Cost and energy efficient reconfigurable embedded platform using Spartan-6 FPGAs

A. Otero, M.LLinás, Miguel L. Lombardo, Jorge Portilla, E. de la Torre y T. Riesgo

Universidad Politécnica de Madrid, Centre for Industrial Electronics (CEI)
C/José Gutiérrez Abascal, 2, 28006, Madrid, Spain.

ABSTRACT

Modern FPGAs with run-time reconfiguration allow the implementation of complex systems offering both the flexibility of software-based solutions combined with the performance of hardware. This combination of characteristics, together with the development of new specific methodologies, make feasible to reach new points of the system design space, and make embedded systems built on these platforms acquire more and more importance. However, the practical exploitation of this technique in fields that traditionally have relied on resource restricted embedded systems, is mainly limited by strict power consumption requirements, the cost and the high dependence of DPR techniques with the specific features of the device technology underneath.
In this work, we tackle the previously reported problems, designing a reconfigurable platform based on the low-cost and low-power consuming Spartan-6 FPGA family. The full process to develop the platform will be detailed in the paper from scratch. In addition, the implementation of the reconfiguration mechanism, including two profiles, is reported. The first profile is a low-area and low-speed reconfiguration engine based mainly on software functions running on the embedded processor, while the other one is a hardware version of the same engine, implemented in the FPGA logic. This reconfiguration hardware block has been originally designed to the Virtex-5 family, and its porting process will be also described in this work, facing the interoperability problem among different families.

Keywords: Embedded Platform, Energy Efficient, Spartan-6, dynamic partial reconfiguration, FPGA

1. INTRODUCTION

Modern Field Programmable Gate Arrays (FPGAs) with run-time reconfiguration allow the implementation of complex systems offering both the flexibility of software-based solutions combined with the performance of hardware. This exclusive combination of characteristics, together with the development of new specific methodologies oriented to the design of those dynamic and partially reconfigurable (DPR) systems, make feasible to reach new points of the system design space, and make embedded systems built on these platforms conquering more and more research importance. For instance, a promising application field of this technology is Wireless Sensor Networks (WSN). Providing WSN nodes with reconfiguration capabilities would allow the implementation of self-adaptive systems that could automatically select the optimum profile, switching between different high performance demanding modules (not implementable in SW-based systems), like specific image processing algorithms, different profiles of a video coder or variable key-size cryptographic engines.
However, the practical exploitation of this technique in fields that traditionally have relied on resource restricted embedded systems, is mainly limited by strict power consumption requirements, as well as by the cost limit imposed by its massive production. Other limitations are the highly dependence of DPR with the specific features of the technology underneath the FPGA, that is, with the particularities of each FPGA family. This increases the adaptation cost of former solutions to new devices, since the cost of keeping updated and consequently competitive by adapting to these platforms may be very high.
In this work, we tackle the previously reported problems, designing a reconfigurable platform based on the low-cost and low-power consuming Spartan-6 FPGA family, recently launched to the market. The full process to develop the platform will be detailed in the paper from scratch, beginning with the selection of the FPGA family, and following with the results of the research process required to find out some reconfiguration details of the device, necessary to design and implement the reconfiguration solution. After these steps, the implementation of the reconfiguration mechanism, including two implementation options, is detailed. The first option is a low area-consumption and low-speed

reconfiguration engine based mainly on software functions running on the embedded processor, while the other one is a hardware version of the same engine, implemented in the FPGA logic that achieves a much faster reconfiguration speed. This reconfiguration hardware block was originally designed to the Virtex-5 family [4], and its porting process will also be described in this work, facing the interoperability problem among different families. In addition, the performance of the reconfigurable platform has been evaluated, considering both the reconfiguration time and its implication in the power consumption.
The paper is structured as follows. Section 2 deals with the specific technological features of the Spartan-6 family, where energy consumption, logic internal structure and reconfiguration related details are shown. Then, a global view of the embedded platform is provided. In section 4, requirements of the reconfiguration engine are offered, while section 5 describes its internals. Section 6 goes throw the portability among different FPGAs families of the reconfiguration engine. Finally, section 7 offers some implementation results, while conclusions and future work are drawn in section 8.

2. SPARTAN-6 ARCHITECTURAL DETAILS

2.1 Selection of the reconfigurable device

Most approaches in the state-of-the art and literature exploiting dynamic and partial reconfiguration (DPR) of FPGAs are based on high performance, but also, high cost families. The main reason of the absence of works addressing lower cost models is the lower interest and effort put by manufacturers in offering support to exploit DPR for these devices, giving higher priority to top performance series. In addition, research efforts needed to successfully exploit this technique in each emerging family, limit portability of systems once they are working.

However, these devices are not suitable to be used in cost or power consumption sensitive applications that in fact are those which can take more advantage of some of the DPR benefits. Specifically, some applications like high-end Wireless Sensor Networks demand computational processing capacities which are completely exceeded by high profile devices, but that can be efficiently served by a combination of low cost devices with DPR. In those cases, DPR is useful to allow the allocation of all the processing tasks involved in the application in a single resource constrained device, increasing the logic density usage, by means of task swapping. Consequently, applications that had thought to be almost unfeasible in such kind of environments, like those based on video sensing, can become realistic. Moreover, this technique also allows coming up with strategies of managing switch-on and switch-off successive activity periods of the FPGA, in order to reduce long-term power consumption. In addition to this low-power consumption requirements, also small size, and consequently, a small footprint package suited to be integrated in wireless sensor "motes", as well as a low cost to be massive produced, regarding applications which can have hundreds or thousands units, have been set as requirements of these kinds of applications.

According to these requirements, restricting device search to those FPGAs which offer DPR, the result is that modern 65-nm and 45-nm reconfigurable devices by Xilinx have been selected because they offer a balanced trade-off between power consumption and performance. Among them, Virtex-5, Virtex-6 and Spartan-6 alternatives have been considered. Regarding power consumption, different simulations with XPE (XPower Estimator) of Xilinx have been carried out comparing the same processing engine in different units of each family. In this case, an ECC accelerator provided by IHP has been chosen, since its integration in a WSN platform has been already proved in [5], and it is a very interesting acceleration engine to be included in secure networks. Simulation results are only shown for those parts where the crypto-engine can be configured, without occupying more than 20% of the FPGA resources, according to an estimation of total required complexity. As shown in Figure 1, the study reveals that the Spartan-6 family has near 10 times reduced static power, while dynamic consumption is slightly under the Virtex-5 value. Dynamic power consumption is shown in Figure 2. This analysis yields the Spartan-6 family to be the best candidate for this reconfigurable platform. Spartan-6 FPGAs have also enough logic resources to implement complex algorithms, since the biggest one has about 6 million equivalent gates. Furthermore, low power consumption versions are also available at the market.

