Using Partial Reconfiguration for SoC Design and Implementation

Yana E. Krasteva[a], Jorge Portilla[a], Félix Tobajas Guerrero[b], Eduardo de la Torre[a]
[a] CEI - Centre for Industrial Electronics. Univ. Politécnica de Madrid, 2d José Gutierrez Abascál, Madrid, Spain 0034 913363191;
[b] IUMA – Applied Microelectronics Institute. Univ. de las Palmas de Gran Canaria, Spain, Campus Universitario de Tafira - 35017 - Las Palmas

ABSTRACT

Most reconfigurable systems rely on FPGA technology. Among these ones, those which permit dynamic and partial reconfiguration, offer added benefits in flexibility, in-field device upgrade, improved design and manufacturing time, and even, in some cases, power consumption reductions. However, dynamic reconfiguration is a complex task, and the real benefits of its use in real applications have been often questioned.

This paper presents an overview of the partial reconfiguration technique application, along with four original applications. The main goal of these applications is to test several architectures with different flexibility and, to search for the partial reconfiguration ”killing application”, that is, the application that better demonstrates the benefits of today reconfigurable systems based on commercial FPGAs. Therefore, the presented applications are rather a proof of concept, than fully operative and closed systems. First, a brief introduction to the partial reconfigurable systems application topic has been included. After that, the descriptions of the created reconfigurable systems are presented: first, an on-chip communications emulation framework, second, an on chip debugging system, third, a wireless sensor network reconfigurable node and finally, a remote reconfigurable client-server device. Each application is described in a separate section of the paper along with some test and results. General conclusions are included at the end of the paper.

Keywords: Partial Reconfiguration, FPGAs, Wireless Sensor Network, Network on Chip, Rapid Emulation, Remote Reconfiguration, Reconfigurable Systems, Debug.

1. INTRODUCTION

Reconfigurable system tries to bring the best of the hardware (HW) and the software (SW) world. The best of the HW world is the inherited parallelism that permits to execute a task much faster that the sequential SW execution model. However, the SW main advantage is the higher flexibility that is represented by the possibility of easily reprogramming or updating the program to be executed.

A reconfigurable common architecture is mix of a microcontroller, or a microprocessor and a reconfigurable array that is represented in most of the cases by Field Programmable Gate Arrays (FPGAs). FPGAs devices have been in the electronic market since the 80’s, but they use is recently being spread due to their higher integration, increased performance and the inclusion of new features, like embedded multiplying, Digital Signal Processing specific blocks and even in some cases analog peripherals, Digital Analog Convertors and Analog to Digital Convertors.

On the other hand, apart from the embedded resources, FPGA reconfiguration techniques have also been improved. Nowadays there are several reconfiguration options, from on board system programming to runtime reconfiguration where the device can be reconfigured while the remaining system is active. From all the reconfiguration techniques, the partial runtime reconfiguration is the most powerful one as it permits not only to made changes in the devices while the system is running, but also to modify pieces of the reconfigurable array while the rest of the configured logic in the FPGA is kept active.

Since the appearance of the first partial reconfiguration based system in the ninety's, the benefits of the use of this technique and systems have been clear: i) silicon reuse – the same FPGA are can be time-shared by different tasks, ii) hot device updates after device deployment that is basically to apply the idea of SW like updates to hardware and iii) reduced design and reconfiguration times and even, reduced power consumption compared to classic full reconfigurable systems.

However, these benefits are questioned and even argued. Therefore the research community has been searching for the killing application that will clearly show the benefits of partial runtime reconfigurable systems over the conventional reconfigurable ones.

The partial runtime reconfiguration application domain is broad, from resource restricted devices running simple applications to high performance computing. For instance, the use of partial reconfiguration techniques in the adaptation process of reconfigurable coprocessor applied to accelerate the SW execution of application like, encryption [1], multimedia [2] and searching algorithms [3] is widely extended. Achieved accelerations in this case are in the range of ten to several hundreds of times. In this area, reconfigurability has also been exploited for high performance computing [4] and all kinds of adaptive filtering for DSP processing, for instance the adaptive FIR filter presented in [5].

Partial reconfiguration has also been used in all types of control systems, like robots [6], power control systems [7], automotive control systems [8], runtime automation control [9], and even space [10] and medical [11] systems.

Furthermore, in evolvable systems partial reconfiguration is used to evolve circuits using genetic algorithms [12]. This evolution process may also continue after circuit design leading to the concept of evolvable IP Cores. Such cores change some parameters in order to adapt themselves to new environments [13].

The list of applications could continue, however from all the currently proposed application, the application that seem to be closer to the industry, and is one of the oldest applications, is the one related to software defined radio [14] and cognitive radio [14].

This paper presents several use cases where the utilization of partial reconfiguration techniques has produced some benefits. There are two applications related to the improvement of SoC design techniques, where it will be demonstrated that methodological improvements or time savings in the SoC design flow can be achieved by the use of FPGA partial reconfiguration. The methodology related applications are: i) on-chip communication emulation framework. The presented application is focused mainly to attenuate the synthesis time disadvantages of current emulation based approaches and ii) n on chip debugging system, where the partial reconfiguration is used to insert different debug and monitoring modules in a reconfigurable system.

Another two applications show the benefits of partial reconfiguration in the final product itself, resulting in enhanced embedded systems where the exploitation partial reconfiguration is an important added-value of the system. The first device is a reconfigurable node for Wireless Sensor Network where a, simple, but reconfigurable, system is defined and used to demonstrate the possibility of remote upgrades of sensor interfaces and data processing adaptations. The final application is remote reconfigurable device that has been integrated in a multimedia oriented client server environment. This reconfigurable system is more complex compared with the previous one, and demonstrates fully user-transparent reconfiguration.

