a compressed bitstream is used or **TOT** in the case of the non-compressed one.

4. Depending on the wake-up policy used: **ZB** for the radio and **TIMER(x)** where x denotes the period in seconds.

II.2.1 Test 1 (HSW, 125, TOT, TIMER (3))

All the different steps to carry out this first test are listed below:

1. A code to power all the islands is loaded into the ATtiny processor.
2. A non-compressed bitstream (4122 kB) is loaded into the Flash memory via JTAG using the IMPACT software Tool. This bitstream includes: SPI controller, Microblaze processor and the *.elf file containing the code to be run on the Microblaze. This code includes the commands that need to be sent to the ATtiny processor to choose the wake-up policies and, in this case, it also includes the SHA1 algorithm configured to encrypt 125 data blocks.
3. Once the bitstream is loaded in memory, the ATtiny is programmed with a code that in first place powers the FPGA and Flash memory. After that, the ATtiny remains in power down mode until the FPGA finishes with the calculations and tells the microcontroller to switch it off. The FPGA has to send the message for being switched off but also the information of how it should be woken up again.

4. Every three seconds the FPGA is woken up, it encrypts the data blocks and go to sleep configuring the ATtiny to wake it up after another 3 s. The consumption profile can be seen in Figure 38.

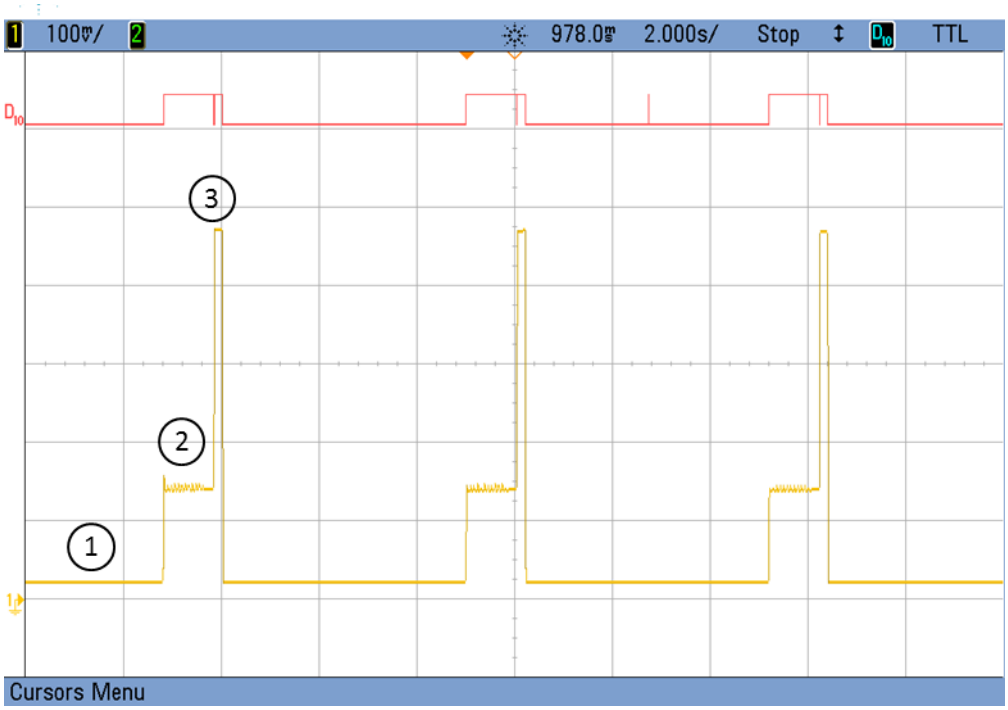


Figure 38: Consumption profile. Test 1.

As it can be seen on the figure, there are three different regions. The first one corresponds to the off-time where all the islands are switched off. The second period is the configuration time and the third one corresponds to the time needed by the Microblaze to encrypt the 125 data blocks using the SHA1 algorithm.

TEST 1	Current (mA)	Time (s)	Islands 5 and 7	Energy (mAh)
1.Off-time	0	3	0	0
2. Configuration time	50	1.02	40	0.0250
3.Computing time	140	0.16	30	0.0075
Total per cycle	190	4.18	70	**0.0325**

Table 11: Test 1. Energy Results.

This test shows how critical the configuration time is. In this case, the consumption during configuration is more than three times the value during computing time. That is the reason why reducing the size of the bitstream is a crucial methodology to reduce power consumption.

II.2.2 Test 2 (HSW, 125, COMP, TIMER (3))

Since the time required for configuration is that critical, this second test shows the reduction achieved by compressing the initial bitstream. The size of the compressed bitstream file is 2130 kB, while the total one was 4122 kB. In Figure 39 the consumption profile is shown. In Table 12, times and consumption results are shown.

TEST 2	Current (mA)	Time (s)	Islands 5 and 7	Energy (mAh)
1.Off-time	0	3	0	0
2. Configuration time	50	0.52	40	0.013
3.Computing time	140	0.16	30	0.007
Total per cycle	190	3.68	70	**0.020**

Table 12: Test 2. Energy Results.