Once the device has been selected, next step was to deal with reconfiguration issues, since at the moment those devices were selected, no state-of-the art work was available reporting feasibility of exploiting this feature. As far as authors know, only [3] reports a successful reconfiguration on Spartan-6, but instead of relying on a general reconfiguration engine, as is the case of this work, it is focused on offering a novel design flow.

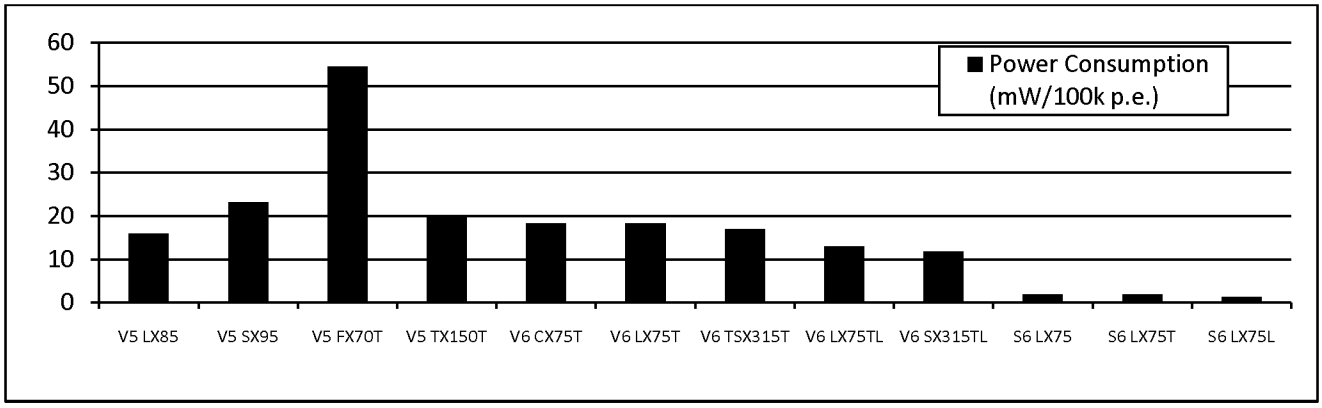

Figure 1. Static Power consumption per 100k equivalent gates

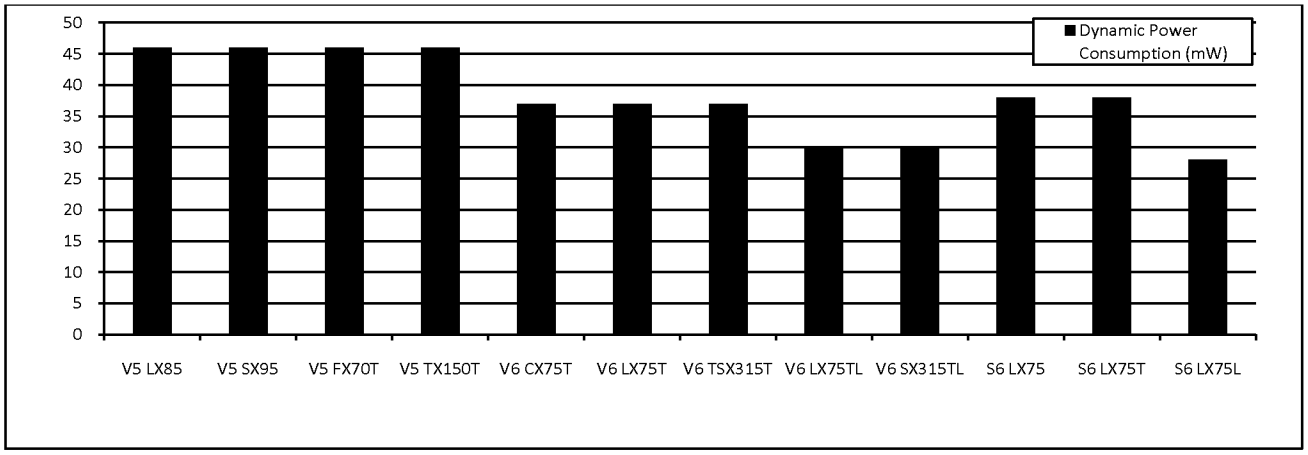

Figure 2. Dynamic power consumption of different Xilinx FPGAs families

2.2 Reconfigurable Fabric Description

The Internal architecture of the Spartan-6 does not differ too much, regarding its main concepts, to other recent Xilinx devices. As it is shown in Figure 3, it is based on Configurable Logic Blocks (CLBs) arranged like a matrix, with specific purpose DSP and BRAM columns embedded in between. Each CLB has an interconnection matrix and two slices, where associated logic functions are implemented in 6-input Look-Up-Tables (LUTs) and registers. There are three different slice types (X, L and M), depending on what functions they implement, such that Slices-L incorporate carry logic and larger multiplexers than the basic Slices-X, while Slices-M have distributed RAM and shift registers inside. As is shown in Figure 3, this matrix is organized in clock regions, which share the same clock signals. Each clock region spans a width of 16 CLB blocks. The full reconfigurable logic can be seen like a set of attached clock regions, where the number of clock regions depends on each specific part of the Spartan-6 family. In addition, different devices of the family differ on the inclusion of specific purpose *hard* blocks for High-Speed serial connectivity (LXT versus LX family). A summary of the different Spartan-6 parts is shown in [2].