In the rest of the paper, each application is described in a separate section. The fast emulation framework based on partial reconfiguration can be found in Section 2, while the use of partial reconfiguration for debug can be found in section 3. The reconfigurable WSN node can be found in Section 4 and the remote reconfigurable devices in Section 5. As four different application are described in this paper, in order to facilitate the reading, in each section and introduction to the application domain have been included, as well as some tests and results related to the application presented in each section. Finally, overall conclusion can be found in section 6.

2. FAST NOC BASED SOC EMULATION

2.1 Introduction

A lot of efforts have been put in finding suitable prototyping approaches for the NoC based System on Chip (SoC) design. There are higher level solutions that permit to take advantage of fast design space exploration, like in [16], where a C++ library is built on top of SystemC, or based on the Matlab simulation environment, like [17]. Other, more accurate solution are based on mix models, systemC and VHDL for instance NoCGeN [18]. Usually, at this abstraction level, traffic generators and traffic consumer models that simulate real core behavior are used in order to reduce the simulation time.

On the other side, there are FPGA based emulation solutions proposed to drastically reduce the simulation and therefore systems evaluation time. For instance, in [19] a HW-SW FPGA based emulation framework is presented and combined with the Xpipes environment. In the same work, four orders of magnitude of speedup are reported. A different approach

can be found in [20], where in order to save some are, parts of a parallel system are loaded into an FPGA sequentially. In the same work, speedups of 80 to 300 in comparison with System C simulation are reported. Anyhow, each new FPGA generation has more logic available and thus permits to emulate bigger systems.

However, emulation not only depends on the available FPGA area, but also suffers mainly from long synthesis time, that can reach hours and, restricted statistic data extraction, compared to software based solution, that depends mainly on the available FPGA interfaces and IOBs.

In this section, hard cores reusability through partial reconfiguration on a flexible architecture is studied in order to reduce the synthesis time compared to a non-partial reconfiguration emulation flow.

2.2 Emulation System

The main idea in this application is to use partial reconfiguration to accelerate the emulation process avoiding resynthesis when variations on the communication setups are tested to explore different design space possibilities. A flexible partial reconfigurable system, described in the next subsection, has been designed and integrated in an emulation framework where the main original idea is to build the emulated communication scheme entirely by reusable hard cores from a hard core library

2.2.1 DRNoC Architecture

The reconfigurable architecture that will be used for the emulation system is presented on the left side of Figure 1. It is a mesh of Reconfigurable Routing Modules (RRMs) that define the application communication strategy. Routing modules can be: buffers, any type of NoC router with up to eight ports that fit in the reserved RRM area, switches, multiplexors feed-throughs and empty designs. RRMs are connected to all its eight neighbors through communication channels that are the only fixed element in the architecture. Apart from this links there is also a local channel that connects RRMs to the system cores through Reconfigurable Network Interfaces (RNIs). RNIs define a standardized interface between cores and the network. They are in charge of an appropriate data formatting. The main NI configurations that can be loaded in RNIs are: serializers, deserializers, packetizers, depacketizers, multiplexers, demultiplexers, buffers, feed-throughs and empty designs.

This architecture has been mapped in a Virtex II FPGA as it can be seen on the right side of Figure 1. The implementation has been done following a partial reconfigurable systems design flow presented in [21]. In this mapping each Core, RRM and RNI are reconfigurable regions of the FPGA, while the communication channels are implemented as bus macros that are special structures used in partial reconfigurable system in order to pass signals from one reconfigurable region another one.

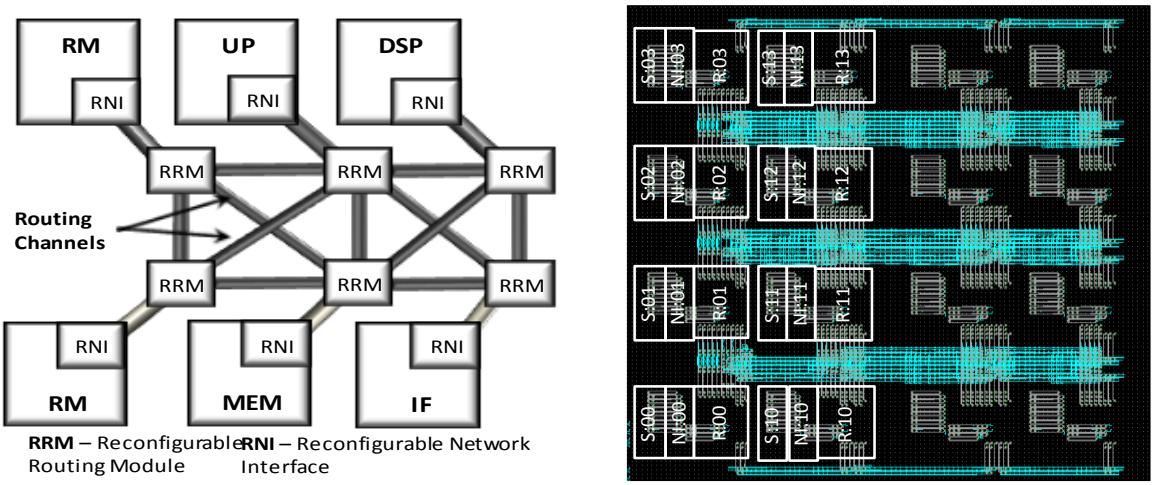

Figure 1. DRNoC Architecture on the left and a 4x4 DRNoC implementation on a Virtex II PFGA

2.2.2 Working Flow

The main distinguishing characteristics of the working flow for design space exploration is based on partial reconfiguration, it exploits hard core re-usability and also, is not restricted to NoC communications schemes.