As it can be observed, only by compressing the bitstream file, the reduction in the power consumption per cycle is reduced 1.6 times. If a partial-initial bitstream file is loaded this reduction is even bigger. This is part of the future work.

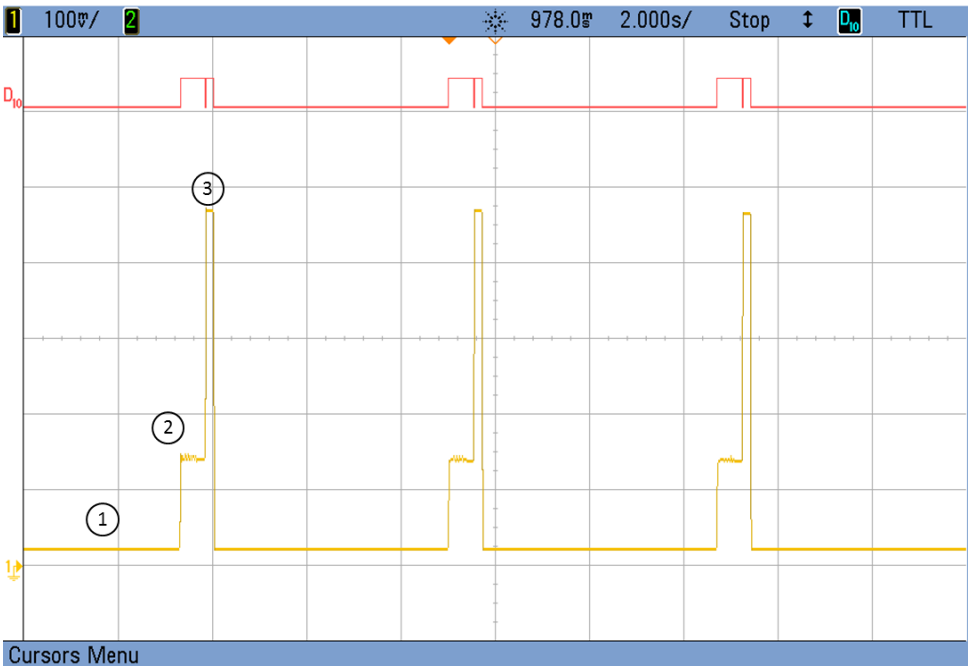


Figure 39: Consumption profile. Test 2.

II.2.3 Test 3 (HW, 125, COMP, ZB)

In this third test, the amount of data blocks to be encrypted remain constant, but the SHA1 algorithm runs as a hardware block and the system wakes up when an event on the radio occurs. Since the UART connection to the ZigBee module can be also accessed by the serial port connected to the PC, the radio signal is emulated by a command sent directly from the computer. The consumption profile can be seen in Figure 40. Besides, numerical results are shown in Table 13.

TEST 3	Current (mA)	Time (s)	Islands 5 and 7	Energy (mAh)
1.Off-time	0	2	0	0
2. Configuration time	50	0.5200	40	0.01300
3.Computing time	140	0.0071	30	0,00030
Total per cycle	190	2.5271	70	**0.01303**

Table 13: Test 3. Energy Results.

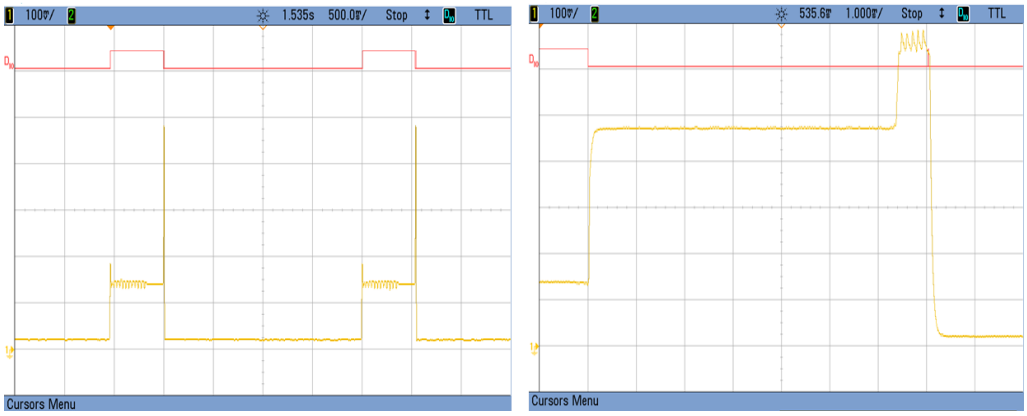


Figure 40: Consumption profile test 3.

In this case, the off-time is not given by a timer but by a radio event. In the left side of Figure 40, two work cycles are shown while on the right side; the computing time is detailed. As it can be seen, even the computing time is divided into two regions. The duration of the first one is 6.2 ms and it corresponds to the time needed by the Microblaze to activate the hardware accelerator. Once the Microblaze activates the SHA1 hardware module, it only takes it 0.9 ms to encrypt 125 data blocks. This means that, the first region can be deleted if the hardware module is loaded directly without depending on the embedded processor. The SHA1 algorithm is designed to encrypt the data in groups of 25 blocks. This can be seen in Figure 40 represented by 5 peaks in the consumption wave form. In this test, 125 data blocks are encrypted so 5 peaks appear, each one corresponding to each group of data.

