Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2008, Article ID 367860, 11 pages
doi:10.1155/2008/367860

Research Article

An Evaluation of Dynamic Partial Reconfiguration for Signal and Image Processing in Professional Electronics Applications

Philippe Manet,[1] Daniel Maufroid,[2] Leonardo Tosi,[3] Gregory Gailliard,[2] Olivier Mulertt,[4] Marco Di Ciano,[5] Jean-Didier Legat,[1] Denis Aulagnier,[6] Christian Gamrat,[7] Raffaele Liberati,[8] Vincenzo La Barba,[9] Pol Cuvelier,[10] Bertrand Rousseau,[1] and Paul Gelineau[2]

[1] *Université catholique de Louvain, Place du Levant 3, 1348 Louvain-la-Neuve, Belgium*
[2] *Thales Communications, Boulevard de Valmy 160, 92704 Colombes, France*
[3] *CESVIT MICROELETTRONICA, Via F. Frediani, 59100 Prato, Italy*
[4] *MBDA, Avenue Réaumur 1, 92358 Le Plessis Robinson, France*
[5] *Tecnopolis CSATA, Str P. Casamassima km 3, 70010 Valenzano Bari, Italy*
[6] *Aerospace Division, Thales, Avenue de la 1ere DFL 10, 29283 Brest, France*
[7] *CEA LIST, CEN Saclay, 91191 Gif Sur Yvette, France*
[8] *ELETTRONICA, Via Tiburtina Valeria km 13, 700, 00131 Rome, Italy*
[9] *Thales Italia, Via E. Mattei 20, 66013 Chieti Scalo, Italy*
[10] *Thales Communications, Rue des Frères Taymans 28, 1480 Tubize, Belgium*

Correspondence should be addressed to Philippe Manet, philippe.manet@uclouvain.be

Received 29 February 2008; Revised 11 July 2008; Accepted 3 November 2008

Recommended by Guy Gogniat

Signal and image processing applications require a lot of computing resources. For low-volume applications like in professional electronics applications, FPGA are used in combination with DSP and GPP in order to reach the performances required by the product roadmaps. Nevertheless, FPGA designs are static, which raises a flexibility issue with new complex or software defined applications like software-defined radio (SDR). In this scope, dynamic partial reconfiguration (DPR) is used to bring a virtualization layer upon the static hardware of FPGA. During the last decade, DPR has been widely studied in academia. Nevertheless, there are very few real applications using it, and therefore, there is a lack of feedback providing relevant issues to address in order to improve its applicability. This paper evaluates the interest and limitations when using DPR in professional electronics applications and provides guidelines to improve its applicability. It makes a fair evaluation based on experiments made on a set of signal and image processing applications. It identifies the missing elements of the design flow to use DPR in professional electronics applications. Finally, it introduces a fast reconfiguration manager providing an 84-time improvement compared to the vendor solution.

1. INTRODUCTION

Signal and image processing applications require a lot of computing resources. Until recently, many of them were implemented using digital signal processor (DSP) that provides flexibility and computing power at a low cost. Nevertheless, since the beginning of the decade, the frequency of the processor cores does not increase anymore while their power consumption increases dramatically with performances [1, 2]. This power wall leads to unmanageable thermal and autonomy issues in embedded systems with an emphasis for battery-powered applications. Despite those limitations, the application roadmaps are still requiring a substantial increase of computing resources. In order to cope with those limitations, consumer electronics applications use the system-on-chip (SoC) approach where optimized accelerators supply DSP and GPP cores to meet the system constraints within a limited power envelope [3]. This approach has the main drawback to be very expensive and leads to complex systems that are difficult to validate.

Therefore, they are used for very high volumes applications like mobile phone handsets [4].

Professional electronics applications are characterized by low volumes and a high design count compared to consumer ones preventing the use of an expensive SoC approach. They rather use DSP or GPP combined with FPGA in order to increase the computation capabilities and meet requirements imposed by the roadmaps. Nevertheless, FPGA designs are static and lack flexibility compared to a pure DSP approach. However, roadmaps' evolution is shifting to multistandards or even software defined applications, requiring a virtualization layer to adapt or change their behavior dynamically. It is for example the case for the software-defined radio (SDR) application [5]. In order to cope with this flexibility issue, DPR of FPGA can be used [6, 7]. Indeed, it brings virtualization upon static hardware making it possible to handle hardware functional blocks like software components. Thank to this feature, SDR is announced by Xilinx to be the main target applications for DPR [8].

During the last decade, DPR has been widely studied in academia [9–11]. There are many works related to the component programming issue as well as case studies for possible applications [12–17]. Nevertheless, all those experiments are carried out for research purpose, and only a few of them take into account its impact for real applications. Indeed, despite all the researches done, it exists today very few real applications using DPR. Therefore, there is a lack of feedback on its use in real products providing relevant research issues to address in order to improve its applicability.

The contribution of this paper is to evaluate the interest and limitations when using DPR in real professional electronics applications, and to provide guidelines to improve its applicability. First, it makes a fair evaluation based on experiments made on a set of seven signal and image processing applications carried out in real conditions. Second, based on a precise analysis of the current flow for real usage, it identifies the missing elements for its use in professional electronics applications with highlights on the issues raised by SDR. Third, it identifies a set of advantages for using DPR in professional electronics applications. Fourth, it provides research directions in order to improve its usage. Fifth, it introduces a fast reconfiguration manager used by the experiments providing an 84-time improvement compared to the vendor solution. To our knowledge, it is the first reconfiguration manager working at this speed on the Virtex 4. Indeed in [18], the full speed has been tested only in Virtex II for an 8-bit ICAP.

This paper is based on the researches carried out as part of the RECOPS project that aims to study the use of DPR in military applications [19]. The evaluation is made on FPGA from the Virtex family from Xilinx Inc., Calif, USA. As they are the biggest matrixes available supporting DPR, they were selected as the FPGA platform for the project.

In the remainder of this paper, Section 2 explains the specificities of professional electronics applications. Section 3 exposes the DPR possibilities in the latest Xilinx Virtex components with its current design flow usable in real applications. Sections 4 and 5 give the main interests for using DPR in professional electronics applications and the issues for their implementation using DPR. Section 6 presents the experiments carried out. Section 7 details the main results obtained, and Section 8 discusses remaining issues raised by this technology based on the experiments, followed by the conclusion in Section 9.