The flow begins with a mapped application Communication Task Graph (CTG), where application tasks are assigned to system cores. For each node of the CTG, a suitable DRNoC element model, traffic receivers (TR) and/or traffic generators (TGs), are assigned if they are available in the hard core library or generated if they are not available (using the hard core design flow). Afterwards, in the first step of the interactive flow, the new CTG is mapped to the currently available in the emulation system DRNoC architecture (a 2x2 for a Xilinx Virtex II XC2V3000 FPGA). TGs and TRs are assigned to slot/cores, and the communication scheme is defined. NIs, routers, P2P and P2M links are assigned to RNIs and RRMs (each CTG to DRNoC mapping, along with an assigned communication scheme is called configuration). Also, in this step, measuring points are defined in terms of number and tracked TGs nodes.

After all configurations to be emulated have been setup the emulation is executed. For each DRNoC configuration, FPGA bitstreams, available in the hard core library or manually provided to the emulation system, are arranged and loaded in the FPGA.

All configurations are emulated consecutively, and results related to each measuring point are independently saved. Later, results can be plotted, analyzed by a user and, if needed, new DRNoC configurations can be added by modifying the CTG to DRNoC mapping and/or the communication schemes definition. This process continues until the best communication scheme and CTG mapping has been found.

2.2.3 Emulation Framework

The framework is distributed in three platforms: i) The DRNoC FPGA board, where measuring points are allocated in TRs and receiving nodes NIs, ii) the measuring system HW control mapped on an XUP board with an XC2VP30 FPGA and iii) a host PC that runs the DRNoC emulation SW.

The HW control system main element is a custom peripheral that is connected to the PPC on one side and directly to the DRNoC FPGA on the other side. The custom peripheral is in charge of controlling the emulation process (run, stop, reconfigure and continue), pulling data out from measuring points registers and buffer it.

A DRNoC Emulation control tool has been designed and runs on the measuring system host PC. The tool is in charge of controlling the entire emulation process following the described working flow and to administrate measured data and configurations. The tool includes a graphical user interface where a user defines the systems that will be emulated.

A set of possible configuration are held in a local library and loaded into the FPGA using the partial reconfiguration technique. After this, the system is emulated and measured latency and throughput data is retrieved from the system and saved in the host PC from where it is plotted and analyzed by the user. Afterwards, new configuration can be defined and loaded in the FPGA if it is needed.

2.3 DRNoC Emulation Framework Test and results

To validate the presented emulation method, a 2x2 DRNoC has been mapped into a Virtex II FPGA and several use cases have been setup an emulated following the emulation working flow and using the emulation software to plot the emulation results. The use cases have involved several partial reconfiguration of the DRNoC architecture in order to build the system to be emulated with hard cores that are held in the configurations library. Apart from this system validation, the needed reconfiguration and emulation time has been compared with the time needed to synthesis the same system (a no reconfigurable approach). For the worst use case, that requires several partial reconfiguration, the achieved speedup compared to a synthesis based solution is around 150 times, while for the best case that requires less reconfiguration to build the system, the achieved speedup is 400 times. Regarding the speedup comparison of the presented emulation approach with a simulation based systems, the achieved speedups are in the range of 500 to 800 for a 3 nodes NoC where each node transmits 10000 packets of 320 bits and for a NoC of 4 nodes that transmits the same amount of data.

These results show that the use of partial reconfiguration in rapid system prototyping is a good application domain. However the restrictions of the currently available partial reconfiguration methods results in a lot of wasted area and thus the systems that can be emulated using this method are quite small.

3. SOC DEBUG USING PARTIAL RECONFIGURATION

3.1 Introduction

The section deals with the use of partial reconfiguration techniques to insert different debug and monitoring modules in a reconfigurable system. The use of this technique has some advantages: first the debug module can be replaced by another one without stopping or modifying the execution of the core under debug, and second, the technique can also be applied for systems running in normal operation mode, where the behavior of a newly added core, and its interaction with the rest of the system needs to be verified priory to be integrated in it.

3.2 Debug System

The proposed, basic debug and monitoring system, is shown in Figure 2. The reconfigurable system is connected through a JTAG cable to a host PC where a tool for debug and monitoring FPGA based systems, called CHDT (Configuration Handling and Debug Tool) [22], is running. Debug and monitoring modules (DM) are hard cores (partial configuration files) that are loaded into the FPGA in an adjacent slot position to where a Core Under Test (CUT) is loaded (see Figure 2). DMs collect and/or control the CUT test outputs/inputs that are connected through a debug port and then this data is send out of the FPGA using the JTAG interface. All the data retrieved from the DM(s) is displayed graphically with CHDT, or with custom interfaces. The tool has also the possibility to detect via JTAG which debug modules are currently loaded in the system. This is done reading the JTAG UserCode register. Unfortunately this CHDT feature is not applicable when dealing with partial reconfiguration, because the reconfiguration of the UserCode register can affect the communication of running modules.