It is important to highlight that even including the time needed by the software to activate the hardware module, the time required for the encryption in the case of hardware is more than 20 times shorter. The total reduction of energy consumption for the whole work cycle is 1.5 times less consumption in the case of the hardware based solution for the encryption of 125 data blocks.

II.2.4 Test 4 (HSW, 2000, COMP, TIMER (5))

In this test, the SHA1 algorithm is run again in the Microblaze. The bitstream file used is the same as in the last two tests but both the number of blocks and the wake-

up event have changed. In this case, 2000 data blocks are encrypted and the system is woken up every 5 s. As it can be seen in Table 14 and Figure 41, the time needed by the Microblaze to encrypt 2000 data blocks is very high (2.73 s) taking into account that during this time the current consumption is 170 mA. In this example, the consumption during the encryption is ten times more than during configuration.

TEST 4	Current (mA)	Time (s)	Islands 5 and 7	Energy (mAh)
1.Off-time	0	5	0	0
2. Configuration time	50	0.5200	40	0.013
3.Computing time	140	2.73	30	0.129
Total per cycle	190	8.25	70	**0.142**

Table 14: Test 4. Energy Results.

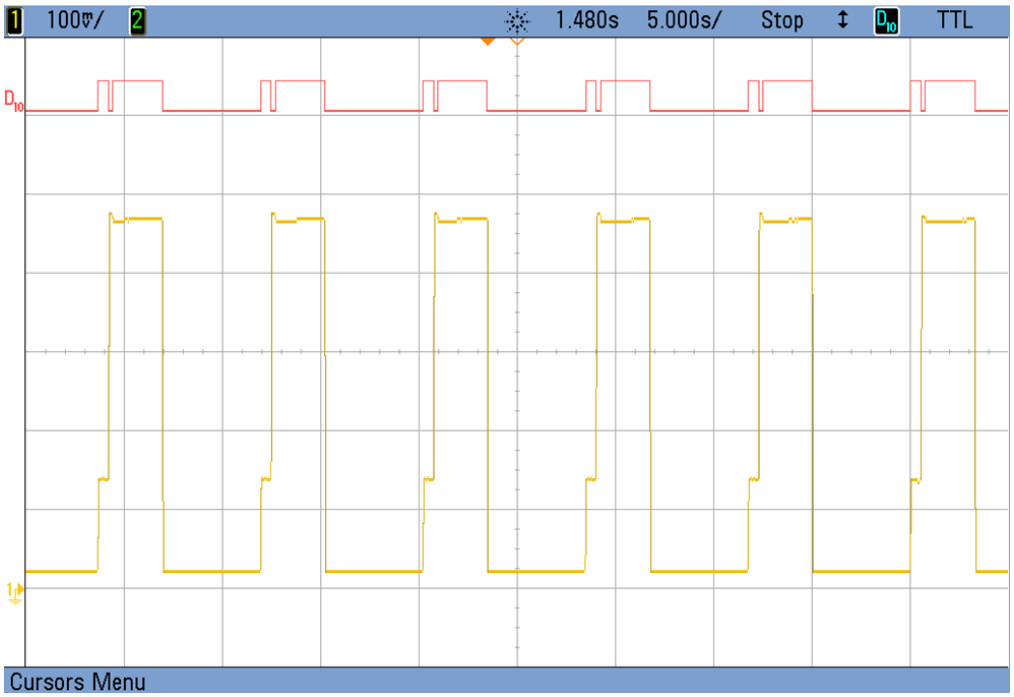


Figure 41: Configuration profile. Test 4.

II.2.5 Test 5 (HW, 2000, COMP, TIMER (2))

In order to demonstrate that the difference in terms of energy efficiency between hardware and software increases when the task to be carried out becomes more complex, the same comparison as between tests 2 and 3 but with larger amount of blocks was made. Numerical results and the consumption waveforms are shown in Table 15 and Figure 42.

TEST 5	Current (mA)	Time (s)	Islands 5 and 7	Energy (mAh)
1.Off-time	0	2	0	0
2. Configuration time	50	0.5200	40	0.013
3.Computing time	134.5	0.0057	30	0.00026
Total per cycle	190	2.5257	70	**0.01326**

Table 15: Test 5. Energy Results.

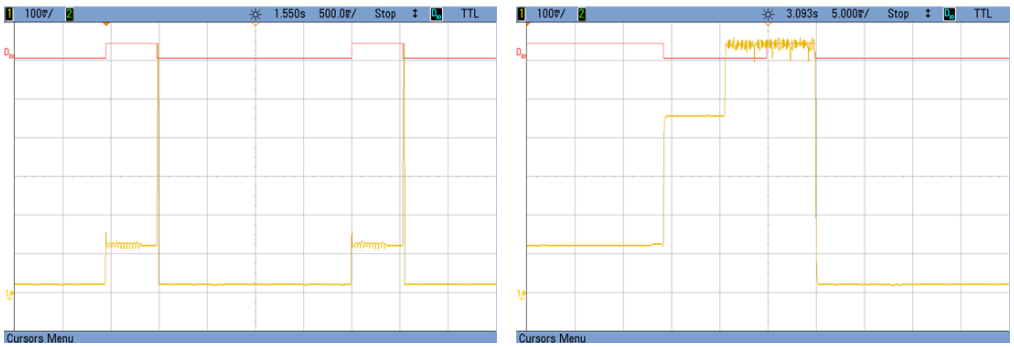


Figure 42: Consumption profile. Test 5.