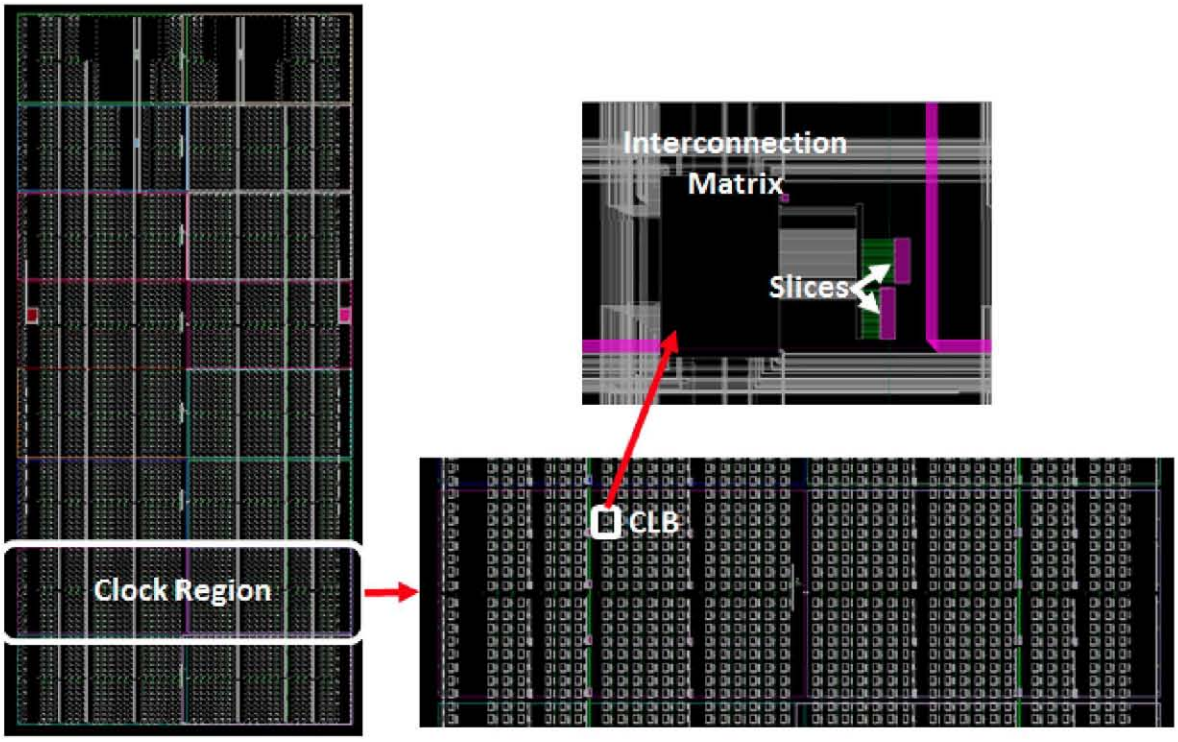

Figure 3. Spartan-6 FPGA architecture

2.3 Reconfiguration details of Spartan-6 devices

The functionality of each configurable element within the FPGA is controlled by information stored in the configuration memory. The smallest information packet that can be read from or written to this memory is called a *frame*. In Spartan-6 it is composed by 65 16-bit words, which include configuration of a column of configurable blocks (DSPs, BRAMs or CLBs), spanning a clock region of the device. Consequently, 2D reconfiguration is possible, since configurable elements allocated in two different clock regions are configured by different frames. That is, the height of the device does not have to be configured entirely in each operation. The change of the configuration content stored in this memory is done by means of read and writes operations of the internal configuration registers. There are different interfaces that allow access to the Spartan-6 configuration registers, like JTAG, SPI or Select MAP, in order to modify at run-time the behaviour of the configurable logic. However, only ICAP (Internal Configuration Access Port) can be accessed from within the FPGA reconfigurable logic itself. This port enables Self-reconfiguration concept, that is, a modification of the logic of the device carried out directly from the same internal FPGA logic, without requiring an external device. The Spartan-6 ICAP interface consists of two 16-bit data buses (both input and output), a clock input and three control signals.

Partial reconfiguration implies to provide some configuration commands, the register programming for those registers in the area being reconfigured, and the bitstream data itself, as can be seen in configuration user guides like [7] or [8]. Main registers of interest for this purpose are the FAR, where the address of the frame to be accessed has to be stored; the CRC, a checksum of the configuration data; the COR, where the configuration options are selected; the FDRI, where the FAR data to be configured is stored, and the FDRO, that provides readback configuration information. In addition, a command register (CMD) is used to trigger the internal state machine of the ICAP.

A broad outline of the reconfiguration process is as follows. First, some configuration commands in order to synchronize the internal configuration logic, as well as to configure some parameters of the process, are sent. This information is called bitstream header. Then, an iterative process including the frame address where to store the configuration data, together with the configuration data itself, are downloaded to the device. This is the bitstream body. Finally, a bitstream tail is sent to the device, in order to finish the process, as well as to assure its correctness, by means of a CRC check.

During a full configuration process, all the FPGA is configured, and consequently, thanks to the self-increment mechanism of the FPGA, only the first frame address has to be sent. However, for carrying out a partial reconfiguration, a suitable sequence of FAR addresses corresponding to the reconfigurable region has to be generated. Consequently, it's very important to know the relationship between configurable elements and its address in the configuration memory. However, FPGA user guides do not provide this information so far. However, it can be observed that there are similarities between the major and minor addressing schemes reported for previous families, like the Virtex ones, and the Spartan-6 addressing scheme. So, the address register (FAR), as well as in Virtex.-5 contains a field to address different clock regions, another field to specify the CLB row (major address) and another one to specify the number of specific frames within a CLB row (minor address) to fully decode the address of regular logic elements within the FPGA architecture.

3. SPARTAN-6 RECONFIGURABLE EMBEDDED PLATFORM

Once the reconfigurable device was chosen and internal details both considering the structure of the FPGA as well as reconfiguration related aspects were clarified by using the *Spartan-6 FPGA Embedded Kit* development board (based on the Spartan-6 LX45T FPGA) [6], the custom reconfiguration-oriented platform was designed. Main components of the platform are shown in Figure 4, including the division between static and dynamic areas, as well as the different peripherals within the static section. Regarding these peripherals, several communication interfaces dedicated to external memory accesses, as well an embedded µBlaze processor, and different low speed communication interfaces addressing external sensors have been included.

Restricting our description to DPR issues, main internal components of the FPGA are the reconfiguration engine, in charge of controlling the reconfiguration process, as well as the virtual architecture, that splits resources among reconfigurable and static regions, and solves the communication issues among these modules. In this work, we will focus on the design of the reconfiguration engine, since it is very dependent on specific low level details of each FPGA family, and no solution addressing this problem has been found in the literature, regarding Spartan-6 FPGAs.