2. PROFESSIONAL ELECTRONICS APPLICATIONS

Nowadays, the electronics market is mainly directed by consumer applications. They are characterized by a very short time to market, high volumes, autonomous and battery-powered applications, and have very few validations or qualification constraints.

However, professional electronics applications are quite different. They are sold in low volumes, and they take longer time for development. They are often maintained, or even updated, during their lifetime that can reach several decades. They need to address precise requirements and validation processes. Moreover, most of them are strongly coupled with other components inside complex systems. It is for example the case for the electronic subsystems in a plane, or for an airport radar part of an air traffic management system. They are also quite diverse. Indeed, a radar application, for example, performs in real time a very large FFT of up to tens of thousands of points and must fit in a small volume in order to be integrated into a plane, while a software-defined radio (SDR) requires a high level of virtualization, with very low power consumption for handheld devices.

For systems with a high level of safety, further validations and certifications are performed. It is the case for systems that may affect the life of human beings. The ED80/DO254 for hardware and ED12B/DO178B for software, in civil aviation, are among the most restrictive standards [20]. They impose to strictly validate and demonstrate that the application requirements are fulfilled in any possible situation, and identify the possible failure modes.

3. DYNAMIC PARTIAL RECONFIGURATION

Current FPGAs are composed by a user programmable logic plane, configured by an underlying memory plane. The logic plane holds fine-grain customizable logic made of LUT and interconnection resources, but also optimized macroblocks like SRAM arrays, DSP accelerators, clock tree managers, I/O modules, and so forth. For most of the components, the configuration memory is filled at startup by a bitstream through a reconfiguration interface. In order to do this, several interfaces are implemented; they are connected to the reconfiguration engine accessing the configuration memory. In Xilinx FPGA, the bitstream is a packet sequence, each packet containing a header and its data. The header holds commands and information for the configuration engine, so that very few external control signals are required during the reconfiguration process.

In some components of the high-end Virtex family from Xilinx, an internal interface internal configuration access port (ICAP) allows accessing the configuration engine directly from the user logic plane. Thanks to this interface, it is possible for an application to perform self-reconfiguration while it is running. Moreover, according to the vendor, the internal implementation of the configuration ensures that (i) modifying a region of the component does not affect the configuration memory of other, unmodified, regions and (ii) when the content of the configuration memory is overwritten by the same content, its corresponding logic can operate normally without being affected. In order to perform a dynamic partial reconfiguration (DPR), a partial bitstream is written into the ICAP. Since it is made of raw configuration data highly component dependent, special tools are required for its generation.

3.1. New DPR possibilities in latest Xilinx Virtex component

The Virtex family encompasses the highest performances and biggest FPGA from Xilinx. Furthermore, DPR was introduced in this family with the Virtex II component. Its layout was organized in columns, defining an entire column of the component as the basic configuration granularity. Therefore, the DPR had to cope with severe hardware constraints that lessen its use in professional applications.

Since the Virtex 4, several improvements facilitate the use of DPR, making it a credible solution for some specific applications like software-defined radio (SDR). The improvements for DPR are on the layout architecture, the clock tree, and the ICAP. The component is still organized in columns but the partial reconfiguration granularity is reduced to a frame, which is here only a part of a column. Therefore, the partial reconfigurable region (PRR) can be almost any height and width. Clock regions are rectangular, allowing to be matched by a PRR. This leads to better timing performances and clock tree management. Moreover, the output frequency and phase shift of the digital clock manager (DCM) can be modified using DPR.

Finally, the width of the ICAP port is extended to 32 bits, and its speed is increased up to 100 MHz. This significantly speeds up the reconfiguration time, which is a main concern when using DPR. Indeed, the HW in the PRR cannot be used during the reconfiguration process. With all those improvements, the HW capabilities are far less a limitation for using DPR in real applications.

3.2. New tools dedicated for reconfiguration

The implementation of DPR in Virtex FPGA requires a nonstandard flow [21]. It is driven by constraints on the area and primitives location. It uses special dedicated resources, bus macro (BM) for communications, and uses the PlanAhead tool.

Area constraints make it possible to partition a design in a fixed part and a reconfigurable part. They are required to force a design to be placed & routed (P&R) in a predefined region composed of a fixed set of frames.

The BMs are slice-based prerouted elements made of simple LUT. They enable the communication between the fixed and the dynamic parts of the design. Indeed, since placement constraints cannot be directly applied to routing resources, they lock the starting or the ending point of the wires to ensure their correct connection after reconfiguration. They can be located on any edge of a PRR (left, right, top, bottom). They can be asynchronous or clocked, have a fixed direction (in or out); they are device dependent. Their main drawbacks are that they consume a significant amount of resources, and their automatic placement is not supported by the tools.

PlanAhead is a proprietary tool from Xilinx that is used to manage the DPR constraints and to drive the implementation process. It is a production tool with graphical interface allowing to handle constraints and area location of a PRR at component level. It provides a hierarchical design view at netlist level and a resource view at component level. Moreover, it makes a rough resources and bitstream size estimation, performs some design rule checking, and exports the results for implementation.

Finally, a specific partial flow is used to P&R the different portions of the design. It is based on a modular flow available in ISE [22]. It requires special patch updates that are currently available for selected customers and research centers. The flow is compatible with the embedded processors (PPC and MicroBlaze). At the end, it provides full and partial bitstreams that can be directly written into the ICAP. Note that with this flow, the modules do not have to follow the restrictive constraints of the XAPP290 [23] anymore.

3.3. The next generation

The latest high-end FPGA available in Xilinx is the Virtex 5. It is fabricated in a 65 nm technology, and it introduces innovations like diagonal routing capabilities and 6-entry LUT. It has new configuration ports and can manage multiple configurations. Regarding DPR, the component will support the features available on Virtex 4. Since DPR is today highly component dependent, specific tools and patches are required on top of the standard flow as for Virtex 4. However, it is not sure that the upcoming components will support DPR, and there are today no clear vendor roadmap regarding this technology.

