Facilitating Processor-Based DPR Systems for non-DPR Experts

Edward Chen[1], William Gruver[1], Dorian Sabaz[2] and Lesley Shannon[1]

School of Engineering Science, Simon Fraser University[1]
Burnaby, BC, Canada
{ekchen gruver, lshannon}@sfu.ca

Intelligent Robotics Corporation[2]
North Vancouver, BC, Canada
{dorian}@iroboticscorp.com

ABSTRACT

Currently, only Xilinx Field Programmable Gate Arrays (FPGAs) support Dynamic Partial Reconfiguration (DPR). While there is currently some Computer Aided Design (CAD) tool support for ISE-based DPR designs, none exists for microprocessor-based designs created in EDK. Creating DPR systems with the limited tool support currently available for ISE-based systems is already a challenging and complex process for novice DPR designers. These difficulties are severely compounded for potential microprocessor-based designs requiring a significant learning curve for novice DPR designers before they can successfully create their first working DPR system. This paper presents preliminary work towards extending the automation in Xilinx®'s current DPR design flow to include microprocessor based systems. The objective is to abstract low level details for novice designers, allowing them to focus on learning how to improve the quality of their design as opposed to how to perform the necessary manual transformations to generate a preliminary functional design. A case study demonstrated that the learning curve required to implement a first working design could be reduced by more than a factor of 15 times by improving the current automation available for microprocessor-based EDK designs.

Keywords

Field Programmable Gate Arrays, Dynamic Partial Reconfiguration, Computer Aid Design (CAD) Tools, SoC Design, Soft Processors

1. INTRODUCTION

Select Field Programmable Gate Arrays (FPGAs) from Xilinx® allow Dynamic Partial Reconfiguration (DPR) of the programmable fabric. DPR allows selected areas of the target device to be reprogrammed while the remainder of the fabric is active. Multiple modules are then able to time-share the same physical area on the target device and thus virtually increasing its physical resources. A DPR system has advantages over conventional implementation methods such as reduced footprint, cost, device count and power dissipation [1,6,7].

DPR systems are typically more complex to implement than non-DPR systems. It is often possible to purchase alternative devices with sufficient logic to satisfy design constraints. However, as power and footprint become essential factors in many of today's commercial products, a DPR system may become a practical solution.

Currently, Dynamic Partial Reconfiguration has been used in applications such as cryptography [3], network security [4], reconfigurable communication [5], and aerospace [6]. In Zeineddini [3], partial reconfiguration is used to devise a secure reconfiguration scheme that minimizes reverse engineering and bitstream cloning. Kao [5] discusses how applications designed for the aerospace industry may be subject to *Single Event Upset* (SEUs) that may occur from in-orbit, space-based, and extra-terrestrial applications. Other literature [6] [7] discusses the applications of DPR in different industries.

Xilinx had previously presented documentation outlining the exact procedures to be followed when implementing an ISE-based DPR design [2]. However, these procedures require a thorough understanding of different Xilinx CAD tools and the DPR design flow for a novice DPR designer to create his initial DPR design. The designer should focus on enhancing his DPR system, rather than the transformation from a non-DPR to a DPR system. Recently, Xilinx has incorporated functionality into the PlanAhead™ software tool that can be used to facilitate the development of an ISE-based DPR system. However, PlanAhead does not directly support the generation of microprocessor-based DPR systems developed in EDK without significant manual alterations. As an increasing number of FPGA designs include microprocessors, there exists a need to facilitate the generation of potential DPR-systems developed in EDK.

This paper presents preliminary work towards extending the automation in Xilinx's current Dynamic Partial Reconfiguration (DPR) design flow to include microprocessor based systems. *EuTOPIA* (**E**DK **TO** **P**lanAhead **I**mplementation **A**utomation) automates the process of converting the non-DPR, microprocessor-based design in EDK into a PlanAhead-based, DPR system. *EuTOPIA* abstracts low level details, allowing designers to quickly generate their initial EDK-based DPR systems. *EuTOPIA* allows novice designers to focus on how to improve the quality of their systems as opposed to how to perform the necessary manual transformations to generate a preliminary functional system.

Figure 1(a) shows the high level overview of the standard EDK-based design flow. Modifications are required to this design flow if the processor-based design implemented in EDK is to leverage DPR. Figure 1(b) illustrates the required modifications to the standard design flow required to implement a microprocessor-based DPR system from EDK. The bolded boxes highlight the modifications to the standard EDK-based design flow for DPR system implementation.

The designers start by implementing a standard EDK project as they would for a static embedded system. Manual modifications of the HDL files are then required before the EDK project is exported to ISE for synthesis. Given a structured input architecture in EDK, *EuTOPIA* generates a PlanAhead project with full and partial bitstreams. Users are then able to modify and optimize their systems as desired. *EuTOPIA* provides users a quick method to generate their systems with the core functionalities and flexibilities of a PlanAhead Project.