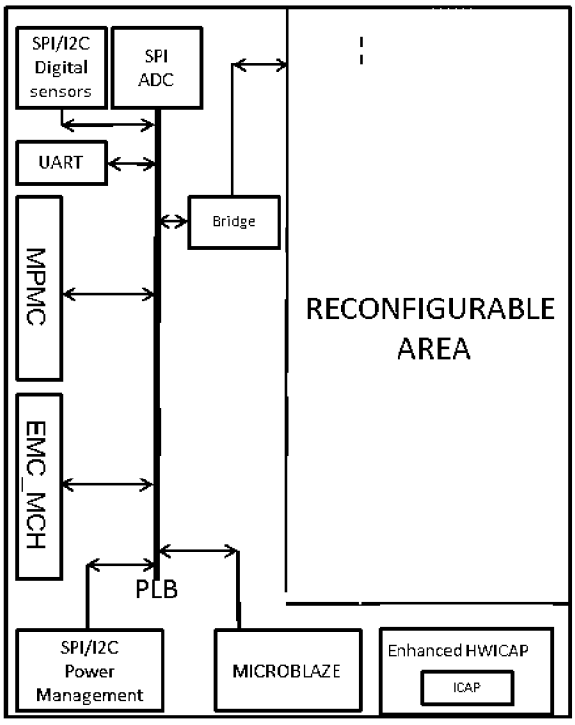

Figure 4. Spartan-6 FPGA architecture

In this platform, reconfiguration is managed by a processor embedded in the FPGA logic. This processor will decide what and when to reconfigure each reconfigurable region of the virtual architecture designed on the device, and a main

requirement of the designed reconfiguration engine is to hide low level reconfiguration details to the applications designer. The followed approach was to design a set of API functions, thus simplifying the configuration of each region to calls to these functions, including the name of the file where these bitstreams are stored in the external memory, as well as four coordinates inside the device and that characterize reconfigurable region entirely. Partial Bitstreams are stored in external CompactFlash memory using Xilinx System ACE [9] and from this memory, they can be read when required. To obtain this configuration information, also readback functions able to read from the configuration memory of the device and send this data to the external memory have been developed. In the following section, information about both requirements and implementation details of the reconfigurable engine will be shown.

4. RECONFIGURATION ENGINE REQUIREMENTS

Hardware support offered by Xilinx in order to carry out dynamic and partial reconfiguration of its SRAM based FPGAs is based on providing an specific IP Core called HWICAP, that acts like a wrapper of the ICAP [1]. This core integrates the ICAP reconfiguration port together with a finite state machine in charge of generating of the necessary control signals to assure a correct data transference from some input FIFOs, also included in the core, to the reconfiguration port. This core has the structure of a peripheral, so that it can be integrated together with other peripherals and an embedded processor within a reconfigurable System on a Programmable Chip (SoPC). Two versions are offered, one using IBM CoreConnect bus technology [10], and the other ARM AXI one [11]. However, guaranteeing the correctness of configuration data available inside these input FIFOs is on the side of the designer, that is, the application designer is in charge of sending correct configuration commands to this port in order to carry out a successful reconfiguration. To control the operation of the HWICAP peripheral from the embedded processor, some drivers are also available. However, functionality included on functions within drivers are restricted to read and write single frames from/to the ICAP. Regarding one frame reconfiguration, configuration data itself is integrated with configuration commands to synchronize and desynchronize the port, destination address within the reconfiguration memory or the CRC value of the package, among others. Consequently, this approach incurs into an extremely high overhead, considering the amount of information that has to be transferred through the ICAP in order to change a single configuration frame.

However, applications running on top of platforms exploiting DPR features of FPGAs, typically require more complex functionalities to be included in the underneath reconfiguration engine, than changing single configuration frames. Typically, this technique is used to swap hardware logic cores running at run-time in the reconfigurable logic, to adapt these cores to run-time changing conditions, to optimize certain parameters of these reconfigurable cores, as well as to delete unused cores at certain moment, reducing application power consumption.

Therefore, a mechanism to reconfigure full regions of the device, as well as to relocate reconfigurable modules from one position to a different one, or even to copy and paste the same module in multiple positions of the FPGA is required in most system implementations. Since HWICAP does not offer coverage to fulfill these requirements, enhanced reconfiguration engines have been addressed in the state of the art. Restricting this summary to on-chip manipulation solutions, allowing autonomous reconfigurable systems, a further classification can be based on the processing approach, having both embedded software and hardware solutions. Examples of software-based solutions are XPART, a C version derived from the set of JBits Java classes and pBITPOS [14], an embedded version of BITPOS. Regarding hardware solutions, a remarkable approach is the REPLICA (Relocation per online Configuration Alteration) filter proposed in [15]. This solution is based on the concept of the bitstream filtering during relocation, in order to reduce the process overhead. Basically, the idea is to implement a finite state machine that parses the bitstream during the configuration, identifying the different fields and commands. Among these fields, it is necessary to detect the FAR, in order to replace the original addresses where the block was synthesized, with the final addresses where it will be relocated, as well as the CRC. The generation of the new addresses where each frame has to be relocated is the heart of the filter. A new version including support for Virtex-2/ Pro devices, called REPLICA2Pro, was introduced in [12]. The BIRF (Bitstream Relocator Filter), presented in [16], is similar to REPLICA in the sense that it is also based on a filter approach, and is restricted to 1D relocation in Virtex FPGAs. However, BIRF claims to have a better performance, but no performance information was included in REPLICA. The main difference is a simplification of the Parser FSM, limiting its behavior to detect the FAR and the CRC. An enhanced version of this proposal, reported in [13], solves the bidimensional relocation problem, addressing Virtex-4 and Virtex-5 devices.