4. ADVANTAGE FOR USING DPR IN PROFESSIONAL ELECTRONICS

The DPR allows using more hardware than that physically present in the FPGA. This can be used to reduce the size of the FPGA and its overall power consumption. This also permits to execute an algorithm with an optimized implementation depending on its parameters and data set. Furthermore, upon the usual speed and power research goals, DPR offers system-level advantages for professional electronics [19]. The advantages considered are the following.

(a) Task speed. By shifting the partitioning between hardware and software toward (faster) hardware. More functions can be implemented in hardware without being limited by the size of the component.

(b) Power reduction. By having less hardware instantiated and running, and by using a smaller FPGA.

(c) Survivability. By allowing selection and operation in a degraded mode when a part of the system is damaged. It is necessary for applications running in harsh environments where environmental conditions can exceed the normal operating range.

(d) Mission change. By allowing to configure the application for an entire mission without interrupting services. The real-time issue is not critical here. The application is configured for a long period. DPR provides an easy and safe way to strongly modify an entire system without having the complexity of implementing all the functionalities in one design. It is very useful and it facilitates the validation of applications interconnected in a complex system.

(e) Environment change. During operation, the application can be developed specifically for several environments and switch dynamically. Here, the real-time issue is critical.

(f) Adaptive algorithm change. By adapting dynamically an algorithm depending on the external conditions. It is a lower granularity than environment change.

(g) Online system test. A system in harsh environment can be damaged. For critical systems, it is necessary to know its level of functionality, for example, to decide to power on a redundant one.

(h) Hardware virtualization. By having more hardware available than that physically present in the FPGA. It allows to manage a set of hardware modules as a component library. It is used for example in SDR applications.

5. IMPLEMENTATION OF DPR IN PROFESSIONAL APPLICATIONS

In order to implement DPR in real applications, some issues need to be carefully handled. The most important ones are the design flow, the constraints brought by the use of DPR on the application board, and the minimum development required for its integration.

5.1. *The design flow issue*

The deep validation requirements or even the certification in critical applications is not only performed on the final product, it also imposes the use of a validation methodology during the whole development process. In this scope, the design flow is a key point. For severe requirements, it must be certified [20]. The validation imposes that at each step of the development, from the first specifications to the final tests, one must be able to verify that the application meets the requirements. For this, it is necessary to have precise simulations and modeling capabilities and to have efficient tools, at least for productivity reasons.

The standard design flow for a digital system is based on a top-down approach. The main steps are the following.

(a) System specification.

(b) Functional modeling and simulation.

(c) Hardware/software partitioning.

(d) Architecture definition.

(e) HW and SW development.

(f) Platform integration.

(g) Validation and qualification.

A complete design is rarely done in one conception pass. Indeed, in order to meet the application requirements, it is often necessary to make several iterations until reaching acceptable performances. The following paragraph focuses on the missing elements for DPR regarding this flow.

5.2. *Identification of missing elements for the dynamic reconfiguration flow*

For each step of the design flow, it is mandatory to have a simulation model of the DPR. For the first steps until the architecture definition, a model can be built with SystemC. For critical designs or for productivity issues, those steps are usually automated by tools. None of the existing tools supports DPR. Nevertheless, it is important to note that FPGAs are generally not well supported. It is for example difficult to find SystemC models for the complex hard or soft IP in FPGA like the PowerPC 405 or the MicroBlaze.

For the hardware development step, it is necessary to have a behavioral model of the ICAP, the BM, and the configuration process. Indeed, a hardware-level model is required to make functional simulation, ensure real-time constraints, and for debug. Without a behavioral model of the hardware, it is only possible to simulate the modules independently and verify the design when committing the application to the final on-board tests. Then, it is possible to verify a DPR design only when performing the platform integration. Therefore, if something fails, it is not possible to reproduce the problem by simulation. Moreover, debugging tools like ChipScope from Xilinx [24] are not working in a reconfigurable region. Without complete behavioral models, the validation and qualification step can only be done on the hardware platform. Serious difficulties occur, when it is necessary to take into account the design flow for validation.

The DPR modeling, using a powerful tool like SystemC [25–28], is mandatory in order to implement complex and challenging applications like SDR [29, 30]. Moreover, in order to maintain consistency in the model but also preserve good validation capabilities, the successive refinements of the SystemC model during the design flow should also support DPR.

5.3. *The hardware platform and constraints*

The board here is the printed circuit board hosting the FPGA. Its constraints are of two kinds. The first is that it requires a large amount of external FLASH memory, in order to permanently store the partial bitstreams, but also fast external memory like DDR in order to load bitstream and perform a reconfiguration at the maximum speed. The second kind of constraints is on the I/O position. For example, with DDR I/Os, the memory controller needs to be placed in front of the I/Os, constraining the placement of the PRR. Regarding those constraints, the standard development boards can easily be used to test DPR on a reference application.

5.4. Developments required for DPR evaluation

The developments required for evaluation are the design of a reconfiguration controller and a scheduler. The controller is based on an interface between the on-chip peripheral bus (OPB) available in the Virtex and the ICAP. It offers the flexibility of the standard bus that allows connecting any type of memory through a standard interface. Furthermore, the bus benefits from DMA services that provide sustained data rate to reach the maximum speed of the ICAP. Since the configuration manager is connected to a bus that can be connected to a MicroBlaze or a PowerPC, the scheduler can be implemented in software, which strongly reduces its development cost. Note that an OPB-ICAP interface is available from Xilinx, but not for all the components, and the full utilization of the ICAP is not supported on the latest one.

6. EXPERIMENTS

The DPR is evaluated in industrial conditions on a broad range of applications mainly in the field of defense applications. The approach is to use a common platform in order to easily share development experiences, make relevant comparisons, and demonstrate the platform flexibility offered by the FPGA implementation. The broad range of applications is required by the diversity of applications and constraints encountered in professional electronics.

6.1. The evaluation applications

The experiments are made on real applications or on a representative part of them. They have been chosen in order to have a representative set of challenging applications in the field of signal and image processing in professional electronics. They have not been specially chosen for a good match with DPR. Nevertheless, they all offer opportunities for improvement by using DPR compared to the usual solutions at least with several advantages discussed in Section 4. They are listed in the following with a highlight on their implementation challenge.

