DESIGNING A RECONFIGURABLE PROCESSING DATAPATH FOR SDR OVER HETEROGENEOUS RECONFIGURABLE PLATFORMS

Jean-Philippe DELAHAYE[1], Pierre LERAY[2], Loïg GODARD[2], Amor NAFKHA[2], Christophe MOY[2]
[1]DGA/CELAR, French MoD, Bruz, France
jean-philippe.delahaye@dga.defense.gouv.fr
[2]IETR/Supelec-SCEE Laboratory, Campus of Rennes, France
{pierre.leray, loig.godard, amor.nafkha, christophe.moy}@supelec.fr

ABSTRACT

The implementation of a dataflow application on a heterogeneous platform has to tackle with several design issues and should take into account many design trade-offs. One of the design issues is the connectivity between processing functionalities due to the variety of communication interfaces in a heterogeneous system. To overcome this issue system designers have often to choose suitable abstraction level software mechanisms to ensure communication between different DSPs/FPGAs, thus introducing some protocol overhead decreasing the application performance. On the contrary, the implementation of the processing datapath architecture we propose here is at a low level of abstraction and offers a lightweight approach for designing flexible baseband applications. It particularly suits to heterogeneous designs involving FPGAs being dynamically reconfigured in the context of Software Defined Radio - SDR - systems.

1. INTRODUCTION

Software Defined Radio (SDR) is expected to be the most appropriate answer to future multi-standards handsets design challenges [1]. We can also predict that for a long time SDR systems will be heterogeneous in terms of computing resources, in order to deal with a wide variety of radio applications and with the increasing throughput needs due to the digital communication improvements. This implies many research activities in the fields of multi-processing and heterogeneous computing. All the more so as dynamic reconfiguration is involved. But even if solutions exist for DSP [2] where it is more natural to support reconfiguration, it is less common in the field of FPGA. In our previous work [3] we investigate a combined approach, from the application side to the implementation on the platform, while modeling the architecture of SDR heterogeneous platforms and detailing the multi-standards applications needs in terms of reconfigurability. The analysis of the reconfiguration requirements is detailed in [4] for multi-standards physical layer applications based on the GSM, UMTS/UTRA/FDD and 802.11g OFDM standards. This permitted to extract from these studies different use cases of reconfiguration corresponding to different granularity levels. A "Hierarchical Distributed Configuration Management" (HDMC) [4] mode has been proposed to control the multi-granularity levels of reconfiguration over heterogeneous resources. This HDMC framework manages the configuration of the various baseband functionalities building the processing datapath.
This paper concerns the design approach of dataflow-oriented applications implemented on heterogeneous processing resources made of GPPs, DSPs and FPGAs.

The next section of this paper overviews the hardware of the chosen SDR system architecture and presents the implementation of the configuration management model. Section 3 introduces the main concepts for the design of a datapath oriented application on a heterogeneous platform. Section 4 presents the dynamic partial reconfiguration and its use in datapath oriented applications. A case study of a reconfigurable FIR filtering function is presented in section 5 as an example of implementation.

2. SYSTEM ARCHITECTURE

2.1. Heterogeneous Hardware SDR platform

This section overviews the system architecture. Details are especially given on the FPGA architecture, which offers a large range of design possibilities. The paper focuses on architectural solutions that could be adopted to achieve partial reconfiguration (PR) and enhance the flexibility of the application. We present other architectural solutions in [5] to get the PR capability manageable on hardware platform which belongs to the E²R EU project.
Fig. 1 shows the SDR platform architecture which is mainly comprising three hardware modules, the GPP, the DSP and the FPGA successively described below.
The **GPP** and the external storage memories are resources used from a standard PC station. The GPP is the host of the

platform. It is responsible of initializing the platform resources, loading the DSP code and the initial FPGA full configuration. The control-oriented functions of each hardware module are started during the initialization phase and afterwards will take over the control of their module at run-time.