A different approach is the Self-reconfiguring Platform (SRP) by Blodget et al. in [17] and [18], which includes the idea of attaching the hardware reconfiguration subsystem as a peripheral of the embedded MicroBlaze processor, using the CoreConnect Bus technology. Inside the hardware subsystem, a cache is included, with capacity to store a single

configuration frame. The software component of the platform defines a low-level API, in charge of the ICAP-cache memory control, and a high level API programmed in C language called XPART, derived from JBits. As a result, it contains methods to random access bitstreams, as well as to relocate partial bitstreams. This software component contains the read/modify/write operation for configuration data, which enables fine-grain hardware modification, like changing LUT equations, as well as copying a rectangular region of configuration memory and pasting it in another position. This can be very useful to some applications, avoiding the access to the external memories The PRR-PRR reconfiguration approach introduced in [19], provides frame relocating from a partial reconfigurable region (PRR) in the device, into another region. This is done in a frame by frame basis, in order to accelerate the relocation process and to reduce the bitstream storing requirements. However, this approach has a huge overhead of repeating the different sequences of the header of each pair of frames, having no possibility of reconfiguring blocks from the external memory. Furthermore, the algorithm to relocate a full rectangular region is implemented in software. Consequently, these features have been included in the portable reconfiguration engine described in this work.

5. SPARTAN-6 RECONFIGURATION ENGINE

Reconfiguration needs of the platform depend on the considered use case of the architecture. In some cases, it will be only required at the initial stages, for system debugging or, in the case of WSNs, for deployment purposes. On the other side, it can be part of the system, in charge of carrying out task swapping or adaptation on the fly, with strict run-time requirements. Consequently, two implementation or profiles of the reconfiguration engine have been created. The first one is a mainly software solution, while the second one, is a hardware based one. Hardware relocation engine is able to transmit configuration data to ICAP at maximum ICAP admission rate, that is, 20 MHz in the case of Spartan-6 devices. In addition, relocation does not introduce any overhead, since it is performed on the fly, during bitstream download process. Also module relocation is performed at maximum ICAP rate, since bitstream is incorporated on the fly. Differently, relocation using software based approach has to be executed sequentially on the embedded processor, introducing a time overhead. In addition, configuration data has to be send also from the processor, after have read it from an external memory. This approach constitutes a bottleneck, increases system bus occupancy and reduces the global performance of DPR.
Reconfigurable embedded platform described in this paper can benefit from both approaches, selecting, for each application, the more suitable one. Area and performance can be balanced, choosing the correct profile in each case.

5.1 Software Reconfiguration Engine

Software version of the reconfiguration engine is build on the top of both, drivers and hardware design (HWICAP) offered by Xilinx. It consists on a function in charge of the generation of the sequence of frame addresses expanding the desired area to be reconfigured, which are combined iteratively with configuration information read from the external Compact Flash. Addressing and configuration information is combined using reconfiguration functions provided by Xilinx. However, these functions perform this operation on each single frame, that afterwards, are sent to the HWICAP to be transmitted through the ICAP configuration port. Hence, this mechanism introduces a high overhead in different ways:

- *Address sequence generation in software.* Addresses have to be incremented accordingly to the FPGA addressing scheme and its internal structure, and a checking process has been incremented to see whether the bound of the region has been reached or not.
- *Frame by frame nature of Xilinx reconfiguration functions. S*ending configuration data in a frame by frame basis to the ICAP, introduces an extra overhead, since each time a full cycle including a new synchronization process, all the configuration commands and configuring all the options, the introduction of configuration data, as well as a compulsory pad frame, have to be included.
- *PLB bus limitations.* Data transmission from the embedded processor to the HWICAP, as well as its inherent overhead reduces dramatically HWICAP performance.

User API offered to carry out DPR using this software profile is similar to hardware based one, thus portability is achieved almost immediately, among both profiles.

5.2 Hardware Reconfiguration Engine

The solution for bitstream relocation proposed in this paper is a version of a former one proposed in [4], ported to Spartan-6 devices. It can be considered like a hardware-based one, with enhanced functionality. As a result, this platform offers fast configuration, with features offered by software solutions, but still with reduced area overhead. Full compatibility with software based one described above is achieved. They can be easily exchanged within an application, since both rely on the common API. Hardware system is integrated as a peripheral structure, to be attached to the microprocessor embedded in the FPGA (hard or soft core), making it compatible with the *CoreConnect Technology*. In addition, its file structure fits perfectly with Xilinx's EDK peripheral cores, so a user with no expertise in dynamic reconfiguration, can just add it to an EDK project the same way any other core is added.
In addition of bitstream relocation, among the hardware implemented options, bitstream readback, replication and relocation of full configurable columns have been implemented. This approach also allows performing fine-grain modifications during the relocation process. As a result, this peripheral can be seen like a final solution to easily provide dynamic reconfiguration capabilities developed so far. Another characteristic that can be very useful to certain applications is the copy and paste property, from a position in the internal memory, to a different one. This can be very useful in applications like the scalable coprocessors shown in [20].
An important structural difference, compared with other approaches, is that this solution, instead of relocating partial bitstreams, headers and tails of the bitstreams are self-generated internally, and only pure configuration data is given as input. As a result, in addition of making easier its porting to new families, as shown in this work, the size of the configuration data of the core is reduced; on the other side, the relocation process is easier, since no parser has to be implemented. Furthermore, to allow a faster portability among FPGAs families and devices, as well as to simplify the generation of the ICAP control signals, the Xilinx HWICAP is included as a subcomponent. HWICAP already includes the configuration control machine, hiding lower level details. In addition, hardware system includes some blocks in charge of control issues, the generation of the FAR, the generation of the header and the tail, as well as the control of the ICAP. In the following subsections, further details of the hardware implementation are provided. A general structure of the block is shown in Figure 5.
Main subcomponents of the system are:

- *Input/Output Registers*: These registers receive configuration commands from the embedded processor. In addition, coordinates of the area to be configured, as well as hardware control instructions are also received. Data transmissions from/to the processor to the block are also carried out by means of these registers. They are mapped in the addressing space of the processor.
- *Input/Output Memories*: These memories store temporally configuration data to be introduced through the ICAP, or readback data from inside the the configuration memory. It is used as a buffer to decouple different clock frequencies of the ICAP and the PLB bus, as well as to store readback data to be realocated in different positions of the FPGA.
- *Control State Machines:* A specific state machine has been included in order to generate the sequence of addresses to cover all the reconfigurable region. A different one has been included to control the HWICAP operations, and a third one in charge of the global control of all the operation has been implemented. Design strategy has been to separate all the functionalities with a high modularity, so as to make easier portability among different FPGAs families.

