Partial and Dynamic reconfiguration of FPGAs: a top down design methodology for an automatic implementation

Florent Berthelot[1], Fabienne Nouvel, and Dominique Houzet

[1]CNRS UMR 6164 IETR-INSA
Rennes 20 av des Buttes de Coesmes
35043 Rennes, France
{florent.berthelot, fabienne.nouvel,dominique.houzet}@insa-rennes.fr

Abstract

Dynamic reconfiguration of FPGAs enables systems to adapt to changing demands. This paper concentrates on how to take into account specificities of partially reconfigurable components during the high level Adequation Algorithm Architecture process. We present a method which generates automatically the design for both partially and fixed parts of FPGAs. The runtime reconfiguration manager which monitors dynamic reconfigurations, uses prefetching technic to minimize reconfiguration latency of runtime reconfiguration. We demonstrate the benefits of this approach through the design of a dynamic reconfigurable MC-CDMA transmitter implemented on a Xilinx Virtex2.

1. Introduction

The recent and next generations wireless systems are being designed to provide a wide variety of multimedia services and seamlessly switch between different wireless standards. Multiple physical layers algorithms must be implemented, such as for example coding, decoder, equalizer, ...The need for Software Defined Radio (SDR) [4] is motivated by the wide range of configuration parameters and flexibility and leads to the concept of reconfigurability. Harvard-based architectures (DSPs) are reconfigurable at system-level and use a temporal scheme implementation of operations but suffers from its power efficiency for repetitive and high throughput computations. Reconfigurable devices, including FPGAs, can fill the gap between hardwired and software technology. Recently runtime reconfiguration (RTR) of FPGA parts has led to the concept of virtual hardware. By changing dynamically the functionality performed by the FPGA over the time, we can address SDR constraints and obtain a scalable system. However reconfiguration latency is a major drawback of runtime reconfiguration on commercial devices and must be considered during HW/SW partitioning process. The objective is to obtain a near optimal scheduling of tasks in time over an heterogeneous architecture [5].

In this paper, we briefly introduce the different reconfiguration levels and describe the impact of runtime reconfigurable component utilization on algorithm-architecture adequation in the second section. Next the way of modeling a partially runtime reconfigurable part of a FPGA with SynDEx is exposed. We discuss on how to generate an automatic management of runtime reconfiguration over the time with SynDEx [6]. A case study based on a runtime reconfigurable MC-CDMA transmitter is presented in last section followed by the conclusion.

2. Reconfiguration Levels

In the case of mobile communications, three main constraints have to be combined : high performance, low power consumption and flexibility. Grain of computation, reconfiguration schemes are open research topics. Three levels of reconfiguration can be considered :

In the System Level Reconfiguration case, the application is very often supported by an heterogeneous architecture (DSPs, FPGAs). The functionalities are distributed onto these components according to their

performances. Either, it is difficult to address a specific function of the application.

With Functional Level Reconfiguration, most of the reconfigurable architectures are based on FPGAs which support partial dynamic runtime configuration and allow flexible hardware. A runtime reconfiguration manager will control, monitor and execute the dynamic reconfiguration.

In the Logic and RTL level Reconfiguration case, the majority of the FPGAs are fine grain since they can be configured at the bit level. Neither, this level is architecture's manufacturer dependant and do not allow the designer to adopt an open methodology.

Considering the SDR constraints, the functional reconfiguration level seems to be the best level for our partial and dynamically reconfiguration of systems.

3. Design flow for dynamic reconfiguration : AAA approach

Some partitioning methodologies based on various approaches are reported in the literature [2]. They are characterized by the granularity level of the partitioning, metrics, target hardware, support of runtime reconfiguration, flow automation and on-line / off-line scheduling policies. Choice of candidates for a dynamic hardware implementation can be guided by some metrics: execution time, memory constraints, power efficiency, reconfiguration time, configuration prefetching capabilities and area constraints.

Among theses methods we have chosen the AAA approach with its tool SynDEx. AAA methodology aims at finding the best matching between an algorithm and an architecture while satisfying time constraints. The reader is refereed to the previous paper [1] which describes all the steps of the methodology and extensions to FPGAs modelization.

Application algorithm is represented by a data flow graph to exhibit the potential parallelism between operations. An operation is executed as soon as its input are available, and is infinitely repeated.

Architecture is also modeled by a graph where the vertices are operators (e.g processors, DSP, FPGA) or media and edges are connections between them. Operators have no internal parallelism computation available but the architecture exhibits the potential parallelism.

Adequation consists in performing the mapping and scheduling of the operations and data transfers onto the operators and the communication media. It is carried out by a heuristic which takes into account durations of computations and inter-component communications.