The **DSP** is a C64 from TI. The DSP executes sub-parts of the processing datapath baseband modulation coding functions. The DSP local data memory stores bitstream files and the DSP is able to send directly onto configuration bus of the FPGA these bitstreams. The DSP is also able to fully reconfigure the FPGA by the external SelectMAP interface. It will be shown later that another way is chosen here (self-reconfiguration).

of Partial Reconfigurable Modules (PRM). The PRMs represent the different contexts of the reconfigurable functions.
The partial reconfigurability is a key feature to enhance the flexibility of applications. The PR allows to select the part of FPGA to reconfigure rather than to fully reconfigure the device each time a sub-part of the FPGA has to be changed.
One of the architectural solutions to allow the use of the partial reconfiguration is to implement a configuration controller, for instance an embedded processor (PPC, µBlaze), linked to the internal configuration access port (ICAP) available from the Virtex-II FPGA series. This is the solution adopted in our SDR platform experimentation and illustrated in the system architecture shown in Fig. 1. This architectural solution allows the FPGA to perform self-reconfiguration. Self or auto-reconfiguration means that the

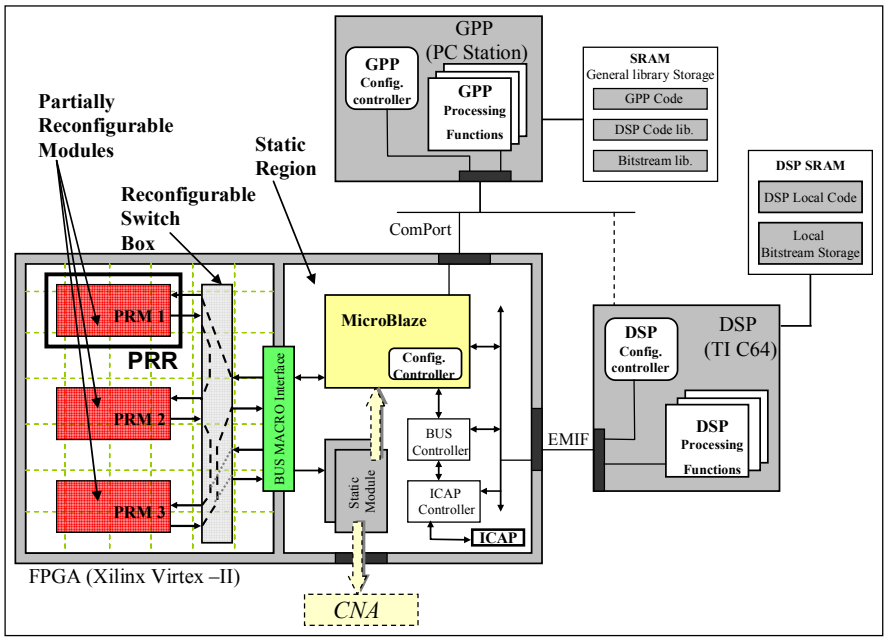

Fig. 1 : The Heterogeneous Reconfigurable SDR Platform Architecture

The **FPGA** of our platform is a Virtex-II device from Xilinx. The FPGA is involved on the platform as follows. The static region is dedicated to the fixed modules for fixed and parameterizable processing functions. The fixed part also contains the modules responsible of the control, as the embedded processor, the configuration interface and the data buses of the processing datapath linked to the GPP and the DSP. The second area of the FPGA is dedicated to the dynamically reconfigurable modules. These modules are the processing functions containing parts which will be potentially reconfigurable to answer the application context switching needs. The dynamically reconfigurable parts are located into Partially Reconfigurable Regions (PRR) in form

reconfigured areas of the FPGA (PR Regions) are managed by the FPGA itself. This latter could be much simpler, as a state machine, but the µBlaze processor will also be responsible of other managing tasks.
The details about the partial reconfiguration features are presented in [5]. We also achieve the PR on the Virtex-4 in the same manner as on the Virtex-II. The improvements between Xilinx Virtex-II PR and V-4 PR are also described in [5].

Components Library: The hardware and software designs of the processing functions are stored in the external storage memory of the platform where they can be stored and extracted by the general configuration management of the system.