Figure 42 is divided into two images. The one on the left side corresponds to two work cycles of the node, while the one on the right represents a detail view of the computing time. As it happened in test 3, computing time is also divided into two regions. The first period represents the time needed by the Microblaze to activate the hardware module while the second period is the time required by the hardware module to encrypt 2000 data blocks. As it can be noticed, the higher the amount of blocks the better the energy efficiency of the hardware base solution becomes.

In this case the relation of the energy consumption is 10 times less using the hardware base solution.

II.2.6 Test 6 (HW, 10000, COMP, TIMER (2))

The same test but for 10000 data blocks was run to prove that the difference of the energy efficiency increases.

TEST 6	Current (mA)	Time (s)	Islands 5 and 7	Energy (mAh)
1.Off-time	0	2	0	0
2. Configuration time	50	0.5200	40	0.013
3.Computing time	134.5	0.0515	30	0.0023
Total per cycle	190	2.5715	70	**0.0153**

Table 16: Test 6. Energy Results.

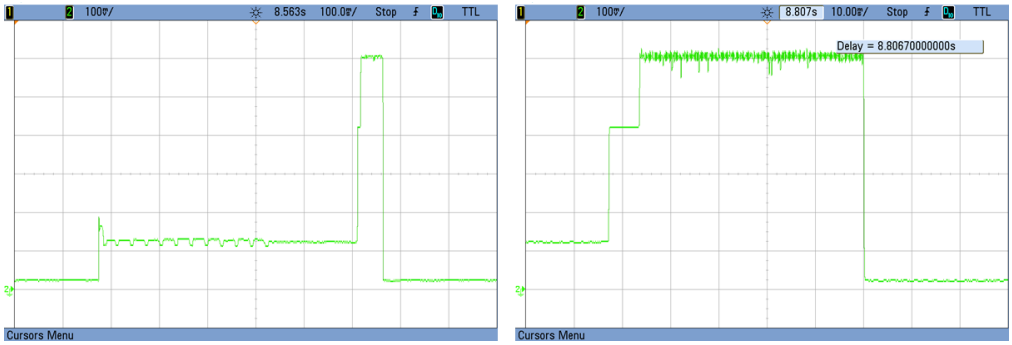


Figure 43: Consumption profile. Test 6.

The comparison between hardware and software when software is running in the HiReCookie platform is now clear. Nevertheless, software tests are still running inside the FPGA, so configuration time is also required. In addition, using an embedded Microblaze inside a high performance FPGA is a completely inefficient choice when only software is required. Thus, for the sake of correctness, the SHA1 algorithm has been also ported to another *Cookie* with a MSP430 microcontroller shown in Figure 44. This microcontroller is widely used in WSNs like the TelosB platform.

Figure 44: Low-Performance processing Cookie.

II.2.7 Test 7 (SW, 2000 and 10000, -,-)

In Figure 45, the microcontroller consumption profile is shown. The time necessary for the MSP430 to encrypt one data block is 7 ms. Time and power consumption to process 2000 and 10000 data blocks are shown in Table 17.

These tests, using a low power *Cookie*, have been carried out with the MSP430 running at 4 MHz and activating low power mode during sleep time.

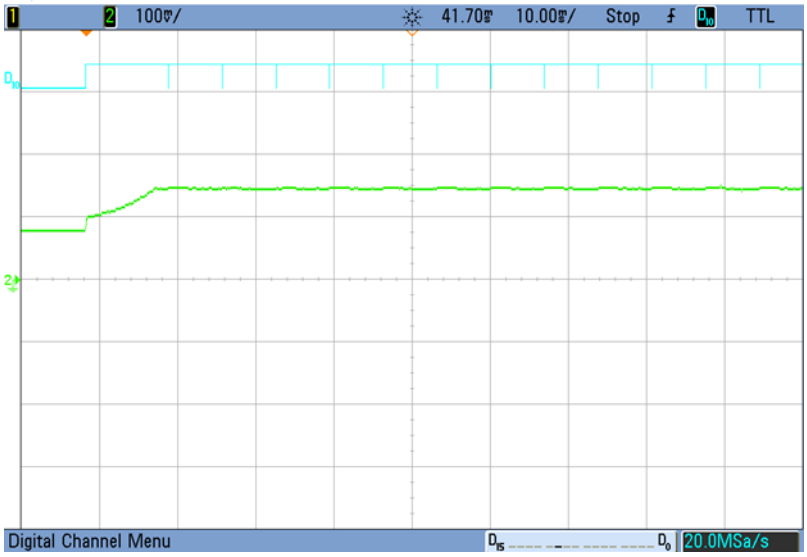


Figure 45: Consumption profile. Test 7.