The result is a synchronized executive represented by a macro-code for each vertices of the architecture.

4. Run time reconfiguration considerations for adequacy

To reduce the difficulty in managing such dynamic reconfigurable application and to provide reliable implementation, following issues must be adressed : automatic or manual partitioning of an application, specification of the dynamic constraints, automatic generation of the C or VHDL core controler.

Considering these features in SynDEx, runtime reconfigurable parts of an component must be considered as vertices in the architecture graph. As shown in the example of Figure 1, runtime reconfigurable parts of a FPGA (D1 and D2) and fixed parts (F1) can be represented as hardware operators of the architecture. (D1 and D2) will integrate both dynamic modules and the control to manage the reconfiguration. An internal media (IL) allows data exchanges between those parts.
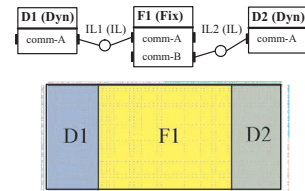

Figure 1. Model of runtime reconfigurable parts of a FPGA with SynDEx

By identifying dynamic parts of the application at a high level, we make dynamic specification flexible and independent of the application coding style. A constraints file will contain the definition of each dynamic module and the associated constraints (loading, unloading, sharing area, dynamic relations, exclusion).

5. Automatic design generation : General FPGA synthesis scheme

Once mapping and scheduling of the algorithm are performed, macro-code is automatically generated and each one must be translated. The translation generates the VHDL code, both for the static and dynamic parts of a FPGA. The final FPGA design is based on several dedicated processes to control:

- communication sequencings,
- computation sequencings,
- operator behaviour,
- activation of reading and writing phases of buffers,

In order to perform reconfiguration of the dynamic part we have chosen to divide this process into two subparts: a configuration manager and a protocol configuration builder. A configuration manager is in charge of the configuration bitstream which must be loaded on the reconfigurable part by sending configuration requests. Configuration requests are sent to the protocol configuration builder which is in charge to construct a valid reconfiguration stream in agreement with the used protocol mode (e.g selectmap). Figure 2 shows different solutions of architectures to reconfigure partially a FPGA.
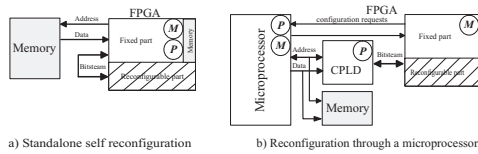

a) Standalone self reconfiguration   b) Reconfiguration through a microprocessor

Figure 2. Different ways to reconfigure dynamic parts of a FPGA

Labels *M* and *P* show where functionalities 'Configuration manager' and 'Protocol configuration builder' respectively are implemented. Locations of these functionalities have a direct impact on the reconfiguration latency. Case a) shows standalone self reconfigurations where the fixed part of the FPGA reconfigures the dynamic area. Case b) shows the used of a processor to perform the reconfiguration. In this case the FPGA sends reconfiguration requests to the processor through hardware interrupts.

The outputs of SynDex tool are both a VHDL entities and process corresponding to dynamic and static modules for final implementation. We synthesize the VHDL code of the static part and of each dynamic part separately in order to obtain separate netlists. The Xilinx Modular back-end flow is used to place and route each module and to generate the associated bitstream, resulting in a typical floorplan. Concerning the place and route constraints, reconfigurable modules have the following properties : the height of the module is always the full height of the device and its width ranges a minimal of four slices. The communications between static and dynamic parts use a special bus macro. This bus is a fixed routing bridge between two sides and

is pre-routed. The current implementation of the bus macro uses eight 3-state buffers, their position exactly straddles the dividing line between designs. All these constrains are fixed in a constraints file, used during the placement and routing. By using SynDEx tool and Xilinx Modular Design flow, we define a top-down and validated methodology, depicted by Figure 3, addressing the complete design flow.
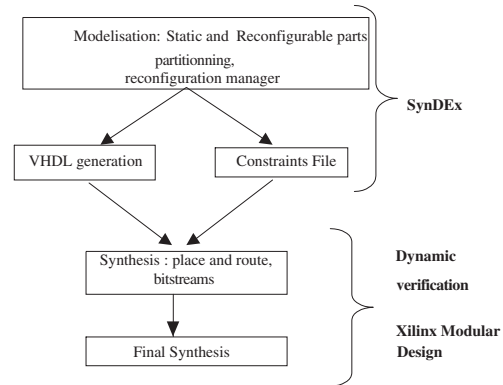

Figure 3. Complete Design Flow : SynDEx tool and Modular Design

6. Implementation example

Our top-down design flow has been tested on a complete application which is a transmitter system for future wireless networks for 4G air interface [3]. This transmitter is based on MC-CDMA modulation scheme. SynDex algorithm graph, depicted by Figure **??**, shows the basic numeric computation blocks of this transmitter. Block *modulation* performs either a QPSK or QAM-16 modulation. This adaptive modulation is selected by the conditional entry *Select* which defines the modulation of each OFDM symbol according to the signal to noise ratio. All other blocks are implemented in the static part of the FPGA.