2.2. Software Management

The Hierarchical and Distributed Configuration Management (HDCM) framework has been deployed on the hardware platform presented above. The HDMC Model detailed in [4] is composed of several Configuration Management Units (CMU). The CMUs are responsible of the configuration management of the reconfigurable datapath which supports a multi-standard baseband application. The CMUs are linked together in a hierarchical structure made up of three levels. It provides an abstraction to handle the configuration data. The HDCM model is implemented on the SDR hardware platform with the following deployment. The general management unit called L1_CM runs on the GPP, indeed the host of the platform. This L1_CM is responsible for the management by parameters of the whole baseband processing. The configuration orders are sent by the L1_CM to the L2_CMUs located on the different hardware modules of the platform. L2_CMUs are responsible of the distribution of the configuration parameters of all the processing functions running on a hardware module. The L2_CMU implemented in the FPGA module runs on the embedded processor (µBlaze). At the third level, the L3_CMUs are associated to the processing functions. Each L3_CMU manages the configuration of the processing component depending on its implementation. Our platform is composed in this example of one L1_CM, three L2_CMUs and several L3_CMUs depending on the number of parameterizable or reconfigurable processing functions implemented in the processing datapath.

3. PROCESSING DATAPATH DESIGN APPROACH

3.1 Principles

It is widely admitted that component modeling methodologies are suitable approaches to design software and also baseband processing applications in the software defined radio systems, as described in [6]. These methodologies allow to define self-dependant functionalities which are reusable, portable and interoperable. The top part of Fig. 2 illustrates the mapping of datapath oriented applications over a heterogeneous hardware platform. This mapping is a key issue when considering the needs of portability, and interoperability. The connectivity and the synchronization of the reconfigurable components are among the issues that SDR designers have to cope with due to the different link interfaces, signal clocks involved on the platforms with DSP, FPGA, GPP. In order to answer these issues, hardware abstraction layers, which aims at providing uniform services over heterogeneous platform, are introduced in software radio systems. Standardization efforts in SDR, overviewed in [7], are particularly strong in the middleware modeling for the improvement of portability, interoperability and ease of the integration of software radio components of different resources.
The component modeling with standard interfaces, as the component example shown in the bottom part of the Fig. 2, aims at avoiding the connectivity issues. This object oriented representation, coming from the work done in the EU CAST project [8], is designed using Java language to obtain an abstraction layer of the reconfigurable processing chain to enable the reconfiguration management at a high level of abstraction.

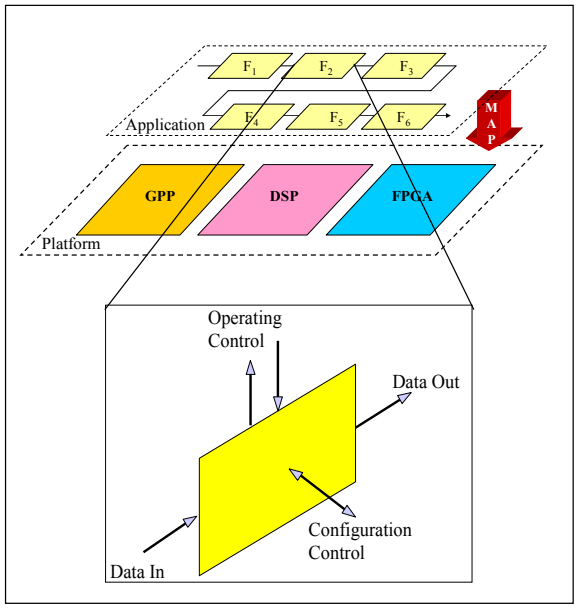

Fig. 2: Dataflow Application over Heterogeneous Hardware

We detail in the next section a component which encapsulates the baseband processing function to provide standard interfaces. Even at a low level of abstraction the interfaces of our component are implemented the model depicted in and override the differences in clock signals and link interfaces of a heterogeneous platform.
The datapath designed with these reconfigurable components is associated with the HDCM configuration management datapath briefly described in section 2.2 and allows to reconfigure dynamically the baseband functionalities to answer external service switches requests.

3.2. Processing Component Encapsulation

The component encapsulation adapter designed for a heterogeneous hardware platform should provide the same behavior for any implementation of functions in hardware on a FPGA or in software on a DSP. This means that the component adapter should provide standard external interfaces independently of any processing functionality that

is encapsulated in it. The external interfaces of the component adapter have to ensure the connectivity between the different HW and SW implementations of processing functionalities to form an entire processing chain.
We propose a component adapter illustrated in Fig. 3. In this component, every external interfaces work asynchronously with handshake signals. The request/acknowledge mechanism ensures the same behavior for any component implementation. Inside the component adapter, the processing function runs synchronously, which allows to encapsulate any of existing IP previously designed.