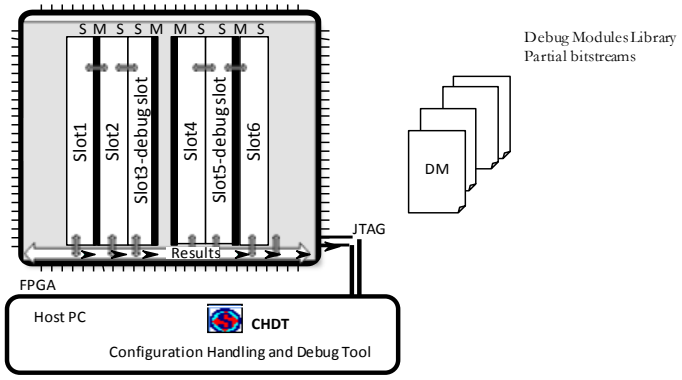

Figure 2. Partial reconfiguration for debug

Like in the previous application, all the available cores are kept in a local library and loaded in the system when needed.

The debugging system can be divided into three parts allocated in different FPGA areas. First, the CUT, which can be positioned in any slot. Second, the DM, which can be placed in a neighboring slot, right or left to the CUT. The communication between both modules is performed using the debug port. The third element is the JTAG controller that is allocated in the fixed FPGA area (FPGA right slot). The controller is in charge of modifying and retrieving data from debug modules and to send it to the CHDT tool running on the host PC.

3.3 Test and Results

CHDT contains a library of debug modules which range from very simple modules like event detectors, monitor registers or event counters, to complex modules like in-circuit emulators for microprocessor SW debugging or trace modules (which would use the embedded RAM of the FPGA for data storage).

It was checked that a given CUT works properly by searching for a certain, expected, transitions and register values, by using as DM a pattern detector module. Currently there are four DMs in the hard core library. As a result it has been noticed that the on chip debug process of reconfigurable systems has been improved with of the use of CHDT. It is extremely difficult to debug hard core behavior problems derived from some reconfigurations. However, some manual operations were required in order to place the monitoring signals in the appropriate positions for connecting them to the debug port bus macro. Debug modules were operated from specific module controller windows, as in the regular use of CHDT for non-reconfigurable systems.

4. RUNTIME HW/SW RECONFIGURABLE SOC FOR WSNS

4.1 Introduction

Wireless Sensor Networks (WSNs) are one of the fastest evolving and most challenging areas in today's electronic industry [23]v and [24]. The requirements to these resource restricted devices constantly increase. They are intended to be autonomous, low power, flexible and context aware.

For large, spread networks, which may be composed of hundreds or thousands of sensor nodes, deployment and network support are two complicate tasks which increase cost importantly. These processes would be easier, allowing cost effective changes or adaptations, if the WSN node has higher flexibility and adaptability features.

The application presented in this section tries to increase the system flexibility by designing a partial reconfigurable system on the node FPGA and provide a simple SW in order to test the system new capabilities.

4.2 Reconfigurable Wireless Sensor Node – Cookie

In order to design the partial reconfigurable system, first a suitable platform has to be available. In this case, the target platform is the Cookie modular sensor node platform presented in [25] and shown on Figure 3 that is composed of four main layers: processing, communications, power supply and sensors. The main distinguishing characteristic of the node is that it includes in the processing layer an FPGA.

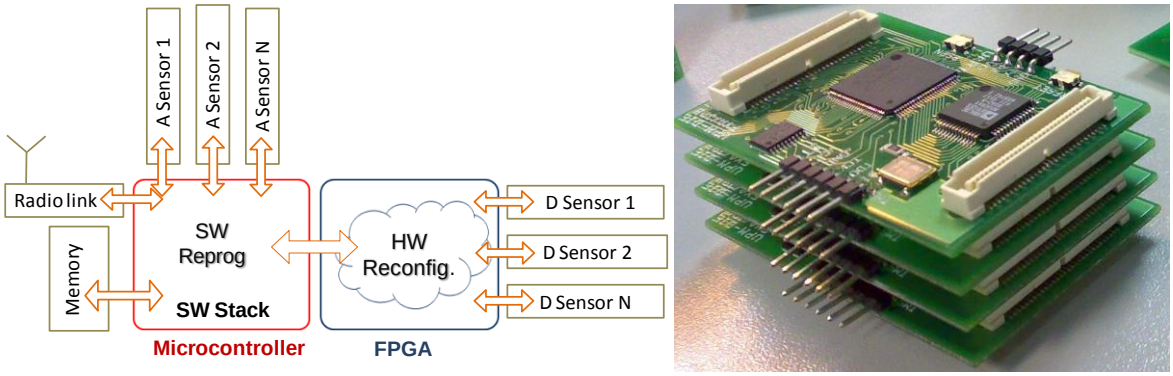

Figure 3 Remote partial reconfigurable sensor node. On the left a schematic view and on the right the PCBs (Printed Circuit Boards)

The dynamics of WSNs from the deployment and sensing point of view, and thus the required reconfiguration rate, are low compared to other systems. Therefore, WSNs will be considered "static". Regarding the reconfiguration process, that includes HW reconfiguration and SW reprogramming, two scenarios have been differentiated and will be targeted. The first scenario covers reconfiguration at network level that is mostly required during deployment, where the final function of each network node is defined (this includes the used sensors and data processing) or, when the network function is changed (like in an emergency situation). Generally this scenario might cover both, SW reprogramming and HW reconfiguration. In the context of the "Cookie" platform, HW reconfiguration is mandatory when modifications affect digital sensors, as they are connected directly to the FPGA (see Figure 3). Contrary, SW reprogramming is required when analog sensors are involved because they are directly connected to the microcontroller ADCs (see the right side of Figure 3). The second scenario is oriented to reconfigurations at node level and mainly involves HW reconfiguration. In this case, a reconfigurable array acts as a reconfigurable coprocessor where computation intensive tasks take advantage of the HW parallelism in order to lighten the microcontroller. With respect to the "Cookie" platform, when dealing with analog sensors, the entire FPGA is perceived as a coprocessor. Differently, when dealing with digital sensors, two functionalities are affected, the coprocessor and the digital sensors control.