We implement this transmitter over a prototyping board from Sundance technology. This board is composed of one DSP C6201 and one FPGA Xilinx Xc2v2000.

Figure 4 shows the resulting design of the reconfigurable transmitter. The FPGA is divided in two parts. The first one is static and implements non reconfigurable logic, the second one takes 8% of the FPGA and is dedicated to the dynamic operator. As modulation functionality is runtime reconfigurable, design

generated by SynDEx for this block is represented by operator *Op_Dyn*.
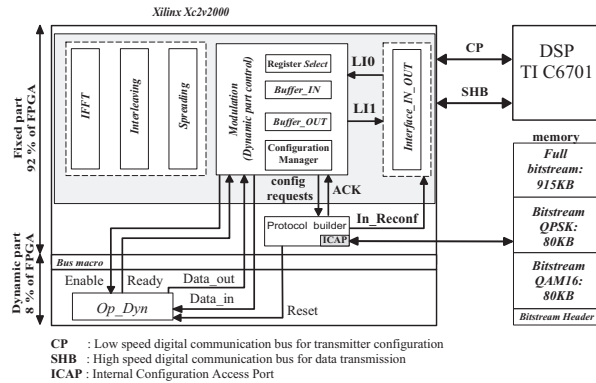

Figure 4. Reconfigurable MC-CDMA transmitter architecture

The DSP can select modulation performed by the dynamic part by sending this value to module *Interface_IN_OUT*. This value is send to block modulation through communication link *LIO*. *Interface_IN_OUT* module receives DSP data from SHB bus. Receiving process can be locked-up during partial reconfigurations thanks to signal *In_Reconf*. If *Select* register value is changed, block *modulation* sends a reconfiguration request to the protocol builder component which is next in charge to address external memory and drive ICAP. The reconfiguration time needed to reconfigure *Op_Dyn* takes about 4ms.

As shown in Table 1, FPGA resources utilization needed to implement QPSK and QAM-16 modulations are more important with a dynamic reconfiguration scheme. This overhead is due to the generic VHDL structure generation, based on the macro code description, which is more adapted for medium-complex data path control. However this gap is decreasing with the number of different reconfigurations needed and the ability of runtime reconfiguration to provide virtual hardware. The flexibility given by this methodology and the automatic VHDL generation can overcome hardware resource overhead.

7   Conclusion

We have described a methodology flow to manage automatically partially reconfigurable parts of a FPGA. It fully exploits advantages given by partially reconfigurable components. The AAA methodology and associated tool SynDEx have been used to perform

Table 1. Fix-Dynamic modulation implementation comparison

mapping and code generation for fixed and dynamic parts of FPGA. Either, SynDEx's heuristic needs additional developments to optimize time reconfiguration. Furthermore, complex design and architecture can support more than one dynamic part. This design flow has the main advantage to target as well as software components as hardware components to implement complex applications from a high level functional description. This methodology can easily be used to introduce dynamic reconfiguration over already developed fixed design as well as for IP block integration.

References

[1] F. Berthelot, F. Nouvel, and D. Houzet. Design methodology for dynamically reconfigurable systems. *JFAAA , Dijon France*, pages 47–52, January 2005.

[2] J. Harkin, T. McGinnity, and L. Maguire. Partitioning methodology for dynamically reconfigurable embedded systems. *IEE Proc-Comput. Digit. Tech*, 147, November 2000.

[3] S. Lenours, F. Nouvel, and J.-F. Helard. Design and implementation of mc-cdma systems for future wireless networks. *EURASIP JASP*, pages 1604–1615, August 2004.

[4] J. Mitola. Software radio architecture evolution: Foundations, technology tradeoffs, and architecture implications. *IEICE Trans. Commn.*, E83-B(6):1165–1172, June 2000.

[5] J. Noguera and R. Badia. A hw/sw partitioning algorithm for dynamically reconfigurable architectures. *Proc. Design Automation and Test in Europe*, pp. 72934, 200t, 2001.

[6] C. Sorel and Y. Lavarenne. From algorithm and architecture specifications to automatic generation of distributed real-time executives: a seamless flow of graphs transformations. *Formal Methods and Models for Codesign Conference, France*, pages 123– 132, June 2003.