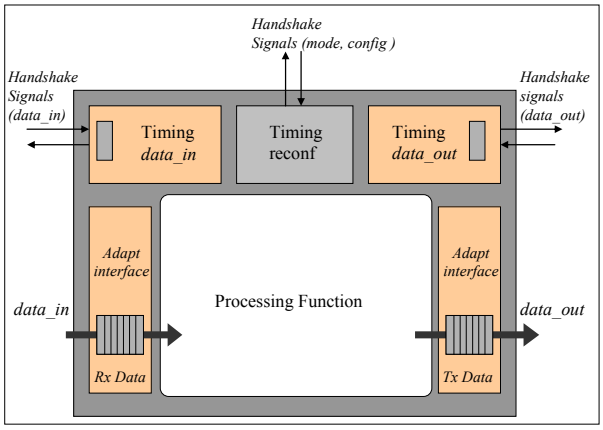

Fig. 3 : Processing Function Component Encapsulation

The component should provide the following interfaces. The "Timing data_out" and "Timing data_in" interfaces are both responsible of controlling the availability of data respectively to produce and consume data. Internally these interfaces generate the enable signal for the processing function.
The Rx (respectively Tx) data interfaces receive the data to process (and respectively give the processed data) in FIFO-like buffers.
Finally, the "Timing reconfig" interface provides the processing mode (idle, init, run) and the configuration signals are used to maintain the data in the data transmission interfaces during the reconfiguration process.
A component encapsulates a baseband function and the instantiated components are connected together to form the complete datapath. The data rate in the datapath is auto-regulated by the asynchronous data transfer mechanism. The application, made up of several components, has a "GALS" (Globally Asynchronous Locally Synchronous) behavior. There are two cases of component connections: two processing components are linked together by physical busses in the case of HW/HW or HW/SW data transmission or shared memory in the case of SW/SW components connection. The component implementations are detailed in the following section.

3.3. Component implementation

The component described in the previous part has different implementations as it aims at building up the processing datapath at a low level of abstraction. So the component has two implementations developed either in C or VHDL depending on the target, respectively DSP or FPGA.

```
/* GALS function header in C*/                          Software function
#include <stdarg.h>

fct(int mode, int * flagMail_in, int * data_in, int * flagMail_out, int * data_out, ...);
```

```
-- GALS function declaration of the entity in VHDL     Hardware function
entity ip_fct is
  Port (
      clk : in std_logic;
      reset: in std_logic;
      --
      din : in std_logic_vector(31 downto 0);
      flagMail_in : in std_logic _vector(Nb_datain_mailbox downto 0);
      reset_flagMail_in : out std_logic;
      dout : out std_logic_vector(31 downto 0);
      flagMail_out: out std_logic_vector(Nb_dataout_mailbox downto 0);
      reset_flagMail_out : in std_logic
);
end ip_fct;
```

Fig. 4: Component Declaration for Hardware and Software Implementations.

The interfaces are identical for both kinds of implementation. The arguments of the C function and the signals of the VHDL component represented in Fig. 4 are the following:

- **Mode** : the mode argument allows to control the processing mode of the function (init, run, bypass are the main modes)
- **flagMail_in** allows to control the availability of new data on the input interface.
- **Data_in** is the data address where the input data are available.
  - In the case of a software function, this argument is a pointer on a memory location (for SW to SW components data transmission) or an address of a physical bus (for the HW to SW components data transmission).
  - In the case of a hardware entity, the signal is the input data bus of the component.
- **flagMail_out** allows to control data presence on the output. The synchronous processing function will not be (re)-started if the output storage memory is full.
- **Data_out** is the same as data_in for the output data of the component.

4. FPGA PARTIAL RECONFIGURATION DESIGN

4.1 Reconfiguration of Functionality.

The partial reconfigurability allows changing selectively segments of a FPGA functionality without suspending operations of the remaining part. There are several benefits for partial reconfiguration (PR): it reduces the configuration time and saves storage memory as the partial reconfiguration files (bitstreams) are smaller than full ones. It may also limit the overhead and the bandwidth spending for code downloading in case of OTAR.