TEST 7	Current (mA)	Time (s)	Energy (mAh)
1.Sleep mode	0	2	0,001
2.Wake-up time	2.4	0.01	6.7 × 10−6
3.Active mode (2000)	3	13.92	0.012
3.Active mode (10000)	3	69.6	0.058
Total value (2000)	-	-	**0.013**
Total value (10000)	-	-	**0.059**

Table 17: Test 7. Energy Results.

The consumption profile shown in Figure 45 represents only a small period of time of 100 ms. In the upper part of the image a signal acting as a trigger (in light blue) represents the time required by the microcontroller to encrypt one data block. As it can be seen in the figure, during sleep time the power consumption of the microcontroller is not 0 A as it is indicated in Table 17. This is because the sleep mode used was not the deepest one even though a deeper mode exists. However, in order to make the results more representative, the numeric comparison has been made like if the consumption during sleep time was 0 A.

As it can be seen in the numeric results between tests 6 and 7, the hardware based solution consumes almost four times less energy to carry out the same task (0.059 mAh SW based solution and 0.0153 mAh the HW based solution), while the time is reduced one hundred and twenty times (69.615 s the SW based solution and 0.5715 s the HW based one).

In Table 18, a comparison between all the different tests is shown.

TEST	Time (s)	Energy (mAh)
HSW, 125, TOT	1.18	0.03250
HSW, 125, COMP	0.68	0.02000
HW,125, COMP	0.527	0.01300
HSW, 2000, COMP	3.25	0.14200
HW, 2000, COMP	0.525	0.01326
HW, 10000, COMP	0.5715	0.01530
SW, 2000	13.93	0.01300
SW, 10000	69.61	0.05900

Table 18: Tests Comparison.

Chapter 6

Conclusions & Future Work

"Only one man ever understood me, and he didn't understand me"

G.W. Hegel

6. Conclusions & Future Work

In this Master Thesis a survey about high performance WSNs along with the opportunities offered by hardware reconfiguration has been reviewed. With the main goal of designing a WSN platform to face very demanding applications, a complete list of requirements has been identified and translated into specifications. According to these specifications, an adaptation of the *Cookie* node has been carried out. A complete redesign of both the power and processing layers has been accomplished. A new processing layer based on a high profile FPGA has been designed, developed and tested with satisfactory results. The convenience of using hardware based solutions together with low power strategies instead of the traditional microcontroller approach, has been also demonstrated in terms of energy efficiency and processing performance. This comparison only makes sense when very intensive tasks are required. The proof of concept has been done using well known encryption algorithms. **In the case of the hardware based solution** a **significant reduction of the energy consumption is observed (about 4 times less), while the processing speed is much higher (120 times).**

Of course, the improvement depends on factors such as the amount of processing required, static power consumption, duty cycle, etc. Nevertheless, as a general rule, it can be concluded that, the more processing, the more the advantages of the hardware based solutions compared to the traditional microcontroller approaches. Thus, given the tendency to include higher complexity tasks within WSNs, it is likely to happen that hardware based approaches, like the one presented here, will appear in the near future.

It is important to highlight that this platform is going to be used as the main hardware platform for two European research projects: SMART [SMART] and RUNNER [RUNNER]. Besides, this research work has been published in the Sensors Journal (Impact factor of 1.774) under the title of "Using SRAM Based FPGAs for Power-Aware High Performance Wireless Sensor Networks" [Valverde'11_2].

Within SMART project, the applications will require the node to: a) encode video, b) compress other sensor data, c) encrypt information and d) reconfigure the node according to variable needs.

Within RUNNER project, the platform will be used as a reconfigurable platform to run hardware algorithms for 3D vision such as stereo matching and 3D objects identification, embedded in an autonomous android-like robot.

Possible future work and tests to continue with this research line are listed below:

1. According to the numeric results, the time needed by the FPGA for configuration at power up, is crucial. The possibility of loading a partial-initial and compressed bitstream from the Flash memory can reduce the energy consumption drastically.
2. Partial and dynamic reconfiguration possibilities still need to be explored and exploited. Methodologies such as start-up sequences or remote configuration need to be developed to provide more flexibility and reduce power consumption. In order for the system to be capable of loading partial bitstreams dynamically, the hardware relocation module still needs to be adapted to this FPGA package.
3. More demanding applications need to be tested. For that issue, new communication layers are being designed to include communication via Ethernet or Wi-Fi. At the same time, a new sensor layer with the inclusion of two micro cameras is also being developed.
4. Some minor changes might impact and improve energy usage, such as a different division of the power islands or the lower power version of the FPGA. In this way, the FPGA with the speed grade (-2) can be replaced by the lower power version with a speed grade of (-1). According to the manufacturer datasheet, this change can save up to 40% of the power consumption while speed is reduced only in 20%.
5. Comparisons between encryption algorithms running in the embedded software, pure hardware modules or in a low profile microcontroller have

been already done. As a future work, the same algorithms should be tested in a high performance microcontroller such as the ARM cortex M3.