(a) *Portable device for remote-control video capture and transfer.* This application realizes image acquisition and transfer with remote control video using wireless communication. Several compression algorithms and several bandwidths have to be supported. The quality of the image and the bandwidth are both adapted depending on the context and the battery charge. All must fit within a minimal HW. The reconfigurable process performs axis motion control, image capture, and data transmission in the same reconfigurable region.

(b) *Blanking management for naval electronic counter measure/electronic support measure systems.* This application generates control signals avoiding interferences between the ECM and ESM systems. The system must operate with a high level of safety and support multiple context switches. Therefore, the design complexity must be as low as possible.

(c) *Real-time image processing unit for missile applications.* It applies several operators on an image with hard real-time constraints. The number of operators is potentially high, and they must be optimized in order to be very fast. The change of operators must be taken within a narrow time slot. The operators are implemented in a reconfigurable region.

(d) *Front end processing for airborne radar.* The maximum speed of the FPGA is used. The functionality must change very quickly, and the scenario depends dynamically on the context of the application. It tests the impact of reconfiguration on the use of high-speed I/O links and the front-end data processing under hard real-time and jitter constraints.

(e) *Short range radio modem.* It is used for local and private data communications and needs to support several data rates. Only the baseband of the modem needs to change but it is a very high computational task with only signal processing. The goal is to obtain the highest data rate in a given situation. The baseband functions of the modem are implemented in a reconfigurable region.

(f) *Software defined radio transmitter.* Only the modulation waveforms are implemented in a reconfigurable region. It tests the waveform parameterization and change. As the SDR application is highly reconfigurable, the maximum variability of the modules is required.

(g) *Software defined radio receiver.* This application focuses on all the upper layers of the SDR. Since they are usually implemented in software, the HW/SW interface with a high variability support is a main challenge when using dynamic hardware. It evaluates the hardware virtualization brought by the dynamic reconfiguration and demonstrates the partial reconfiguration of the FPGA by software from the SCA core framework. It has been published in [31].

Each demonstrator covers its own area of assessment, but together, they cover a large panel of activities and techniques. The tests are carried out to emphasize the different areas where the use of DPR can be interesting. It is not possible to detail them all here. Nevertheless, we will highlight features of the software defined radio application.

6.2. The software-defined radio experiments

The opportunity for SDR is to use the virtualization brought by DPR in order to be implemented in hardware, rather than in software. It is thus a very good candidate to take benefit of the dynamic reconfiguration. The experiments conducted with the SDR transmitter for multiband, multimode radio, are performed by switching between two different waveforms, the D8PSK and the 16QAM. The reconfiguration is controlled by the internal Power PC core of the FPGA. The mappings of data patterns to symbol, and symbols to in-phase and quadrature components, are done in the PRR. The pulse-shaping filter is also implemented in the PRR. The experiments with the receiver implement the SCA layer of the SDR in an FPGA platform using DPR.

6.3. The evaluation platform

The platform targeted for all the experiments is the ML410 development board from Xilinx. It hosts an XC4V-FX60 with sufficient external storage resources for storing the partial bitstreams for all the applications. No other specific board development is needed by using the platform. Nevertheless, the boards ML403 and ML405 are also used for some experiments (due to the late availability of the ML410 board). The components used by all the experiments are the XC4V-FX12, FX20, FX60, and LX60.

6.4. The reconfiguration controller

For hard real-time applications that need to change frequently the reconfigurable module, a high bandwidth through the ICAP port is required. It is for example the case for the image processing application, which applies sequentially a set of reconfigurable functions for each image in a narrow time frame. The theoretical speed of the ICAP is 100 MHz, and its width can be programmed as an 8-bit or a 32 bits port allowing a bandwidth of 0.75 or 2.98 Gbit/s. The ICAP is usually connected to an OPB bus. This allows to use easily all the existing memory resources of the board by means of standard interfaces. The bus is driven by a microprocessor that can be a hard embedded PPC core or by a soft MicroBlaze core. Unfortunately, when the experiments were carried out, the interface provided by the vendor was able to operate at no more than 40 MHz in the 8-bit mode, and was designed for the Virtex II component. Those performances are not sufficient to meet the real-time constraints of the image processing application. Therefore, an improved version based on the OPB-ICAP interface for Virtex II was developed. The improvements allow working with the ICAP in the 32 bits mode at 100 MHz. Moreover, it also provides DMA services allowing to use the OPB bus with the required bandwidth.

Figure 1 shows a functional description of the reconfiguration controller implementing the OPB-ICAP interface. It is viewed as an OPB peripheral through an IPIF from the bus.

It is written in VHDL, and generic parameters allow selecting the memory size as well as the ICAP mode of 8 or 32 bits. A control and status register allows writing and reading bits to command and control the reconfiguration controller. It implements a DMA able to steer data from the bus at a sustained rate. The parameters are first written in two registers, a DMA_START_ADDR register that holds the base address of the data segment and a DMA_BURST_SIZE register that defines its size (max 1024 words). Then, a start bit in the CTRL/STATUS register is set to start the transfer. When it is finished, a status bit is set in the same register. The BRAM memory is used as a buffer. It was originally implemented in the Virtex II controller to convert the data from a 32 bits stream from the bus to an 8-bit stream toward the ICAP. Nevertheless, its instantiation is not mandatory. Indeed, the ICAP can be controlled at data word level by an enable signal, thus a single 32 bits register can be used to split the 32 bits data in four 8-bit words. When using the DMA, the buffer is bypassed. The reconfiguration controller supports read back, allowing to transfer the content of selected configuration frames to the bus. The DMA is not supported with this feature. Configured in the 32 bits mode, the reconfiguration manager occupies 973 slices and 1 BRAM. It corresponds to 3.7% of the mid-range XC4VLX60 Virtex 4 component.

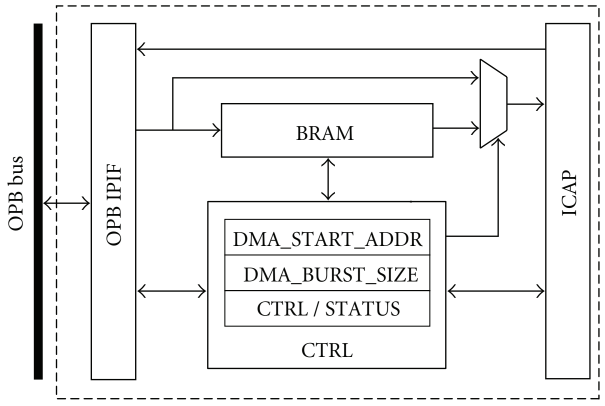

Figure 1: Reconfiguration controller.