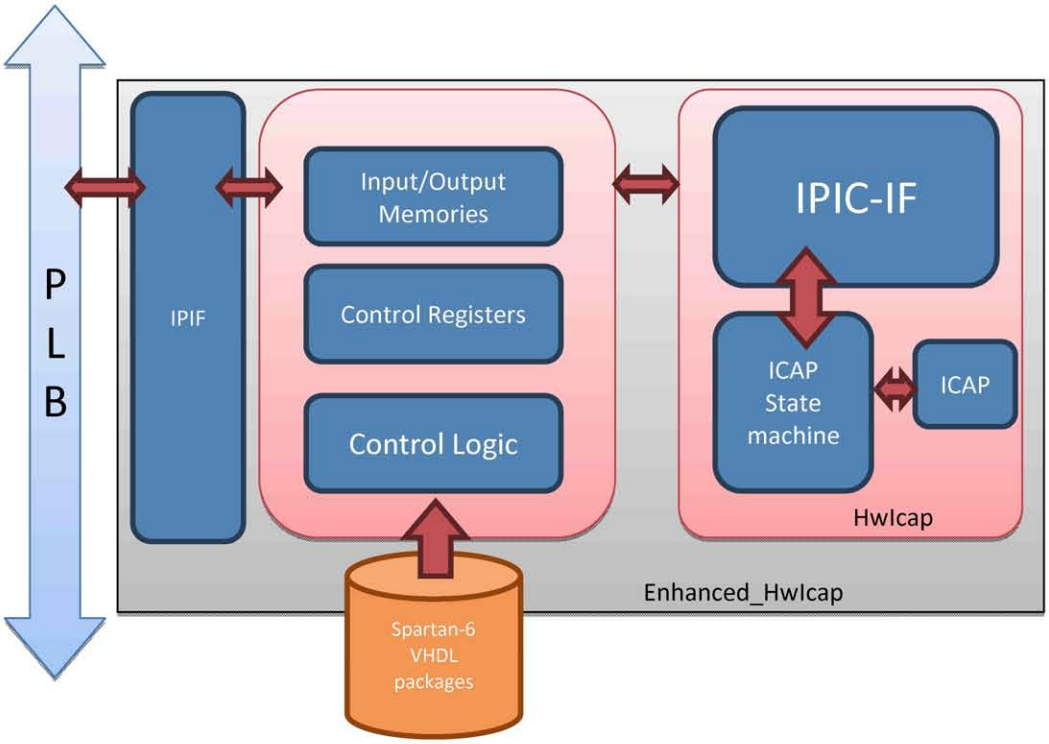

Figure 5. Enhanced HWICAP internal structure

6. FAMILY AND DEVICE PORTABILITY OF RECONFIGURATION ENGINES

Most State-of-the art works related with dynamic and partial reconfiguration have been carried out in academic environments. Problems burdening its industrial applicability are mainly related with the lack of widely extended and complete design flows and supporting tools, as well as interoperability problems among reconfigurable devices. DPR is a technique very dependent on low level details of the device, including its specific architecture, some parameters related with configuration file format, as well as control signals of the configuration port. Consequently, implementation of DPR solutions based on new devices is typically delayed certain time after the delivery of these new devices. In addition, manufacturers do not provide low level details at the early moment. To overcome these problems, as well as to reduce the cost of adopting new configuration solutions in new devices regarding work effort, the proposed reconfiguration engine has been designed so as to be easily ported into new emerging devices. The porting strategy relies on two main characteristics : i) to include *vhdl* packages that define FPGA dependent parameters, and ii) reusing Xilinx control engine (HWICAP) without reallocation possibilities, to avoid customizing low level configuration details. In the next subsections, we will evaluate both design strategies.

6.1 VHDL Packages

Regarding the parameterization of the reconfiguration engine, the main differences among FPGA families include the number and distribution of elements within the reconfigurable fabric, as well as the structure of the bitstream. Anyway, the reconfigurable fabric can be modeled as an array of reconfigurable elements. Different reconfigurable elements can be characterized by the number of configuration frames, also dependent on the FPGA family. This information is needed to generate the sequence of addresses to span the full region. For instance, in Spartan-6, possible reconfigurable elements are CLBs, DSPs, BRAMs or IOs [7]. Hence, distributions of these configuration elements inside the array, as well as the number of frames per each configurable element type have been parameterized in a family-specific VHDL package. So, different packages are available for Virtex-5 and Spartan-6 FPGAs. These packages are used like the entry point to the FSM to cover the reconfigurable region that has to be read or write, so as frame addresses corresponding to the entire region can be generated using this information. The strategy is as follows. FSM remains without changes for all the FPGAs families, while only information included in the *package* has to be changed.

On the other hand, differences regarding the structure of the configuration bitstream have to be considered also., All Xilinx FPGA families have the same parts in the bitstream, since they are divided into header, body and tail. *Header* includes a sequence of configuration commands, like some synchronization words, instructions to select the options during the reconfiguration process or CRC checks; *Body* is an iterative sequence of reads or writes commands, as well as frame address and configuration information itself; *Tail* includes other commands like CRC checks or desynchronization words. To make easier portability overcoming differences among bitstream structures, differently to state-of-the art works, the strategy has been to compound bitstream at run-time, instead of parsing it, to identify the fields that have to be changed. Accordingly, information regarding the different commands, as well as the sequence corresponding to the header and tail, both for reading or writing operations, are included in another VHDL package. In this manner, sending configuration information to the ICAP is reduced to an iterative process, incorporating information included in both VHDL packages. So, header and tail data are provided automatically from the logic itself, once these packages are read during the synthesis processes. overcoming the lack of tools for generating partial bitstreams.
These packages are the basis to adapt this block to new families and devices. The main problem about this is the lack of information from the manufacturer which is not included it in the user guides. The structure of the FPGA, can be inferred from the layout graphic tools, like *FPGA Editor* or *PlanAhead* tools. These tools show the distribution of the different resources in the device. However, the number of configuration frames, as well as the number of configuration words for each frame, have to be inferred from user guides, or by the similarities with previous published bitstream formats.