References

[Akyildiz'11] I.F. Akyildiz, T. Melodia, K.R. Chowdhury, "Wireless Multimedia Sensor Networks: Applications and Testbeds", *in Proceedings of the IEEE 2008, vol. 96, nº10, pp. 1588-1605.*

[Avvenuti'09] M. Avvenuti, C. Baker, J. Light, D. Tulpan, A. Vecchio, "Non-intrusive Patient Monitoring of Alzheimer's Disease Subjects Using Wireless Sensor Networks", *in Proceedings of the World Congress on Privacy, Security, Trust and the Management of e-Business, pp. 161-165, Washington DC, USA, 2009.*

[Bellis'05] S. J. Bellis, K. Delaney, B. O'Flynn, J. Barton, K. M. Razeeb, C. O'Mathuna, "Development of field programmable modular wireless sensor network nodes for ambient systems", *in Computer Communication, vol. 28, pp. 1531-1544, 2005.*

[Benbasat'05] A.Y. Benbasat, J.A. Paradiso, "A Compact Modular Wireless Sensor Platform", *in Proceedings of the 4th IEEE/ACM International Conference on Information Processing in Sensor Networks (IPSN), pp. 410–415, Los Angeles, California, USA, April. 2005.*

[Braun'09] L. Braun, J. Becke, B. Glas, O. Sander, V. Stuckert, K. D. Muller-Glaser, J. Becker, "Car-to-Car Communication

Security on Reconfigurable Hardware", *in IEEE Vehicular Technology Conference, pp. 1-5, Barcelona, Spain, Jun 2009.*

[Buratti'09] C. Buratti, A. Conti, D. Janakiram, D. Dardari, R.Verdone, "An Overview on Wireless Sensor Networks Technology and Evolution", *in Sensors Journal, vol. 9, pp. 6869-6896, 2009.*

[Cantoni'06] V. Cantoni, L. Lombardi, P. Lombardi, "Challenges for Data Mining in Distributed Sensor Networks", *in International Conference on Pattern Recognition, vol. 1, pp. 1000-1007, 2006.*

[Chalivendra'08] G. Chalivendra, R. Srinivasan, "Murthy, N.S. FPGA based re-configurable wireless sensor network protocol", *in International Conference on Electronic Design, pp. 1-4, Penang, Malaysia, 2008.*

[Chao Hu'09] Z. Chao Hu, P. Liu Yingzi, Z. Zhenxing, M.Q.H. Meng, "A novel FPGA-based wireless vision sensor node", *IEEE International Conference on Automation and Logistics, pp. 841-846, Shenyang, China, 2009.*

[Charoenpanyasak'11] S. Charoenpanyasak, W. Suntiamorntut,"The Next Generation of Sensor Node in Wireless Sensor Networks", *in Journal of Telecommunications, vol. 9, issue 2, July 2011.*

[Chen'08] W. Chen, H. Aghajan, R. Kleihorst, "Real-Time Human Posture Reconstruction in Wireless Smart Camera Networks", *in International Conference on Information Processing in Sensor Networks, pp. 321-331, St. Louis, Missouri, USA, 2008.*

[Cheng'09] W. Yung-Cheng, L. Tzu-Yun, L. Tzu-Shiang, W. Jiing-Yi, S. Jyh-Cherng, J. Joe-Air, C. Wen-Dien, T. Chien-Tsung, H. Chien-Kang, "A WSN-based wireless monitoring system for intradialytic hypotension of dialysis patients," *in IEEE Sensors Journal, 2009, pp. 1959-1962, 25-28.*

[Corke'10] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, D. Moore, "Environmental Wireless Sensor Networks", *in Proccedings of the IEEE 2010, 98, nº 11, pp. 1903-1918.*

[Culler'05] J. Polastre, R. Szewczyk, D. Culler, "Telos: enabling ultra-low power wireless research," *Fourth International Symposium on Information Processing in Sensor Networks, 2005 (IPSN), pp. 364- 369, 15, Los Angeles, California, USA, April 2005.*

[Dilmaghani'11] R.S. Dilmaghani, H. Bobarshad, M. Ghavami, S. Choobkar, C. Wolfe, "Wireless Sensor Networks for Monitoring Physiological Signals of Multiple Patients", *in IEEE Transactions on Biomedical Circuits and Systems, vol. 5, Nº 4, pp. 347-356, 2011.*

[Dutta'05] P. K. Dutta, D. Culler, "System Software Techniques for Low-Power Operation in Wireless Sensor Networks", *in Proceedings of the IEEE/ACM International conference on Computer-aided design, pp. 925-932, Washington DC, USA, 2005.*

[Fornaciari'98] W. Fornaciari, V. Piuri, "Virtual FPGAs: Some Steps Behind the Physical Barriers", *in Parallel and Distributed Processing. pp. 7-12, 1998, Orlando, Florida, USA*.