The recent improvements in standard design flows from Xilinx [9] and the use of PlanAhead tool for PR allow more design opportunities. This permits to take benefit as much as possible of the flexibility offered by dynamically reconfigurable FPGA, as illustrated in section 5 with the example of the FIR.

4.2. Reconfigurable Hardware component interconnection.

Different kinds of communication data link architectures are possible for the interconnection of reconfigurable IP inside the FPGA. The bus link is the more flexible and allows easily to connect/disconnect IPs on it. The direct connectivity of a ring link architecture best suits dataflow oriented applications in terms of performance but limits the flexibility of a reconfigurable datapath compared to the bus link. To increase the flexibility of the point-to-point datapath architecture, we propose to introduce the concept of a Reconfigurable Switch Box (RSB) that has different configurations to directly connect IPs between each other. This solution benefits of the point-to-point connectivity and increases the flexibility of the datapath. Two different configurations of the RSB are illustrated in Fig. 5. The main drawback of this solution is the design effort as it adds one more reconfigurable block in the FPGA.

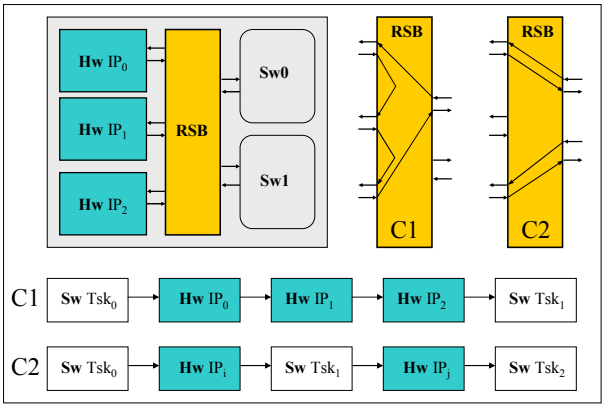

Fig. 5 : HW/SW Function Reconfiguration

5. IMPLANTATION EXAMPLES OF FIR PROCESSING MODULE

We describe in [5] several implementations of partially reconfigurable processing modules using the partial reconfiguration capability of the Virtex-4. We propose in [5] various design solutions to perform partial reconfiguration of processing functions on FPGA, like the convolutional coding, the constellation mapping or the FIR filtering. The partial configuration feature of the FPGA is used to modify either structural elements or some parameters in the processing module architecture. In the next paragraph, we detail three different types of partial reconfiguration implementations applied on FIR IP.

The same FIR IP design has been reused for all implementations. The FIR IP is based on a semi-parallel FIR filter architecture with a four-multipliers MAC structure. Several FIR functionalities have been implemented, up to a 32-taps FIR filter. The complexity for the FIR filter implementations is 325 slices (58% for the FSM control part, and 42% for the processing part). The FIR IP with this four-multipliers MAC has been used to implement the three kinds of partially reconfigurable IPs. The processing part is reused in each implementation of a FIR as a common IP.

Depending on the nature of the processing module, the PR region (red rectangle in Fig. 6) may support different parts of a module. The three cases illustrated in Fig. 6 correspond to different implementations of the FIR:

- For the full reconfigurable IP (Fig.6.1): the full structure of the FIR is reconfigured.
- For the semi reconfigurable IP (Fig. 6.2): Only the FSM of the FIR that control the operative part is reconfigured.
- For the IP with the reconfigurable macro (Fig. 6.3): The set of coefficients is located in the PR Region which refers to the case of a FIR whose coefficients must be changeable. FIR coefficients are stored in FPGA memory blocks configured as ROM and the ROM blocks are reconfigured to change the coefficients of the FIR. Consequences are more important than expected at a first glance. This approach prevents from designing and consuming communication resources to access several sets of coefficients: this includes address and data buses for RAM access. ROM is directly reconfigured through the configuration plane.

This FIR example shows that even concerning a very common functionality, many considerations can be taken into account, and consequently a plurality of implementations may be derived.