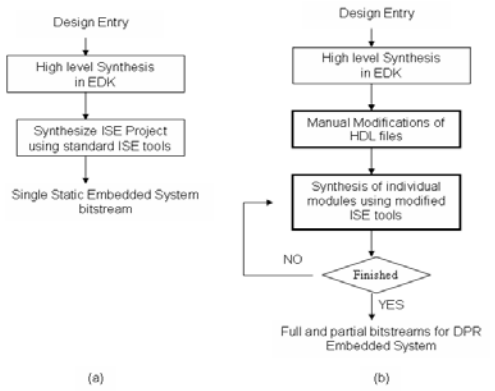

Figure 1: High level EDK-based design flow for (a) standard non-DPR systems and (b) the modified design flow for EDK-based DPR systems

The high-level overview of *EuTOPIA* is shown in Figure 2.

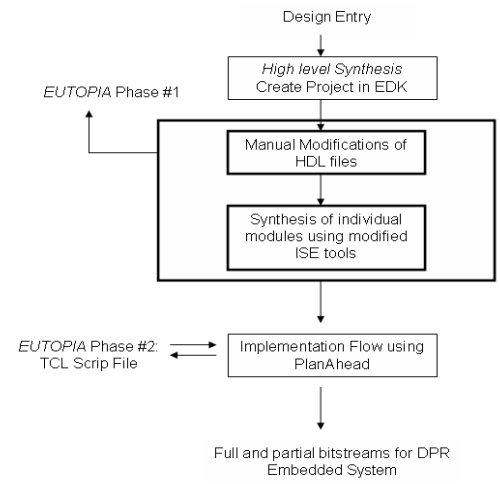

Figure 2: High-level Overview of *EUTOPIA*

EuTOPIA uses a two-phase approach to generate the PlanAhead project. In Phase#1, *EuTOPIA* modifies HDL files in the EDK project to conform to the DPR design flow. The modified HDL files are then synthesized using ISE. In Phase#2, a TCL script that contains detailed implementation instructions for PlanAhead is generated. PlanAhead then invokes this TCL script to create the DPR system. Current version of *EuTOPIA* supports three Partially Reconfigurable Regions (PR-Regions). The locations and sizes of the PR Regions and the types of Bus Macros used are automatically assigned. PR Regions are configured as OPB slaves and data-width are fixed at 8 bits.

EuTOPIA is not designed as a replacement for either EDK or PlanAhead, but is meant to complement the functionality provided by both tools. It allows novice users to quickly generate their initial DPR systems and focus their effort on enhancing their designs, rather than the transformation from non-DPR to DPR systems. After the initial DPR system is established, experienced designers can pursue more optimized designs by modifying the existing PlanAhead project.

2. Preliminary Results and Future Works

An identical experimental DPR system is implemented twice, once using *EuTOPIA* and once manually by an experienced user. It was found that the implementation time for *EuTOPIA* was **15X** faster than manual implementation. Users are also required to understand not only the intricate details of the DPR flow, but also the detailed usage of ISE and PlanAhead for manual implementation. The minimal time required to have a thorough understanding DPR, ISE and PlanAhead is approximately 50+ hours. In contrast, only 2-3 hours are required to have the nominal knowledge of the underlying process and the input constraints to *EuTOPIA*. Each additional implementation of this experimental DPR system requires only 20 minutes for *EuTOPIA*, but approximately 1.5 hours for manual implementation. Over the design-cycle of a product where designers may generate tens or even hundreds of different implementations of their DPR design, these time savings would be significant.

Future versions of this CAD tool will provide designers with more flexibility in the options they can choose for their DPR systems designs. Current upgrades being developed include: increase the number of allowable PR Regions, user-specify the locations and sizes of each PR Region, perform architectural exploration, increase the data-width between the OPB Bus and the PR Regions, support FSL and PLB connections to the PR Regions, and the inclusion of additional development boards and FPGA device types.

3. References

[1] J. Becker, M. Huebner, and M. Ullmann, "Power estimation and power measurement of Xilinx Virtex FPGAs: trade-offs and limitations", 16th Symposium on Integrated circuits and Systems Design, SBCCI), 8-11 Sept 2003, pp.283-288

[2] "Two Flows for Partial Reconfiguration: Module Based or Difference Based", Xilinx Application Note XAPP290 (V1.2), www.xilinx.com/bvdocs/appnotes/xapp290.pdf, Dec 8, 2007.

[3] A Zeineddini, "Secure Partial Reconfiguration of FPGAs " , M. Sc Thesis, George Mason University , Fairfax, VA, USA, 2005

[4] R.V. Kshirsagar, R.M. Patrikar : "Design of a Reconfigurable Multiprocessor Core for Higher Performance and Reliability of Embedded Systems", *IEEE proceedings of IFIP 14th International conference on Very Large Scale Integration*, VLSI-SoC 2006,pp 251-254, Oct. 16-18, 2006,Nice,France

[5] C. Kao. "Benefits of Partial Reconfiguration" Xilinx Xcell Journal Fourth Quarter 2005 pp65-68

[6] N. Dorairaj, E. Shiflet, and M. Goosman. "PlanAhead Software as a Platform for partial Reconfiguration" Xilinx Xcell Journal Fourth Quarter 2005 pp68-71

[7] P. Lysaght, B. Blodget, J. Mason, J. Young and B Bridgeford: Invited Paper: Enhanced Architecture, Design Methodologies and CAD Tools for Dynamic Reconfiguration for Xilinx FPGAs. FPL2006: 1-6