A Device-Controlled Dynamic Configuration Framework Supporting Heterogeneous Resource Management

H. Tan and R. F. DeMara
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450

Abstract - In this paper, a lightweight autonomous reconfiguration approach is developed for Field Programmable Gate Arrays (FPGAs). Under the Multilayer Runtime Reconfiguration Architecture (MRRA) paradigm, hardware configuration information is read and operated on directly at runtime to provide low overhead dynamic reconfiguration. This enables a standardized set of Application Programming Interfaces (APIs) for uniform access to heterogeneous logic and other resources. A prototype MRRA system is developed for Xilinx Virtex II Pro family of FPGAs to exercise partial reconfiguration capability. The Virtex II Pro On-Chip PowerPC core is used to control these reconfiguration protocols implemented in user logic. These two features make an autonomous reconfiguration system possible, allowing a FPGA to efficiently reconfigure itself under the control of a microprocessor core instantiated within the FPGA fabric.

1 Introduction

As more and more applications in multimedia, cryptography and evolutionary systems can benefit from dynamic reconfiguration [1], autonomous reconfiguration of FPGAs becomes an important issue. In this paper, with the partial reconfiguration technique from Xilinx [6] and powerful FPGA devices equipped with on-chip CPU cores, a new *autonomous dynamic reconfiguration* approach is addressed.
This paper is organized in the following manner. In Section 2, the related research is identified. In Section 3, the autonomous reconfiguration problem is defined along with the MRRA design. Section 4 introduces the current MRRA prototype which is described in detail. Section 5 describes the current results. Section 6 provides the conclusions to-date.

2 Related Research

With the appearance of the partial reconfiguration technology in recent years, a series frameworks for dynamic reconfiguration have been developed. Such recent proposed tools include [2]-[5].
Moraes, Mesquita, Palma and Moller developed a set of tools for partial reconfiguration on Virtex XCV300 devices [4]. While providng useful features, some steps with their tools must be carried out manually. Also, their environment does not support core relocation capabilities.
Raghavan and Sutton's tool called JPG was developed for Xilinx Virtex devices [5]. The JPG tool is based on the Xilinx Java-based JBits API, which allows an application to instantiate a component, generate its corresponding bitstream, and download it to a Virtex FPGA. However, because of the its Java interpretation overhead, the tool has some speed and scalability limitations, which conflict with the objectives of the performance improvement benefits sought by partial reconfiguration. And JPG can only be used by Virtex FPGAs.

A two-layer framework for Virtex II devices had been suggested by Blodget, McMillan [2] and Fong, Harper, Athanas [3] separately. The system enables self-reconfiguration through the reconfiguration hardware interface Internal Configuration Access Port (ICAP) inside the Xilinx FPGA. However, because of limitations with the use ICAP, the bit stream must be manipulated directly. Thus, the range of potential applications is limited.

A more powerful general-purpose framework for a wide variety of practical applications would be useful to integrate and optimize existing reprogrammable technologies and theories. Ideally, this approach would provide a set of standardized Application Programming Interfaces (APIs) for uniform access to heterogeneous physical resources and thus enable more sophisticated autonomous dynamic reconfiguration.

3 MRRA Design

In order to accommodate the large number and wide variety of reconfiguration processes required by different applications, a 3-layer framework, called the *Multiple Runtime Reconfiguration Architecture (MRRA)* is being developed to carry out the partial reconfiguration tasks efficiently.

Figure 1 shows the conceptual organization of the MRRA framework. The bottom layer of the architecture is the *Reconfigurable Resource Layer*. The configuration bit stream is downloaded to the targeted FPGA's reconfigurable units using the hardware interface at this layer. When the initial configuration and the run-time partial reconfigurations are carried out, the resources are configured dynamically. Input and output data of the FPGA system is also passed between the logic control from the top layer and the bottom-layer FPGA reconfigurable units through this path to perform the routine tasks. Internal or External RAMs may be used as a temporary data and `.bit` file buffer in this layer to accelerate the transfer process through pipelining and speed matching.

The middle layer is called *Translation Layer*. In this layer, the logic circuit descriptions for a palette of tasks are translated into specific physical details by a hardware-dependent translation engine. The new partial `.bit` file is generated to reconfigure a specific area of a device at this layer, while the remainer of the device continues operation.

The top layer is the *Algorithm Layer*. In this layer, the data of the task routines are utilized by the high-level applications. New configurations are also generated based on hardware-independent algorithms. The configurations, including the routing information and functional units, are described in an abstract logic form at this layer. This acts as an entry file for further processing by the lower layers.