Again, like on all the presented in the paper use cases, a partial reconfigurable system has been designed following g the method presented in [21]. For this case, the reconfigurable system has to support runtime changes in the sensor interfaces, in the data processing (coprocessor) and also in the uC-to-FPGA communication. The last one derives from the differences in the data processing of the analog and digital sensors. To do this, three reconfigurable regions or slots have been defined in the target Spartan 3 FPGA (XC3S200). As the system application is specific and well defined, wireless sensor networks, the slot functionality is also fixed and their size is adequate to this functionality. There is one

slot where the communicating interface with the external microcontroller is defined, one that defines the co-processing of the systems and, the last one where sensor interfaces are loaded.

A simple control SW has been created with the goal of evaluating the possibility of sending new configurations using the WSNs restricted network. The node control receives or sends new SW programs and HW configurations through the network and loads them in the microcontroller program memory. Regarding the FPGA programming, the available Spartan 3 FPGA does not have an internal configuration port (ICAP). The remaining FPGA configuration options are Select Map and JTAG. To keep the modularity and generality of the Cookie platform, JTAG has been selected as the reconfiguration interface. Furthermore, a SW implementation of the JTAG controller [26], provided by Xilinx, has been used and runs on the uC. The controller uses Boundary Scan configuration files as input and controls the FPGA JTAG interface signals connected to a microprocessor port. The input for the FPGA programmer is simply a pointer to the configuration to be loaded and therefore it satisfies the modularity property of the platform. On the other hand, the main disadvantage of using JTAG is the higher reconfiguration time, compared to other programming interfaces.

The available in the Cookie platform ZigBee communication module has been used for network data transmissions. The uC of the node, used to transmit the configuration data, sends commands to the ZigBee module to manage the communications channel and transmits the raw configuration data.

4.3 Test and results

In order to test the systems reconfigurability, several hard cores have been designed for the reconfigurable node: uC to FPGA interfaces, several FIR filters and several sensor interfaces, like temperature and acceleration. These configurations have then been transmitted to the target sensor node through the WSN network and loaded into the FPGA using partial reconfiguration. The tests performed have validated the reconfigurable system and have permitted to test the transmission of new configurations through the network.

For testing configuration transmissions/receptions and retransmissions, a set of HW configurations and the FPGA programming SW have been transmitted to a Node Under Reconfiguration (NUR) using different types of transmission media: i) using the available wireless communication in the sensor node and ii) using a serial cable connection and also using different types of packet formats: 16 and 8 bytes data transmission. Results show that the overhead in the transmission time when using a wireless media is around 2.5 times more, and the data rate is almost three times less, compared with a cable download. Reliability in WSN applications is an important aspect, even more if the network is also used for transmitting device updates. Tests have shown that the 16 bytes transmission fails with a high rate in long distance connections (with intermediate hops) compared to 8 bytes based transmissions that are much more reliable, but almost 40 % of the available energy budget has to be spent in this task. The ZigBee standard is not well suited for this kind of remote updates and therefore the amount of data to be transmitted has to be kept as low as possible and partial reconfigurable systems in this aspect are better than the classic reconfigurable ones.

The memory used for keeping FPGA partial configuration files and SW programs is the microcontroller program memory. Taking into account that this memory is 62 KB, this permits to keep the SW program and two or three partial configuration filers. However a full FPGA configuration exceeds the microcontroller memory.

Tests have shown that the remote node configuration has permitted to successfully change the sensor, currently used by the node, by a context switch achieving some node flexibility improvements.

Furthermore, partial reconfiguration has permitted to allocate some configuration locally, although the memory of the device is quite restricted.

Generally, the efforts of adding the partial runtime reconfigurable system in the node have not been high. Furthermore, the cost of programming the FPGA is negligible compared to the total node power consumption. The highest penalty (highest cost) is the configuration transmission and reception.

5. REMOTE RECONFIGURABLE SYSTEM

5.1 Introduction

The application presented in this section, similarly to the previous one is oriented to remote reconfigurable devices. But, differently from the WSN application, here the complexity of the entire systems is considerable. In this application, the final application is not restricted, like in the WSN node where sensing was the main point.

The overall system, shown in Figure 4, is comprised of various elements with different functionality, resulting in a scalable and easily extensible client-server environment.