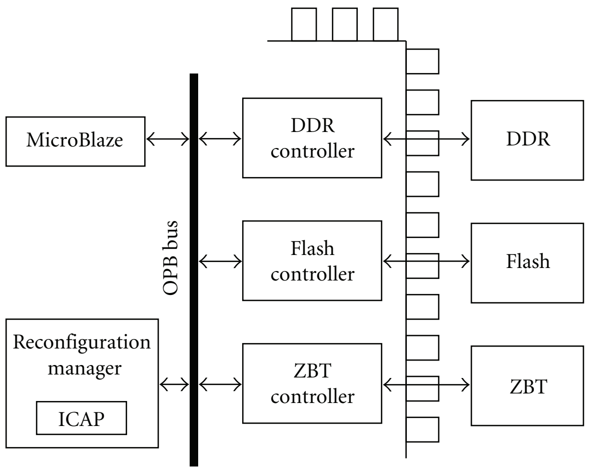

Figure 2: Memory organization.

The memory organization for the image processing application is given in Figure 2. The external flash is used to permanently store the bitstreams. The DDR and zero bus turnaround (ZBT) are used as fast memories. The bitstream size for the reconfigurable region is around 300 KB for the biggest. In order to obtain high-speed transfer, the partial bitstreams need to be stored in a fast external memory. Indeed, the permanent flash memory is very slow; it can only deliver 8 bits at 10 MHz. Therefore, the ZBT SRAM or the DDRs are used since they are the only memories on the board able to sustain the 100 MHz, 32 bits, throughputs. When the application starts, the bitstreams are first copied from the flash to the ZBT SRAM or the DDR depending on the speed requirements of the application.

7. RESULTS

The results are presented here from a system-level approach with highlights on the virtualization in the SDR experiments. Finally, measures of the performances of the ICAP are detailed.

The characteristics of the seven applications are given in Table 1. The first column gives the XC4V component

Table 1: Applications characteristics.

	Component	N. reconf. region	PRR size [%]	Reconf. time [ms]	Bitstream size [KB]	Real-time constr.
(a) Im. acq.	FX60	1	16	2.2	330	20 ms
(b) Naval	FX12	1	28	195	90	1 s
(c) Im. proc.	FX60	1	13	1.3	17	5 ms
(d) Radar	FX20	1	33	0.47	166	1 ms
(e) Modem	FX12	1	32	3.5	241	1 s
(f) SDR Tx	FX12	1	9	31.6	38	1 s
(g) SDR Rx	FX12	2	12	15.5	46	1 s

used, then the number of reconfigurable regions and the relative size of the reconfigurable regions counted in slices. The fourth column gives the reconfiguration time followed by the maximum size of all the partial bitstreams targeted to a region. Note that not all the applications use the fast reconfiguration manager. Moreover, they can use 8-bit or 32 bits interfaces, and their bus may be interrupted by other tasks. Therefore, the reconfiguration time summarizes the application performances for doing a reconfiguration while respecting its real-time constraints. The last column gives the real-time constraint for performing a reconfiguration.

7.1. System-level assessment

Most of the advantages listed in Section 4 are tested by the experiences. The results are given in Table 3. For each advantage, a rating between −2 and +2 evaluates its benefit in the application. An empty evaluation means that the advantage was not tested during the experiments by the application. A rating between parenthesis means that the advantage was not directly evaluated during the experiments but rather estimated. Results show that the most interesting advantages are virtualization, environment, and mission change. They allow changing the functionality of a system under weak real-time constraints. Moreover, in all the applications using this feature, the architecture of the system remains the same while an instance of a particular function is changed. This leads to add more functionalities without increasing the complexity of the data path of the application. The telecommunication applications change their waveforms simply by reconfiguring a region. The adaptive algorithm change is obtained by the virtualization advantage. It was only lightly tested by the experiments, since this feature is today rarely used in the applications due to its novelty. There was no increase of the tasks' speed of the application, except for the SDR applications, where more accelerators can be instantiated in the FPGA instead of being executed on a DSP. For the other applications, the accelerators are always instantiated in hardware. The power reduction is then obtained by using a smaller FPGA when using DPR, reducing the leakage and the dynamic power consumption generated by unused hardware. The survivability possibility of the application is lower for the SDR application because the DPR brings new failure modes to the system. Since they are not yet well characterized, they have to be cautiously handled. The online system test is experienced by the radar and the image processing application where every reconfigurable module has its own communication lines. Therefore, it is possible for the application to monitor the good operation of the functions and the reconfiguration process.

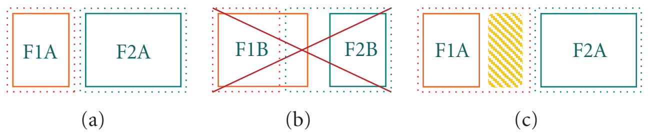

Figure 3: PRR reservation.

Most the evaluated features bring the promised advantages to the application. Nevertheless, a drawback comes from the size of the design. Indeed, where few virtualization is used, there is a significant size overhead when using DPR compared to the static implementation.

7.2. The hardware virtualization

Specific hardware virtualization tests are carried out in the SDR receiver application. The experiments compare implementations of crypto algorithms for an SDR application. The radio has two channels implemented and running at the same time, one requiring encryption and the other decryption. The algorithms can be changed dynamically and can be different for each channel. Three functions are considered, PLAIN, SCRAMBLER, and DES. Plain returns the clear message (no encryption), scrambler performs basic bit scrambling, and DES is a standard symmetric algorithm. The implementation is done with two PRR and a static region. Since the PRRs are statically defined, they need to be sufficiently big to fit the biggest module, and a significant amount of resources can be wasted. This problem is illustrated in Figure 3 for two PRRs in three situations. In Figure 3(a), the PRRs are dimensioned to hold F1A and F2A. When reconfiguring with F1B and F2B in Figure 3(b), the place left by F2B cannot be used by F1B. Therefore, the PRR reserved for F1 in Figure 3(c) needs to be maximal.