6.2 Configuration Port Control

Low level control of the ICAP, that is, timing issues and communication protocols, are also dependent on device specific features. Consequently, instead of implementing ad-hoc control in the designed logic, the vendor's full HWICAP has been included within the enhanced reconfiguration engine. Thus, to upgrade the support for different device families, this IP can be changed completely, exploiting manufacturer support. This strategy avoids having to customize low level control details of the port when porting it to new devices. For instance, Virtex-5 configuration engine relies on HWICAP version 4, while Spartan-6 reconfiguration engine uses version 6. The main enhancement between both versions is the instantiation of the Spartan-6 ICAP, as well as some minor differences in its control state machine. Regarding the reconfiguration engine side, any modification of the control FSM has been necessary to support new HWICAP version. Compatibility is assured, since, a standard IPIF [10] interface based on CoreConnect bus technology is used in all available HWICAP versions.
Furthermore, other important difference between Virtex-5 and Spartan-6 ICAP ports is its bus-width. Virtex-5 data input/output configuration port is 32-bits, while in the case of Spartan-6 low-cost devices is 16 Bits. However, this fact, that might be a major problem when porting it among both families, can be easily overcome by means of the HWICAP reutilization strategy, described in this section. This module includes a generic value that directly accomplishes the necessary adaptation, by discarding unused bits. Hence, adapting the reconfiguration engine to different configuration word sizes can be straightly carried out by means of the modification of the data width included in the VHDL packages, by appending a sequence of zeros to the actual configuration words.

7. IMPLEMENTATION RESULTS

In this section, results related both with resource occupancy and performance of the designed reconfiguration engine are offered. In addition, comparisons are carried out, considering software versus hardware solutions, as well as implementations using Virtex-5 and Spartan-6 devices.
Synthesis values are compared with similar approaches. The first analysis is offered in terms of resource occupation, as Table 1 shows. The values of this table have been selected trying to compare the results of each work in similar conditions or operation profiles. Consequently, the REPLICA2Pro results correspond to a profile without CRC checking, while data corresponding to BIRF2D area have been estimated from relative percentage results. HWICAP occupation is removed in all cases, and shown in the last row for relative comparison.

Table 1. Resource occupation results

Approach	Slices	Memory Requirements
Proposal for v5 (Without HWICAP)	296	4 blocks of 36 Kbits
Proposal for sp6	309	10 blocks of 18 Kbits

(Without HWICAP)		
BIRF 2D	80	No internal memory
REPLICA2Pro	322	No internal memory
HWICAP	594	No Block RAM is used

According to this table, values obtained for the proposed module are comparable with Replica2Pro, in spite of the implementation of the new functionalities in hardware, and the portable generic design approach. The introduction of the enhanced control capabilities does not imply a significant area overhead, since the total used slices are, for instance, 2% of the total resources of a Virtex-5 FX70T, which is in the small-medium range of the family. In this version, four 36 Kbits memory blocks have been dedicated to store partial *bitstreams*, but this value can be adapted to different applications. In the case of the LX45 Spartan-6 device, reconfiguration engine occupancy constitutes a 4%. Furthermore, since Xilinx HWICAP occupies a maximum of 460 Slices in a SPARTAN-6, resource consumption overhead of the proposed hardware engine versus software based one, which relies only on HWICAP, is a 67%.

Regarding the maximum operation frequency, the proposed solution implemented in Virtex-5 can operate at up to 208 MHz. The reported frequencies of the reference works in similar conditions, that is, using the synthesis and simulation information, are 160 MHz for BIRF (also on Virtex-5) and 35 MHz for Replica2D (on Virtex-4). In spite of these values, the running frequency when control blocks are integrated in the system is limited also by the maximum operation rate of the ICAP. Xilinx guarantees a maximum frequency of 100MHz. Consequently, at this speed, the addition of relocation features does not include an extra overhead. Regarding Spartan-6 reconfiguration engine, maximum speed achieved are 137 MHz. As in the case of Virtex-5, it does not constitute and extra overhead, since maximum working speed of ICAP port are 20 MHz.

The behavior of the block can be seen in the ChipsCope Pro Analyzer captures shown in Figures 6 and 7, corresponding to a reading and a writing operation from/to the configuration memory, respectively.

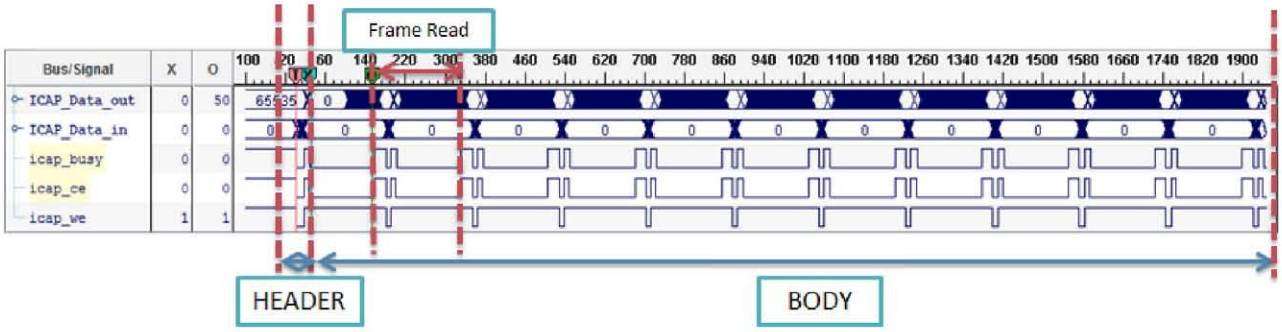

Figure 6. Read operation from the configuration memory

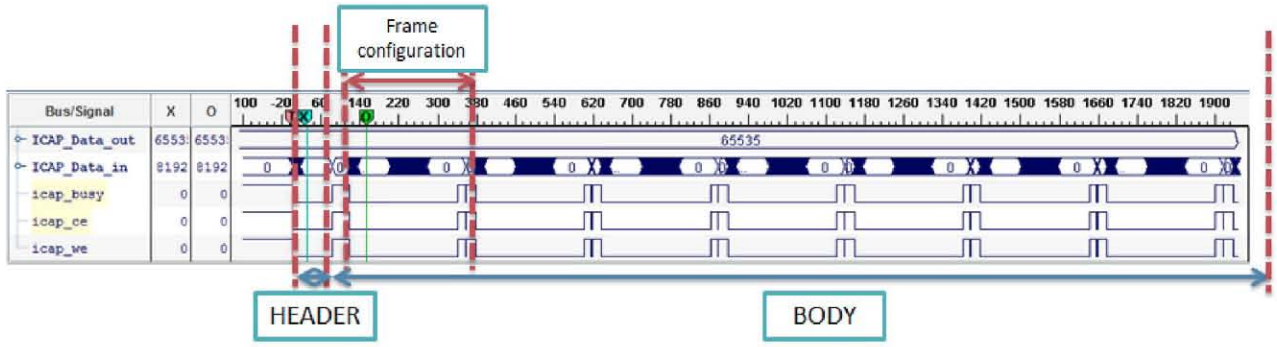

Figure 7. Writeback operation in the configuration memory

In Figure 6, a reading operation is shown. It includes the bitstream header sending, generated using the information coming from the VHDL packages, followed by a sequence of frame readings. Data read within each operation includes configuration information as well as a previous pad frame. Before each new frame reading, corresponding commands are sent to the ICAP. In Figure 7 a writing operation is shown. During each frame writing, commands, configuration information as well as pad frames are sent to the ICAP.