The MRRA architecture can be implemented into Loosely-Coupled System (LCS) or System-on Chip (SOC). Figure 1 has depicts the possible hardware interfaces for both the LCS and SOC implementations.

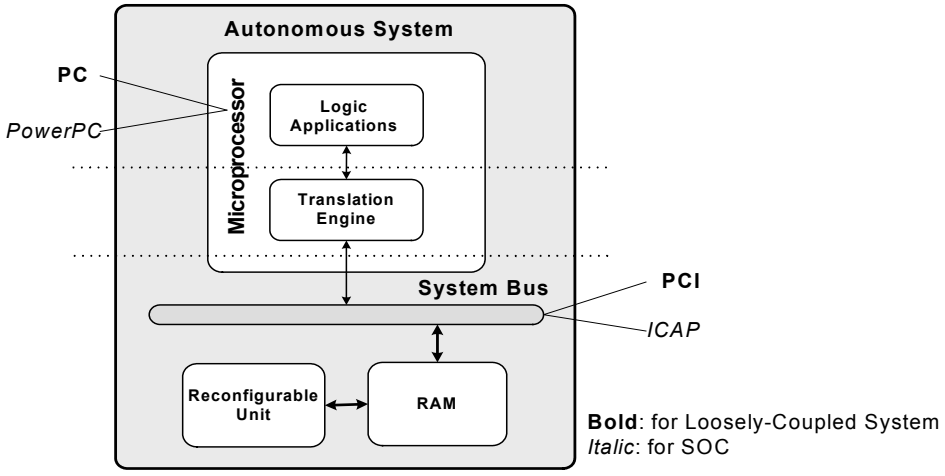

Figure 1: Multi-layer Runtime Reconfiguration Architecture

The major benefits of this MRRA framework including increased design productivity, portability, resource utilization, and autonomous operation.

4 Prototype Development

Figure 2 shows the Loosely Coupled prototype of the MRRA architecture developed for Xilinx Virtex II Pro platforms with Module-Based Partial Reconfiguration [7] technique. This solution was based on two COTS hardware components: the Avnet Virtex II Pro development board and a high-end Pentium-based Host PC. These components communicate through the PC's standard PCI bus interface.

The reconfiguration layer of this prototype includes an on chip hardware subsystem and their communication APIs, which then have an on chip PowerPC part and a host PC part. Both the Translation Layer and the Algorithm Layer reside in the Host PC. The on-chip hardware subsystem includes two subsets named *fixed subset* and *reconfigurable subset.* The *fixed subset* are composed of an on-chip PowerPC 405 CPU core, block RAMs, all the onboard peripherals such as SRAM and RS232 interface and their control interfaces generated inside the FPGA. In order to connect the PowerPC core with all the heterogeneous FPGA resources inside the chip, a 3 segment path is also established, which includes an *On-chip Peripheral Bus* (*OPB*), a *Processor Local Bus* (*PLB*) and a *bridge core* (*PLB2OPB*) providing access to the OPB from the PLB.

All the remaining modules comprise the *reconfigurable subset*. In order to maintain the correct communication with the PowerPC through the OPB bus with those reconfigurable and reloacated modules, a simplified *Intellectual Property Interface* and *Bus Macro* structures [7] have been adopted. Thus, all the modules can communicate and be controlled seamlessly.

The data communication of this prototype is bi-directional. As shown in Figure 2, the input path for the LCS begins at the PC. The PC sends the data over the PCI bus to the on-board SRAM with an interrupt request, which will influence the corresponding register of the on-chip interruption controller. Then V2Pro/PowerPC receives the interrupt request, executes its Interrupt Serivce Routine and finally reads data from the SRAM. On the other hand, the Output path data starts from V2Pro/PowerPC to the on-board SRAM. The PC keeps polling the SRAM for data available flag and then receive the data over PCI bus when the flag was set to ready.