The resources consumption for the three modules and the static part of the design for an XC4VFX12 component is presented in Table 2. "The static + max PRR" is the DPR version of the design; the static part and two PRRs are implemented. In the "static + all modules" version, all modules are implemented statically. The resource gain shown in Table 2 is not very important when using DPR.

Table 2: Resource consumption.

	Plain		Scrambler		DES		Static	Static + max PRR		Static + all modules		Avail
	Encr	Decr	Encr	Decr	Encr	Decr						
Slice	53	53	69	69	379	379	2674	3432	63%	3676	67%	5472
RAMB 16	1	1	1	1	8	8	17	33	92%	37	103%	36
DSP48	0	0	0	0	0	0	0	0	0%	0	0%	32
PPC405	0	0	0	0	0	0	1	1	100%	1	100%	1
Bitstream (KB)	31,7	31,7	33,7	33,4	46,3	46,1	—	—	—	—	—	—

Table 3: Advantages evaluation.

	Task speed	Power reduction	Survivability	Mission change	Environment change	Algorithm change	Online system test	Virtualization
(a) Im. acq.		+1		(0)				+2
(b) Naval				0		+1		
(c) Im. proc.	0	+1	(0)	(0)	(+1)	(+1)	(+1)	+1
(d) Radar	0			+2	+2		+2	
(e) Modem	0	+1		+2	+2	+1		+2
(f) SDR Tx	+1	(+1)		+2	+2			+2
(g) SDR Rx	(+1)	(+1)	−2	(0)	+1	+1		+2

This is due to the choice of the algorithms implemented. Indeed, the plain and scrambler are very small compared to the DES that directs the size of the PRR. Nevertheless, even for that, the RAMB 16 consumption for the static design overuses the resources available in the component by 103%. Thus, without DPR, it cannot fit into the component. Furthermore, there are only three algorithms supported here. For applications with a reasonable level of flexibility as it is the case for SDR, many more algorithms need to be supported, like 3DES, AES, and so forth. For those applications, it is not possible to use a static design.

7.3. The ICAP performances

For industrial and security reasons, the bitstreams need to be encrypted. In the components used for the experiments, the ICAP is not usable when the bitstream encryption is enabled. The ICAP throughput is an issue for the designs using DPR in hard real-time applications, and it is carefully measured by the experiments. The ICAP was successfully tested at 100 MHz in 32- and 8-bit modes. The write and read-back modes were tested. Note that for reaching this speed, we found that the data needs to be sent on the falling edge of the ICAP clock. This is maybe due to a clock skew between the user logic and the ICAP. Table 4 summarizes the performances measured by connecting the ICAP to a basic GPIO, with the OPB-ICAP provided by Xilinx for Virtex II and with our reconfiguration manager. The partial bitstreams are stored in a DDR with a clock at 100 MHz. The Xilinx interface is always in 8-bit mode. The maximum theoretical bandwidth of the ICAP is 2.98 Gbit/s. It is not reached due to DMA overhead. Nevertheless, our custom OPB-ICAP is 84 times faster, compared to the vendor's module. 4 times are due to the data word size, 2.5 times due to frequency, and 8.4 times to the DMA accesses.

8. DISCUSSION

The development of applications for the experiments shows that designs using DPR must handle specificities like communication interfaces between modules, the execution scenarios, and the bitstream handling. Even if DPR has many open technology issues, all the applications tested can potentially gain from its use. For this, there are still many researches to do in order to bring DPR and dynamic HW at sufficient maturity level.

8.1. Communication interface

A high-level interface must be designed to handle the communication between the fixed part and the dynamic modules. This interface is similar to the one separating components in a static design but has to be uniform for all the modules targeted to be instantiated in a given PRR. It provides an abstraction of the accesses to dynamic modules allowing their use, as they were statically instantiated.

During the development of the experiments, the opportunity to use a dedicated interface for dynamic reconfiguration appears clearly. Nevertheless, regarding the number of application tested and their specific needs in terms of connections, latencies and throughputs, it was not possible to find out a solution to this problem. Therefore, the classical methodology for modeling and implementing interfaces was used.

The SDR application requires a communication transparency between the functional blocks of an application. This

Table 4: ICAP performances.

ICAPmode	GPIO	Xilinx OPB-ICAP	Custom OPB-ICAP	Theor.
8 bits	—	32.4 Mbit/s	0.56 Gbit/s	0.75 Gbit/s
32 bits	22.2 Mbit/s	—	2.8 Gbit/s	2.98 Gbit/s

requires the support of innovative middleware services for reconfigurable platforms.

8.2. *Execution management*

One of the main challenges when using dynamic reconfiguration is the management of the dynamic modules. This includes spatial and temporal management. Moreover, with hard real-time constraints, the bitstream is loaded from fast memory locations that have also to be managed.

In all the experiments carried out, the management is done at the operating system level. Nevertheless, since this solution adds substantial complexity to the operating system, the ideal solution would provide all those services transparently to the application.

The latency for loading a bitstream is important regarding the operating frequency of the component. A real-time application must have the control of the latencies to avoid undetermined behavior. Therefore, the reconfiguration management must be carried out by a scheduler controlled by the operating system of the application, which needs to access all the relevant parameters, like the reconfiguration latencies and status.

For preemptive scheduling, the context switch must be supported in the PRR. Therefore, the state hold in the PRR must be saved and restored.

8.3. *Bitstream handling*

The size of a complete bitstream reaches several megabytes for a high-end component, and for a partial bitstream, it is of the order of hundreds of kilobytes. For experiments from hard real-time applications, the reconfiguration speed can be a key factor for improving the performances by using DPR. This leads to use a high-performance reconfiguration manager using DMA to fetch the bitstream from a fast external memory at the maximum throughput. The configuration manager is controlled by a processor running the scheduler. This approach is used to release the pressure on the processor when transferring bitstream at word level. Moreover, a dedicated bus connecting memories release the processor bus allowing its use for other tasks. No other high-level abstraction for handling the bitstream is used by the experiments. This basic low-level solution has the disadvantage of mixing the bitstream handling with the application, which strongly complexifies the design.