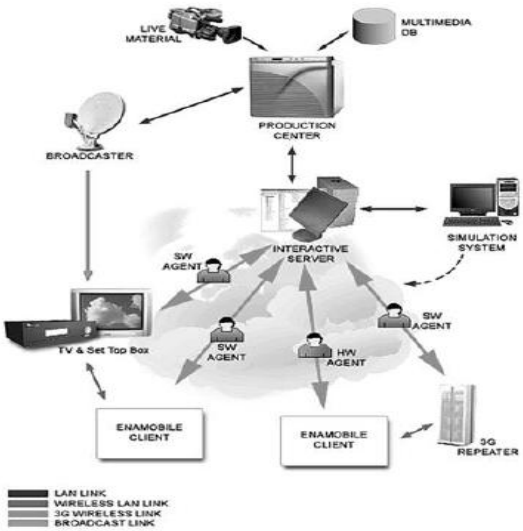

Figure 4. ENAMORADO Environment

The production center holds the Content Production Environment (CPE), which is primarily responsible for the production of the provided contents, in different formats, taking either live material and/or pre-recorded information. The Interactive Multimedia Server (IMS) interacts with the CPE and constitutes a central regulator of the system operability. The IMS supports the modeling of the full delivery scenario, from the production center to the terminals, as it controls the content delivery functionality in various formats and towards several devices. It also initiates the decision management process and retrieves content and event notifications from the CPE. The IMS accepts and analyses user requests for content delivery in various protocols (HTTP, WAP, etc.). It is the responsible for analyzing the profile data, send by the client devices, and take the final decision of which is the most appropriate configurations to be delivered to the client and, also to notify the Streaming Server to provide the client with content in the appropriate format. The Simulation Server (SS) selects the most appropriate content format depending on both user requirements and network conditions. This data is provided to the IMS in order to select the most appropriate configuration that will be sent to the client device.

The ENAMORADO environment deals with reconfigurable clients, called Enamobiles, able not only to update its software by downloading "software plug-ins", but also to adapt its hardware. This allows filling the gap between hardware and software, achieving potentially much higher performance than software, while maintaining a higher level of flexibility than fixed hardware. New hardware and software configurations are downloaded from the IMS to the client and installed in the respective embedded microprocessor and FPGA for a proper content reproduction. The FPGA can allocate more than one configuration at the same time and it is possible that some of these configurations need to be replaced without interrupting the mission of the other tasks. This implies the development of a dynamic partial reconfiguration strategy, on the fine grain FPGA and a full set of tools to control, negotiate, download, replace, delete and change configurations position within the FPGA have been developed.

5.2 ENAMORADO Reconfigurable Client

Differently from the previous use case, the needed flexibility and versatility by the reconfigurable system are higher as the foreseen applications to be executed in the system are broad (audio, video, control, data treatment, etc).

The reconfigurable device system is composed of two boards. First a custom board with a Virtex II XC2V3000, specially designed to meet partial reconfiguration requirements (peripheral elements placement among others) can be seen on the left side of Figure 4. Second, a commercial tablet PC development board, called XINGU, with a 600 MHz XScale microprocessor and two wireless networks, Wi-Fi for the short range connectivity and GPRS or 3G for the wide range. The board also has a compact flash memory and an expansion connector that is connected to the system bus. The board front view can be seen on the right side of Figure 5. The XScale runs a Linux operating system with the OPIE environment and the entire client software stack that includes; the low level drivers to control the FPGA programming, the FPGA manager that will be described next and the proxy server for the communication with the remote server.

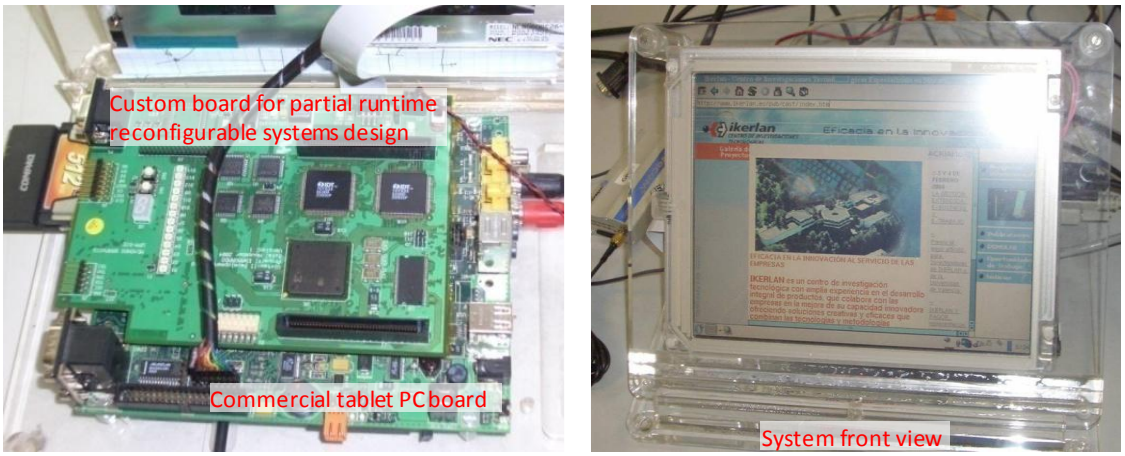

Figure 5. Remote reconfigurable device integrated in a client server system

A partial runtime reconfigurable system has been designed for this application and its main distinguishing characteristic is its high flexibility that permit to load hard cores in the system architecture which shapes and functionality is not known in advance (contrary to the previous application). This is achieved by defining an architecture where all reconfigurable slots are identical in shape and components and have a standardized communication interfaces. As a result, the main characteristic of the architecture are that it permit a hard core to be loaded in any slot position and that slots can be grouped to allocate bigger cores.