8. CONCLUSION AND FUTURE WORK

In this paper we provide the design of an Embedded Platform built using a low-cost reconfigurable device, but offering a similar performance compared with a high-end one. To come up with the platform, all the steps from the scratch have been described. These steps include the selection of the device, a Xilinx Spartan-6 FPGA, as well as the implementation of a reconfiguration engine within the platform. Regarding this engine, it has been directly adapted from a design previously provided, by the same authors, for Virtex-5 FPGAs. This adaptation has been carried out easily, thanks to the modular and parameterized nature of the proposed reconfiguration engine.

Future Work includes the implementation of an NPI (Native Port Interface) controller to provide the Enhanced Reconfiguration Engine with direct access to an external RAM memory, in order to reduce data transference overhead. In addition, a study of the costs associated to the reconfiguration process, including the effects of reconfiguration speeds, is being carried out. Coming out with an optimal reconfiguration strategy is the final objective of this work, so as it can be included in the final architecture of the embedded system platform.

ACKNOWLEDGEMENTS

This work was supported by the Artemis program under the project SMART (Secure, Mobile Visual Sensor Networks Architecture) with number ARTEMIS-2008-100032.

REFERENCES

[1] HWICAP Product Specification: http://www.xilinx.com/support/documentation/ip_documentation/xps_hwicap.pdf

[2] Spartan-6 Family FPGAs: http://www.xilinx.com/publications/prod_mktg/Spartan6_Product_Table.pdf

[3] Koch, D.; Beckhoff, C.; Tørrison, J.; , "Advanced partial run-time reconfiguration on Spartan-6 FPGAs," *2010 International Conference on Field-Programmable Technology (FPT),* , vol., no., pp.361-364, 8-10 Dec. 2010

[4] Otero, A.; Morales-Cas, A.; Portilla, J.; de la Torre, E.; Riesgo, T.; , "A Modular Peripheral to Support Self-Reconfiguration in SoCs," *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD),* , vol., no., pp.88-95, 1-3 Sept. 2010

[5] J. Portilla, A. Otero, E. de la Torre, T. Riesgo, O. Stecklina, S. Peter, and P. Langendörfer. "Adaptable Security in Wireless Sensor Networks by Using Reconfigurable ECC Hardware Coprocessors," *International Journal of Distributed Sensor Networks*. Hindawi Publishing Corporation. Volume 2010, Article ID 740823, 12 pages

[6] Development Board: http://www.xilinx.com/products/devkits/EK-S6-EMBD-G.htm

[7] Spartan-6 Configuration Guide: http://www.xilinx.com/support/documentation/user_guides/ug380.pdf

[8] Virtex-5 Configuration Guide: http://www.xilinx.com/support/documentation/user_guides/ug191.pdf

[9] System ACE: http://www.xilinx.com/support/documentation/data_sheets/ds080.pdf

[10] IBM CoreConnect: http://www.xilinx.com/products/ipcenter/dr_pcentral_coreconnect.htm

[11] ARM AXI: http://www.arm.com/products/system-ip/amba/amba-open-specifications.php

[12] Kalte, H. and Porrmann, M. 2006. REPLICA2Pro: task relocation by bitstream manipulation in virtex-II/Pro FPGAs. In *Proceedings of the 3rd Conference on Computing Frontiers* (Ischia, Italy, May 03 - 05, 2006). CF '06. ACM, New York, NY, 403-412. DOI=

[13] Corbetta, S.; Morandi, M.; Novati, M.; Santambrogio, M.D.; Sciuto, D.; Spoletini, P.; , "Internal and External Bitstream Relocation for Partial Dynamic Reconfiguration," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol.17, no.11, pp.1650-1654, Nov. 2009

[14] Krasteva, Y.E.; de la Torre, E.; Riesgo, T.; Joly, D.; , "Virtex II FPGA Bitstream Manipulation: Application to Reconfiguration Control Systems," *Field Programmable Logic and Applications, 2006. FPL '06. International Conference on* , vol., no., pp.1-4, 28-30 Aug. 2006

[15] Kalte, H.; Lee, G.; Porrmann, M.; Ruckert, U.; , "REPLICA: A Bitstream Manipulation Filter for Module Relocation in Partial Reconfigurable Systems," *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International* , vol., no., pp. 151b- 151b, 04-08 April 2005

[16] Ferrandi, F.; Morandi, M.; Novati, M.; Santambrogio, M.D.; Sciuto, D.; , "Dynamic Reconfiguration: Core Relocation via Partial Bitstreams Filtering with Minimal Overhead," *International Symposium on System-on-Chip, 2006.*, vol., no., pp.1-4, 13-16 Nov. 2006

[17] Blodget, B., James-Roxby, P., Keller, E., McMillan, S., and Sundararajan, P. : "A self-reconfiguring platform", In Proceedings of the Field-Programmable Logic and Applications, September 2003. Springer-Verlag

[18] Blodget, B.; McMillan, S.; Lysaght, P.; , "A lightweight approach for embedded reconfiguration of FPGAs," *Design, Automation and Test in Europe Conference and Exhibition, 2003* , vol., no., pp. 399- 400, 2003

[19] Sudarsanam, A.; Kallam, R.; Dasu, A.; , "PRR-PRR Dynamic Relocation," *Computer Architecture Letters* , vol.8, no.2, pp.44-47, Feb. 2009

[20] Otero, A.;Krasteva, Y.E.; de la Torre, E.; Riesgo, T.; "Generic Systolic Array for Run-Time Scalable Cores", *6th International Symposium on Applied Reconfigurable Computing, ARC* 2010, Bangkok, Thailand, March 17-19, 2010