8.4. *Technology issues*

There is today a single vendor providing large FPGA matrices supporting DPR. This causes several issues for its use in professional electronics applications. Upon the commercial and strategic ones, it lessens its use for safety applications. Indeed, a good level of safety is often reached by using redundant systems with the same specifications but designed by separate teams using different components coming from different vendors.

There is also no clear roadmap, from the vendor, regarding the future support of this technology in the upcoming high-end components. Therefore, this increases the risk of using it in real professional applications. Indeed, after their first release most of those applications must be maintained and upgraded during decades. On the contrary, consumer electronics products last no more than few years with barely any updates. The maintenance is performed to correct bugs, adapting the products with new client specifications or replacing old and damaged components. For this, the original (or equivalent) components need to be used to avoid going again into the heavy validation process. In this scope, the obsolescence and future compatibility of the components are a key point that needs to be addressed by the vendor.

The design flow provided is weak and experimental. It is not possible to model DPR during all the steps of an application development. SystemC can be used for first high-level steps but then it is difficult to use other tools, like for HW/SW partitioning, since DPR is not integrated by the tool vendors. For the low-level steps, there is a lack of behavioral and HW model, allowing to simulate and validate the designs before the platform integration into the final board. All those drawbacks are due to the novelty of this technology and to the fact that it brings the completely new concept of dynamic HW. Indeed, all the HW models used so far are static models, and none of the HDL languages used for description implement dynamic HW. Moreover, DPR is transverse to all abstraction layers of an application, from the reconfiguration port and wires at bit level to the scheduler at operating system level. Therefore, there are many research opportunities on dynamic HW at all abstraction levels. For its use in real applications, an IP library with components like reconfiguration manager, scheduler, and communication interface is necessary to abstract the additional complexity and minimize the increase of development efforts. Furthermore, since the applications become quite complex as the design effort for implementing and validate SoCs in large components, the most important challenge for the DPR is certainly its transparency. The designer should not worry about the reconfiguration details.

8.5. *Application benefits*

Many benefits can be taken from DPR in real professional electronics applications. Indeed, upon the criteria listed in Section 4 and directly evaluated by the experiments, DPR brings a new dimension of flexibility on the HW allowing its virtualization like with SW library. All experiments are taken from highly constrained applications, often integrated in complex systems. Therefore, the flexibility brought by the DPR enables to ease their development, integration, maintenance, or evolution. The use of very large components as high-end FPGA is enabled thanks to the system-on-chip approach based on IP assembling and reuse. In this scope, DPR brought the flexibility to modify IP dynamically.

The improved reconfiguration manager used by some experiments makes possible to use DPR directly in constrained hard real-time algorithm and applications. Nevertheless, using the maximum reconfiguration speed leads to a huge memory traffic causing internal and external bus congestion and therefore to power consumption. Some experiments mitigate bus congestion by using dedicated bus connecting the reconfiguration controller to memories. However, the power consumption is still important.

When the dynamic modules are changed too frequently, the power consumption increases and a time overhead appears because the PRR cannot be used during a reconfiguration process. This is the case when the reconfiguration is performed in a hot loop body of the algorithm. The most significant advantage pointed out by the experiments is the use of DPR for hardware virtualization when the reconfiguration is not part of the algorithm but is rather used to select the more suitable one in a given situation, leading to adaptive systems. The experiments show that adaptive algorithms are not frequent. Indeed, the algorithm designers are used to consider a static implementation. Here again, there is a need for a model of dynamic machine for algorithms designers.

Finally, the classical area, speed, and power metrics are not necessary improved when using DPR. The area is always improved when using DPR for virtualization where a significant part of dynamic HW resides in large- and low-cost external flash memory. Hence, the static power consumption is reduced compared to a static solution using a bigger component. However, the dynamic power consumption and speed improvements depend on the application.

9. CONCLUSION

During the last decade, DPR has already been widely studied as a research topic. However, it exists today very few real applications using it. This paper evaluates the use of DPR for real signal and image processing applications in professional electronics. An emphasis is put on the SDR that is the main target application for DPR as announced by Xilinx. Moreover, it provides relevant feedback in order to improve its applicability. For this, it is based on a set of seven real applications in signal and image processing with experiments carried out under real conditions. As the design flow is a key point for professional electronics applications, it makes a precise analysis of the current flow available and highlights its missing elements. It shows how SystemC can be used to supplement the first steps of the flow. Those steps are crucial for the modeling and validations of the SDR application.

Upon the classical area, speed, and power metrics, professional electronics applications can also benefit from DPR for online system test, mission/environment changes, algorithms adaptive changes, survivability, and hardware virtualization. The paper provides research directions to improve the applicability of DPR, the modeling of dynamic hardware, the definition of a generic reconfiguration interface, the state handling with preemptive scheduling, and the transparent bitstream handling by the application.

Finally, it shows how the reconfiguration interface provided by the vendor can be improved by using DMA and resolving a clock jitter issue. Thanks to those optimizations; it is possible to achieve a reconfiguration speed of 2.8 Gbits/sec close to the theoretical limit.

ACKNOWLEDGMENTS

This research is part of the RECOPS project. It is a EUROPA contract from the European Defense Agency, funded by French and Italian Ministries of Defense and the Walloon Region in Belgium. B. Rousseau holds an F.R.S.-FNRS fellowship (Belgian Fund for Scientific Research).

REFERENCES

[1] K. De Bosschere, W. Luk, X. Martorell, et al., "High-performance embedded architecture and compilation roadmap," in *Transactions on High-Performance Embedded Architectures and Compilers I*, vol. 4050 of *Lecture Notes in Computer Science*, pp. 5–29, Springer, Berlin, Germany, 2007.

[2] M. J. Flynn and P. Hung, "Microprocessor design issues: thoughts on the road ahead," *IEEE Micro*, vol. 25, no. 3, pp. 16–31, 2005.

[3] J. Song, T. Shepherd, M. Chau, et al., "A low power open multimedia application platform for 3G wireless," in *Proceedings of IEEE International Symposium on Systems-on Chip (SOC '03)*, pp. 377–380, Tampere, Finland, November 2003.

[4] M. Hammes, C. Kranz, D. Seippel, J. Kissing, and A. Leyk, "Evolution on SoC integration: GSM baseband-radio in 0.13 μm CMOS extended by fully integrated power management unit," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 236–245, 2008.