To deal with the FPGA reconfigurability, a middleware (FPGA manager) has been defined and has been integrated in the system software. This middleware is composed of a set of applications for managing, deleting, shifting and loading hard cores into the FPGA. For making the system versatile and portable, a hardware abstraction layer that specifies the currently available, in the reconfigurable system, virtual architecture has been defined. The manager is also in charge of loading empty and dummy hard cores in the respective slots when needed (if a hard core has been inactive for a long time) and also, to control and update the FPGA state string and the FPGA registry file. The registry file defines each core loaded in the FPGA. It has been kept as simple as possible and is a text file composed of several fields: hard core ID, slot position, used memory, a field that shows if the core is active and a time reference.

Apart from this, the manager is in charge of the communication with the upper layer of the SW stack that includes a proxy server responsible of the configuration negotiation, with the IMS server, process. New configurations are selected automatically by the IMS server taking into account a set of device characteristics (device profile). Part of the included data is related to the device reconfigurability, like the type of reconfigurable device (FPGA) and its architecture. The file also contains a list of cores, available locally in the unit, either currently loaded, or kept in memory. The profile is formatted in an XML file and is locally updated, in the client device, each time a new core is installed on it or a device characteristic is changed (for instance the amount of memory). The profile is sent to the IMS server each time a user accesses the service and it is then used during the configuration selection process.

5.3 Reconfiguration flow

The process of adapting the reconfigurable client device is as follows. A user through a reconfigurable device, accesses the IMS serve after a user logging procedure, where apart from the data related to the user, the reconfigurable device characteristics are also send transparently to the server. When a user selects a content to be displayed, a picture, a movie, an audio file etc, the IMS in combination with the SS analyzes the device and the user profile in order to select which is the best content quality that can be provided, depending on the network state and the user preferences. After this, a proper hard core configuration is selected and sent to the client device fully transparent. Once a new configuration is received in the client, the FPGA manager is executed and the FPGA partially reconfigured priory to display the content.

5.4 Test and Results

A set of hard cores have been designed and a set of configurations for the client device have been generated. These cores, along with each hard core definition file have been uploaded to the IMS server. Afterwards, the entire reconfiguration flow, from the user logging to the FPGA programming, including file updates has been tested.

Firs simple and small hard cores have been used for validating the system. By repeating the process several times in a specific order, the FPGA manager have been forced to relocate hard cores and the proxy server has been forced to update the device profile and the FPGA registry files.

Test results have shown that the entire process takes around 15 seconds in the worst case (executing all the flow steps), measured from the time right after the content selection by the used to the end of the FPGA programming. It is important to mention that the time needed to download the content depends on the network state. Therefore, the time needed from the reception of a new HW/SW configuration to the end of the system update process have been measured and is around 6 seconds.

After the reconfigurable system and the flow have been validated, further test have been done with bigger hard cores that are part of complete device configuration or codec's (HW/SW). Two codec's have been designed and have been integrated in the ENAMORADO environment. First, a filter that takes a .jpg piece of a picture (8x8 matrix) from the microprocessor, saves it in the on chip memory, applies a filter to modify the image and returns data to the microprocessor to be displayed. This process is repeated for the entire image. The hard core occupies three slots and has been fully integrated in the reconfigurable system. After the core has been installed in the system, the user can apply the filter to any image downloaded from the content server. Second, an open source software video decoding application has been selected, the VLC (Video AN Client media player), and the source code has been modified. Some resource consuming functions, like among others the IDCT (Inverse Discrete Cosine Transform), which is the main decompression operation for many multimedia formats, has been rewritten and adapted to be a hard core. The main problem here was that this hard core occupies almost all reconfigurable slots. Therefore it has not been fully integration in the reconfiguration flow. In this case the FPGA has been updated under full configurations.

Generally, the performed tests have shown that the reconfiguration flow of the system permits to have higher flexibility and versatility compared with the previous use case. The hard cores that can be consumed by the systems are broad in terms of used slots and can be designed by no partial reconfiguration designers. The limit is put by the available space like was in the case of the VLC player. Also, problems with the on-chip communication have been noticed. Flexibility in this sense is restricted to the on-chip communication protocol that is defined in the FPGA fixed area and each hard core that is loaded in a slot has to be able to support the defined communication protocol. The DRNoC architecture, described in the first section of this paper could be a good solution to this problem.

6. CONCLUSION

Four applications of partial reconfiguration have been presented. The first two ones are related to the use of partial reconfiguration during system on chip design. Firs, a complete NoC based SoCs emulation framework has been presented where the complete system is build by reusable hard cores and second, a debug approach where modules are introduced in the system using partial reconfiguration. The second two applications are related to reconfigurable devices. The first one is oriented to resource restricted embedded and application specific device where a partially reconfigurable system has been designed and mapped on a wireless sensor node that has an FPGA. Second, a more sophisticated, flexible and application independent reconfigurable system has been designed, also for an embedded device, but with more resources. In this case, the reconfiguration process is granted by a complete client server environment where the device updates are fully transparent for the client.

All the applications have been validated, tested and some test results have been included in each section.

Based on the grained experience, it could be said that the main problem is not relay related to finding the partial reconfiguration killing application domain, as it has been state in the paper abstract. All domains, including the presented in this paper, could be a killing one. Unfortunately, the main deficiency that remains and does not permitted to say that the killing application has been found, is the area related problems. A lot of area is wasted due to partial reconfiguration. This has not permitted to fully exploit reconfigurable systems and to demonstrate the applications real value. For instance the emulation framework would be more interesting if, an 8x8 DRNoC were used or, if the MPEG codec was fully integrated in the remote reconfigurable system.

Finally, as a general evaluation, from the presented applications probably the best domain, nowadays, is the emulation framework, after that the WSN and finally the multimedia remote reconfigurable device.