[Garcia'09] R. Garcia, A. Gordon-Ross, A.D. George, "Exploiting Partially Reconfigurable FPGAs for Situation-Based Reconfiguration in Wireless Sensor Networks", in *IEEE Symposium on Field Programmable Custom Computing Machines, pp. 243-246, Napa, California, USA, 2009.*

[Glesner'11] F. Philipp, M. Glesner, "Mechanisms and Architecture for the Dynamic Reconfiguration of an Advanced Wireless Sensor Node", *in International Conference on Field Programmable Logic and Applications, pp. 396-398, Chania, Crete, Greece, 2011.*

[Grieco'09] L.A. Grieco, G. Boggia, S. Sicari, P. Colombo, "Secure Wireless Multimedia Sensor Networks: A Survey", *in International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, pp. 194-201, Sliema, Malta, 2009.*

[Hammel'07] T. Hammel, M. Rich, "A Higher Capability Sensor Node Platform Suitable for Demanding Applications", *in Information Processing in Sensor Networks, pp. 138-147, Cambridge, Massachusetts, USA, April 2007.*

[Harjito'11] B. Harjito, S. Han, "Wireless Multimedia Sensor Networks Applications and Security Challenges", *in International Conference on Broadband, Wireless Computing, Communication and Applications, pp. 842-846, Fukuoka, Japan, November. 2010.*

[He'06] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, A. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, B. Krogh, "VigilNet: An Integrated Sensor Network

System for Energy-Efficient Surveillance", *in ACM Transactions on Sensor Networks, vol. 2, Nº 1, pp. 1-38, 2006.*

[Hübner'10] M. Hübner, J. Meyer, O. Sander, "Fast Sequential FPGA Startup based on Partial and Dynamic Reconfiguration", *in IEEE Computer Society Annual Symposium on VLSI, Lixouri, Kefalonia, Greece, pp. 190-194, July 2010.*

[Ji-gang'09] T. Ji-gang, Z. Zhen-xin, S. Qing-lin, C. Zeng-qiang, "Design of Wireless Sensor Network Node with Hyperchaos Encryption Based on FPGA", *in International Workshop on Chaos-Fractals Theories and Applications, pp. 190-194, China, 2009.*

[Krasteva'11] Y.E. Krasteva, J. Portilla, E. de la Torre, T. Riesgo, "Embedded Runtime Reconfigurable Nodes for Wireless Sensor Networks Applications", *in IEEE Sensors Journal, vol.11, no.9, pp.1800-1810, Sept. 2011.*

[Kulkarni'06] P. Kulkarni, D. Ganesan, P. Shenoy, Q. Lu, "SensEye: a Multi-tier Camera Sensor Network", *in Proceedings of the 13th annual ACM International Conference on Multimedia, pp. 229-238, New York, USA, November 2006.*

[Leligou'10] H.C. Leligou, L. Redondo, T. Zahariadis, D.R. Retamosa, P. Karkazis, I. Papaefstathiou, S. Voliotis, "Reconfiguration in Wireless Sensor NeTworks", *in Developments in E-systems Engineering (DESE), pp.59-63, 6-8, London, England, September 2010.*

[Libelium'09] http://www.libelium.com/products/waspmote

[Limberopoulos'07] D. Lymberopoulos, N. B. Priyantha, F. Zhao, "mPlatform: a Reconfigurable Architecture and Efficient Data Sharing Mechanism for Modular Sensor Nodes," *in Proceedings of the 6th IEEE/ACM International Conference on Information Processing in Sensor Networks (IPSN), pp. 128-137, Cambridge, Massachusetts, USA, April 2007.*

[Marzencki'11] M. Marzencki, P. Lin, T. Cho, J. Guo, B. Ngai, B. Kaminska, "Remote Health, Activity, and Asset Monitoring with Wireless Sensor Networks", *in IEEE International Conference on e-Health Networking Applications and Services, pp. 98-101, Columbia, MU, USA, June 2011.*

[Muralidhar'08] P. Muralidhar, P. Rao, "Reconfigurable wireless sensor network node based on Nios core", *Fourth International Conference on Wireless Communication and Sensor Networks, pp. 67-72, Allahabad, India, 2008.*

[Nachman'05] L. Nachman, R. Kling, R. Adler, J. Huang, V. Hummel, "The Intel® mote platform: a Bluetooth-based sensor network for industrial monitoring," *Fourth International Symposium on Information Processing in Sensor Networks (IPSN), pp. 437-442, 15 April 2005.*

[Nahapetian'07] A. Nahapetian, P. Lombardo, A. Acquaviva, L. Benini, M. Sarrafzadeh, "Dynamic Reconfiguration in Sensor Networks with Regenerative Energy Sources", *Proceedings of the conference on Design, automation and test, Nice, France, 2007.*

[Nedelcu'10] A.V. Nedelcu, D. Talaba, V.C. Stoianovici, M. Machedon-Pisu, I. Szekely, "Conceptual integration of wireless sensor networks with 3D virtual environments", *in IEEE International Conference on Wireless Communications, Networking and Information Security, pp. 446-450, Beijing, China, 2010.*