[5] A. C. Tribble, "The software defined radio: fact and fiction," in *Proceedings of IEEE Radio and Wireless Symposium*, pp. 5–8, Orlando, Fla, USA, January 2008.

[6] http://www.xilinx.com/prs_rls/dsp/0626_sdr.htm.

[7] J.-P. Delahaye, J. Palicot, C. Moy, and P. Leray, "Partial reconfiguration of FPGAs for dynamical reconfiguration of a software radio platform," in *Proceedings of the 16th IST Mobile and Wireless Communications Summit*, pp. 1–5, Budapest, Hungary, July 2007.

[8] C. Kao, "Benefits of partial reconfiguration," *Xilinx Xcell Journal*, vol. 2005, no. 55, pp. 65–67, 2005.

[9] N. McKay, T. Melham, and K. W. Susanto, "Dynamic specialisation of XC6200 FPGAs by partial evaluation," in *Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 308–309, Napa Valley, Calif, USA, April 1998.

[10] P. Merino, M. Jacome, and J. C. Lopez, "A methodology for task based partitioning and scheduling of dynamically reconfigurable systems," in *Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 324–325, Napa Valley, Calif, USA, April 1998.

[11] S. R. Park and W. Burleson, “Reconfiguration for power saving in real-time motion estimation,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’98)*, vol. 5, pp. 3037–3040, Seattler, Wash, USA, May 1998.

[12] J. Becker, A. Donlin, and M. Hübner, “New tool support and architectures in adaptive reconfigurable computing,” in *Proceedings of IFIP International Conference on Very Large Scale Integration (VLSI-SoC ’07)*, pp. 134–139, Atlanta, Ga, USA, October 2007.

[13] M. Alderighi, F. Casini, S. D’Angelo, M. Mancini, S. Pastore, and G. R. Sechi, “Evaluation of single event upset mitigation schemes for SRAM based FPGAs using the FLIPPER fault injection platform,” in *Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT ’07)*, pp. 105–113, Rome, Italy, September 2007.

[14] H. Wang, J.-P. Delahaye, P. Leray, and J. Palicot, “Managing dynamic reconfiguration on MIMO Decoder,” in *Proceedings of the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS ’07)*, pp. 1–8, Long Beach, Calif, USA, March 2007.

[15] P. Sedcole, B. Blodget, J. Anderson, P. Lysaghi, and T. Becker, “Modular partial reconfigurable in Virtex FPGAs,” in *Proceedings of International Conference on Field Programmable Logic and Applications (FPL ’05)*, pp. 211–216, Tampere, Finland, August 2005.

[16] K. Wu and J. Madsen, “Run-time dynamic reconfiguration: a reality check based on FPGA architectures from Xilinx,” in *Proceedings of the 23rd IEEE Norchip Conference (NORCHP ’05)*, pp. 192–195, Oulu, Finland, November 2005.

[17] T. Pionteck, C. Albrecht, and R. Koch, “A dynamically reconfigurable packet-switched network-on-chip,” in *Proceedings of the Design, Automation and Test in Europe Conference (DATE ’06)*, vol. 1, pp. 1–4, Munich, Germany, March 2006.

[18] C. Claus, F. H. Müller, J. Zeppenfeld, and W. Stechele, “A new framework to accelerate Virtex-II Pro dynamic partial self-reconfiguration,” in *Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS ’07)*, pp. 1–7, Long Beach, Calif, USA, March 2007.

[19] P. Manet, D. Maufroid, L. Tosi, et al., “RECOPS: reconfiguring programmable devices for military hardware electronics,” in *Proceedings of the Design, Automation and Test in Europe Conference (DATE ’07)*, pp. 1–6, Nice, France, April 2007.

[20] RTCA DO-254/EUROCAE ED-80, “Design assurance guidance for airborne electronic hardware,” April 2000.

[21] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, “Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAs,” in *Proceedings of International Conference on Field Programmable Logic and Applications (FPL ’06)*, pp. 1–6, Madrid, Spain, August 2006.

[22] http://www.xilinx.com/ise/.

[23] Xilinx XAPP290, “Two flows for partial reconfiguration: module based or difference based,” September 2004.

[24] http://www.xilinx.com/ise/optional_prod/cspro.htm.

[25] F. Ghenassia, *Transaction-Level Modeling with SystemC*, Springer, New York, NY, USA, 2005.

[26] Y. Qu, K. Tiensyrja, and J.-P. Soininen, “SystemC-based design methodology for reconfigurable system-on-chip,” in *Proceedings of the 8th Euromicro Conference on Digital System Design (DSD ’05)*, pp. 364–371, Porto, Portugal, August-September 2005.

[27] A. Schallenberg, W. Nebel, and F. Oppenheimer, “OSSS+R: modelling and simulating self-reconfigurable systems,” in *Proceedings of the 16th International Conference on Field Programmable Logic and Applications (FPL ’06)*, pp. 1–6, Madrid, Spain, August 2006.

[28] A. Herrholz, F. Oppenheimer, P. A. Hartmann, et al., “The ANDRES project: analysis and design of run-time reconfigurable, heterogeneous systems,” in *Proceedings of the 17th International Conference on Field Programmable Logic and Applications (FPL ’07)*, pp. 396–401, Amsterdam, The Netherlands, 2007.

[29] G. Gailliard, E. Nicollet, M. Sarlotte, and F. Verdier, “Transaction level modelling of SCA compliant software defined radio waveforms and platforms PIM/PSM,” in *Proceedings of the Design, Automation and Test in Europe Conference (DATE ’07)*, pp. 1–6, Nice, France, April 2007.

[30] G. Gailliard, B. Mercier, M. Sarlotte, B. Candaele, and F. Verdier, “Towards a systemC TLM based methodology for platform design and IP reuse: application to software defined radio,” in *Proceedings of the 2nd International European Workshop on Reconfigurable Communication-Centric Systems-on (RECOSOC ’06)*, pp. 131–138, Montpellier, France, July 2006.

[31] M. Sarlotte, B. Counil, P. Gelineau, R. Chau, and D. Maufroid, “Partial reconfiguration concept in a SCA approach,” in *Software Defined Radio Technical Conference and Product Exposition (SDR ’07)*, Denver, Colo, USA, November 2007.