REFERENCES

[1] Martinez I. G., "Coprocesadores Dinámicamente Reconfigurables en Sistemas Embebidos basados en FPGAs," PhD thesis, Universidad Autonoma de Madrid (2006).

[2] Millberg, M., Nilsson, E., Thid, R., Kumar, S. and Jantsch A., "The nostrum backbone - a communication protocol stack for networks on chip," VLSI Design, 693–696. IEEE Computer Society, 2004.

[3] Oliver, T., Schmidt, B. and Maskell, D., "Hyper customized processors for bio-sequence database scanning on fpgas," Proceedings of the 13th international symposium on Field-programmable gate arrays ACM/SIGDA, 229–237 (2005).

[4] El-Araby, E, Gonzalez, I. and El-Ghazawi, T.A., "Bringing high performance reconfigurable computing to exact computations," Proceedings of the International Conference of Field Programmable Logic and Applications, 79–85 (2007).

[5] Choi, G., S. and Lee, H., "A self-reconfigurable adaptive fir filter system on partial reconfiguration platform," IEICE Transactions, 90-D (12),1932–1938 (2007).

[6] Upegi A., "Dynamically Reconfigurable Bio-inspired Hardware," PhD thesis, ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE (2006).

[7] Vasarhelyi, J., Imecs, M., Szabo, C. and Incze, I., I., "Run-time reconfiguration of tandem inverter for induction motor drives," Proceedings if the 12th International Power Electronics and Motion Control Conference, EPE-PEMC, 408–413 (2006).

[8] Becker, J., Hübner, M., Hettich, G., Costapel,R., Eisenmann, J.and Luka, J., "Dynamic and Partial FPGA Exploitation," Proceedings of IEEE 95(2), (2007).

[9] Alt, N., Claus, C. and Stechele, W., "Hardware/software architecture of an algorithm for vision-based real-time vehicle detection in dark environments,". Proc. DATE, 176–181(2008).

[10] Osterloh, B., Michalik, H., Fiethe, B. and Kotarowski, K., "Socwire: A network-on-chip approach for reconfigurable system-on-chip designs in space applications, " NASA/ESA Conference on Adaptive Hardware and Systems, 51–56 (2008).

[11] Massey, T., Dabiri, F., Noshadi, H., Brisk, P., Kaiser, W. and Sarrafzadeh, M., "Towards reconfigurable embedded medical systems," Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability HCMDSS-MDPnP., 178–180 (2007).

[12] Sekanina, L., "Evolvable Components," volume 9783540403777. Springer-Verlag New York, LLC, September 2008.

[13] Sekanina L., "Towards evolvable ip cores for fpgas," Proceedings of the 2003 NASA/DoD Conference on Evolvable Hardware, page 145 (2003).

[14] Design and Reuse News. Xilinx and isr technologies announce software defined radio kit supporting partial reconfiguration and sca-enabled soc. http://www.design-reuse.com, 2006.

[15] Delorme, J., Martin, J., Nafkha, A., Moy, C., Clermidy, F., Leray, P. and Palicot, J:, "A fpga partial reconfiguration design approach for cognitive radio based on noc architecture," Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference, 355–358 (2008).

[16] Coppola M., Curaba, S., Grammatikakis, M., D., Locatelli, R., Maruccia, G. and Papariello, F., "Occn: a noc modeling framework for design exploration," Journal of Systems Architecture, 50(2-3),129–163(2004).

[17] Saeidi Samira Khademzadeh Ahmad Mehran and Armin. Smap, “An intelligent mapping tool for network on chip,” Signals, Circuits and Systems ISSCS, 1–4 (2007).
[18] Chan J. and Parameswaran, S., “Nocgen: A template based reuse methodology for networks on chip architecture,” vlsid, 00:717 (2004).
[19] Genko, N., Atienza, D., De Micheli, G., Benini, L., Mendias, J., M., Hermida, R. and Catthoor, F., “A novel approach for network on chip emulation,” In ISCAS (3), 2365–2368 (2005).
[20] Pascal T. Wolkotte, Philip K. F. Holzenspies, and Gerard J. M. Smit, “Fast, accurate and detailed NoC simulations,” In Network on Chips, 2007
[21] Yana E. Krasteva, E. de la Torre and T. Riesgo, ”Virtual Architectures for Partial Runtime Reconfigurable Systems. Application to Network on Chip based SoC Emulation,” in Proceedings of IEEE Annual Conference of the IEEE Industrial Electronics Society, pp187 (2008),
[22] M. Garcia, E. de la Torre, F.Ariza, and T. Riesgo. “Hardware and software debugging of fpga based microprocessor system through debug logic insertion,” Proc. of the 14th Field-Programmable Logic and Applications (FPL”04), (2004).
[23] L.Q. Zhuang, K.M. Goh, and J.B. Zhang. “The wireless sensor networks for factory automation: Issues and challenges,” IEEE Conference on Emerging Technologies and Factory Automation, ETFA., 141–148 (2007).
[24] A. Willig. “Recent and emerging topics in wireless industrial communications: A selection,” IEEE Transactions on Industrial Informatics, 4(2):102–124 (2008).
[25] J. Portilla, A. de Castro, E. de la Torre, and T. Riesgo. “A modular architecture for nodes in wireless sensor networks,” Journal of Universal Computer Science (JUCS), 12(3), 328–339 (2006).
[26] Xilinx inc. Xilinx application notes, XAPP 058 v0.4 : In-System Programming Using an Embedded Microcontroller, October 2007.