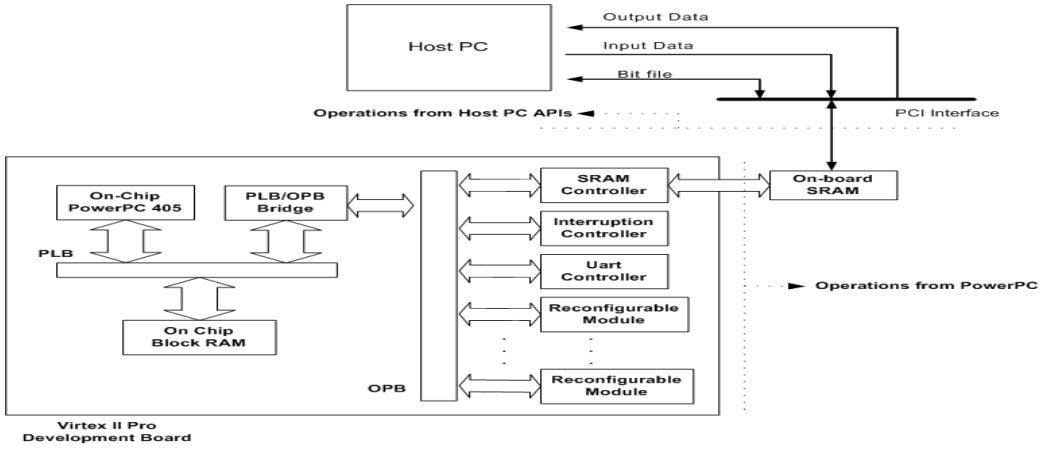

Figure 2: Loosely-Coupled MRRA System

Standardized communication APIs based on the above communication protocol have been developed in the LCS framework. For the PC part, API's can be categorized into 3 groups: *Initialization Operations*, *Configuration File Operations* and *On-board Memory Operations*. For the On-Chip part, on top of the basic embeded operating system, the APIs for the external interruption operations and memory/register operations are provided.

5 Current Results

In the current prototype, a Single Error Correction Double Error Detection (SECDED) circuit has been incorporated as the reconfigurable module. Preliminary tests shows that both data operations and `.bit` file reconfigurations has been carried out successfully in this autonomous system.

Table I: Resources Utilization

Resource name	Number of Available	Number of Used	utilization
IOBs	396	85	21%
Slices	4928	1805	36%
BRAM	24	44	54%
TBUFs	2464	352	14%
PPC405	1	1	100%
BUFGMUXs	4	1	25%

Table 1 lists the utilization of different kinds of reconfigurable resources for the current autonomous reconfiguration prototype used in the SECDEC case study. As listed in the table, the current hardware infrastructure has only occupied one third or even less of the reconfigurable resources, which provides sufficient area for large reconfigurable tasks or multiple reconfigurable tasks to be implemented based on this system. For the on-chip software system, only half of the block RAM has been engaged in current prototype, which also enables the possibility of further expansion.

6 Conclusion

In this paper, a 3-layer architecture named MRRA, is described for autonomous reconfiguration. A preliminary loosely coupled prototype based on the MRRA architecture has been developed and demostrated. Simple communication and synchronization protocol is also designed between microprocessors for data processing. Currently the top-level integration of a *Genetic Algorithm (GA)* based fault handling application for sel-repairing evolvable hardware is under development. Further performance evaluation of this prototype will follow along with practical area management modules. More metrics are also expected to be designed in the future for deeper and more thorough analysis for antonomous reconfiguration systems.

Acknowledgments

This research was supported in-part by NASA Intelligent Systems NRA Contract NNA04CL07A.

References

[1] Michael Barr, “A Reconfigurable Computing Primer”, Multimedia Systems Design, pp. 44-47, Sep. 1998.

[2] B. Blodget, S. McMillan, and P. Lysaght. “A lightweight approach for embedded reconfiguration of FPGAs”, in Proceedings of Design, Automation and Test in Europe Conference and Exhibition, pp. 399 – 400, 2003.

[3] R.J. Fong, S.J. Harper, and P.M. Athanas. “A versatile framework for FPGA field updates: an application of partial self-reconfiguration”, in Proceedings of 14th IEEE International Workshop on Rapid Systems Prototyping, pp. 117 – 123, 9-11 June 2003.

[4] D. Mesquita, F. Moraes, J. Palma, L. Moller, and N. Calazans. “Remote and partial reconfiguration of FPGAs: tools and trends”, Parallel and Distributed Processing Symposium, 2003. Proceedings. International, 22-26 April 2003.

[5] A. K. Raghavan and P. Sutton. “JPG - a partial bitstream generation tool to support partial reconfiguration in virtex FPGAs ”, in Proceedings of International Parallel and Distributed Processing Symposium,IPDPS 2002, pp. 155 – 160, 15-19 April 2002.

[6] Xilinx, Inc. “Virtex-II Pro Platform FPGA User Guide”, v2.4, Aug. 2004.

[7] Xilinx, Inc. “Two Flows for Partial Reconfiguration: Module Based or Difference Based” , v1.1, Nov 2003.