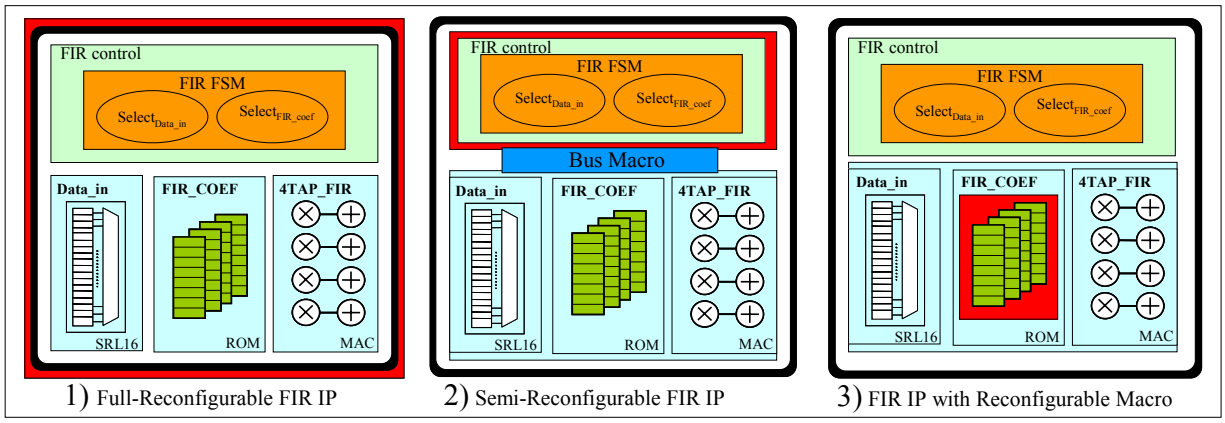

Fig. 6 : Partially Reconfigurable FIR Filter Implementations

Table 1 sums up the different kinds of implementation of the FIR IP. The choice of one solution versus the others is depending on the requirements of the application in terms of reconfigurability.

Reconfigurable IP	**Bitstream Size**	**Reconfiguration Time**
FPGA Full-reconfiguration	830kB	16 ms
Full-reconfigurable FIR	74,7 kB	1,5 ms
Semi-Reconfigurable	25,7 kB	510 µs
Reconfigurable Macro	2 kB	40 µs

Tab.1. Bitstreams Size and Reconfiguration Time on a Virtex-II 2000

6. CONCLUSION

Currently, standard programming interfaces are generally proposed at a high level over the hardware abstraction software. The component design using standard programming interfaces are designed at a high level of granularity which could limit the flexibility of the application.

Our approach of designing a processing datapath with a standard reconfigurable component adapter for each baseband function allows to rapidly implementing communication chains over a heterogeneous platform and increases the flexibility of the application. The proposed processing datapath associated with a configuration management HDCM structure allows to abstract the configuration orders, mechanisms and demonstrates the feasibility of a flexible SDR system with lightweight software.

7. REFERENCES

[1] J. Mitola, "The software Radio architecture," *IEEE Comms Mag*, vol. 33, no. 5, pp. 26--38, May 1995.

[2] C. Moy, A. Kountouris, A. Bisiaux, "HW and SW Architectures for Over-The-Air Dynamic Reconfiguration by Software Download," SDR Workshop of the IEEE Radio and Wireless Conference, Boston, USA, Aug. 2003

[3] J.-P. Delahaye, J. Palicot, P. Leray, "Managing Dynamic Partial Reconfiguration on Heterogeneous SDR Plaforms" SDR Forum Technical Conference 2005, Los Angeles, Nov. 2005.

[4] J.-P. Delahaye, J. Palicot, P. Leray, "A Hierarchical Modeling Approach in Software Defined Radio System Design," SIPS 2005, Athens-Greece, Nov. 2005.

[5] J.-P. Delahaye, C. Moy, P.Leray, J. Palicot, "Partial Reconfiguration of FPGAs for Dynamical Reconfiguration of a Software Radio Platform", IST Mobile SUMMIT 2007.

[6] A. Kountouris, C. Moy, "Reconfiguration in Software Radio System", 2nd Workshop on Software Radio, (Karlsruhe, Germany), March 2002.

[7] M. Scoville, S. Berger, R. C. Reinhart, J. E. Smith, "The Software-Defined Radio & Cognitive Radio Inter-Consortia Affiliation", MILCOM conference 2006.

[8] IST-CAST EU project, "Configurable Radio with Advance Software Technology", 2002.

[9] Xilinx Inc, "Early Access Partial Reconfiguration User Guide" UG 2006.