[Nguyen'10] N. H. Nguyen, Q. T. Tran, J. M. Leger, T. P. Vuong, "A real-time control using wireless sensor network for intelligent energy management system in buildings", *Workshop on Environmental Energy and Structural Monitoring Systems (EESMS), pp.87-92, 9-9 Sept. 2010.*

[Otero'11] A. Otero, M. Llinas, M. L. Lombardo, J. Portilla, E. de la Torre, T. Riesgo, "Cost and Energy Efficient Reconfigurable Embedded platform Using Spartan 6 FPGAs", *SPIE, pp. 806700 – 806706, Prague, Czech Republic, 2011.*

[Pinto'10] A. Pinto, Z. Zhang, X. Dong, S. Velipasalar, M.C. Vuran, M.C. Gursoy, "Energy Consumption and Latency Analysis for Wireless Multimedia Sensor Networks", *in IEEE Global Telecommunications Conference, pp.1 - 5, Miami, Florida, USA, December 2010.*

[Portilla'06] J. Portilla, A. de Castro, E. de la Torre, T. Riesgo, "A Modular Architecture for Nodes in Wireless Sensor Networks", *Journal of Universal Computer Science (JUCS), vol. 12, nº 3, pp. 328 - 339, March 2006.*

[Portilla'10_1] J. Portilla, A. Otero, E. de la Torre, "Adaptable Security in Wireless Sensor Networks by Using Reconfigurable ECC

Hardware Coprocessors", *in International Journal of Distributed Sensor Networks (IJDSN), 2010, 12 pages.*

[Portilla'10_2] J. Portilla, A. Otero, E. de la Torre, O. Stecklina, S. Peter, P. Langendorfer, "Adaptable Security in Wireless Sensor Networks by Using Reconfigurable ECC Hardware Coprocessors", *International Journal of Distributed Sensor Networks (IJDSN), 2010, 12 pages.*

[RUNNER] http://inetsis.es/noticias/2011/06/10/runner-project-reconfigurable-ultra-autonomous-robots

[Rup'09] S. Rup, R. Dash, N.K. Ray, B. Majhi, "Recent advances in distributed video coding", *in IEEE International Conference on Computer Science and Information Technology, pp. 130-135, Beijing, China, 2009.*

[Shixing'10] L. Shixing, X. Wujun, Z. Yongming, "Research and Implementation of WSN in Fire Safety Applications," *in 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM), 2010, pp.1-4, 23-25, Chengdu, China, September 2010.*

[SMART] http://www.artemis-smart.eu/home.aspx

[Smith'07] R.B Smith, "SPOTWorld and the Sun SPOT," *in 6th International Symposium on Information Processing in Sensor Networks (IPSN), pp.565-566, 25-27 April 2007.*

[Stelte'06] B. Stelte, "Toward development of high secure Sensor Network Nodes using an FPGA-based Architecture", *in*

International Wire & Cable Symposium, pp. 539-543, Istanbul, Turkey, 2010.

[Tseng'07] Y. Tseng, Y. Wang, K. Cheng, Y. Hsieh, "iMouse: An Integrated Mobile Surveillance and Wireless Sensor System", *in Computer, vol. 40, Nº 6, pp. 60-66, 2007.*

[Valverde'11_1] J. Valverde, V. Rosello, G. Mujica, J. Portilla, A. Uriarte, T. Riesgo, "Wireless Sensor Network for Environmental Monitoring: Application in a Coffee Factory", *in International Journal of Distributed Sensor Networks, 2011.*

[Valverde'11_2] J. Valverde, A. Otero, M. Lopez, J. Portilla, E. de la Torre, T. Riesgo, "Using SRAM Based FPGAs for Power-Aware High Performance Wireless Sensor Networks". *In Sensors Journal 2012, 12, 2667-2692.*

[Wu'11] D. Wu, S. Ci, H. Luo, Y. Ye, H. Wang, "Video Surveillance Over Wireless Sensor and Actuator Networks Using Active Cameras", *in IEEE Transactions on Automatic Control, vol. 56, Nº 10, pp. 2467-2472, 2011.*

[Xiao'11] H. Xiao, H. Ogai, "A distributed localized decision self-health monitoring system in WSN developed for bridge diagnoses," International Conference on Communication Software and Networks (ICCSN), pp.23-28, 27-29, May 2011.

[Yamashita'06] S. Yamshita, T. Shimura, K. Aiki, K. Ara, Y. Ogata, I. Shimokawa, T. Tanaka, H. Kuriyama, K. Shimada, K. Yano, "A 15x15 mm, 1 μA, Reliable Sensor-Net Module: Enabling Application-Specific Nodes," *in Proceedings of the 5th*

IEEE/ACM International Conference on Information Processing in Sensor Networks (IPSN'06), pp. 383–390, Nashville, Tennessee, USA, April 2006.

[Yan'11] S. Yan, L. Le, L. Hong, "Design of FPGA-Based Multimedia Node for WSN", *7th International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1-5, Wuhan, China, 2011.*

[Yong'06] W. Yong, G. Attebury, B. A Ramamurthy, "Survey of Security Issues in Wireless Sensor Networks", *in IEEE Communications Surveys & Tutorials, vol. 8, pp. 2-23, 2006.*