High-level Cognitive Radio Design using Xilinx FPGAs

Jörg Lotze[†], Suhaib A. Fahmy[†], Juanjo Noguera[*], Linda Doyle[†] and Robert Esser[*]

[†]*Centre for Telecommunications Value-chain Research*
University of Dublin, Trinity College
{jlotze,suhaib.fahmy,linda.doyle}
@tcd.ie

[*]*Xilinx Research Laboratories*
Xilinx Ireland
{juanjo.noguera,robert.esser}
@xilinx.com

1 Introduction

Cognitive radios are computationally intensive wireless systems that implement highly demanding digital signal processing algorithms, on different platforms. General Purpose Processors (GPPs) are the most programmable and flexible platforms, though is reflected in their relatively poor performance. On the other hand, Application Specific Integrated Circuits (ASICs) provide high performance at the cost of reduced flexibility.

Modern Field Programmable Gate Arrays (FPGAs) in addition to providing programmable logic resources, integrate embedded processors and on-chip memory resources on a single die. They are programmable computing platforms, with run-time reconfigurable logic fabric [1].

Xilinx FPGAs are excellent computing platforms for Software Defined Radio (SDR) and cognitive radio applications, since they provide some measure of the flexibility afforded by GPPs along with performance and power efficiency closer to that of ASICs.

However, using FPGAs requires significant hardware design experience. Typically radio experts do not possess this experience and therefore tend to use GPP-based cognitive radio platforms with their disadvantages. We propose a framework to allow radio designers to benefit from FPGAs without the need of hardware design experience by abstracting the radio design process from the FPGA implementation details.

2 An FPGA-based Cognitive Radio Framework

Objectives

A primary goal for the proposed framework is to allow communications engineers (*i.e.*, non-hardware experts) to quickly design and implement cognitive radios on FPGAs, thus gaining from the significant performance and power advantages they provide. FPGAs are traditionally considered to be hardware devices, which are difficult to use since they require experience in low-level hardware design.

The radio designer on the other hand prefers to focus on the radio design and not think about implementation details on the FPGAs. We will develop a set of high-level tools that enable the radio designer to choose radio components from a library and connect them together to create a functional description of the cognitive radio without knowledge of low level component implementation details.

A second objective is to show that FPGAs are *programmable* computing platforms and not rigid computing devices. Hence, an important goal of the project is to demonstrate that using FPGAs we can implement cognitive radios which are *programmable* at run-time.

System Architecture

When executing a cognitive radio on an FPGA, the DSP components can be implemented in either the PowerPC processor or in the logic fabric. Each DSP component consists of two implementations: one software implementation for PowerPC execution, and one hardware implementation for execution in the FPGA's logic fabric. However, both implementations are functionally identical.

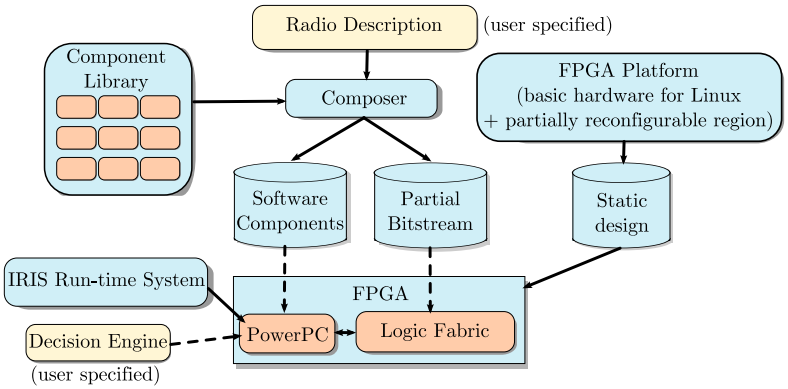

Figure 1: The system architecture.

To decouple the radio design as much as possible from the implementation details we propose a system architecture which decouples the design-time system and the run-time system, as shown in Figure 1. All the radio designer has to provide is an eXtensible Markup Language (XML) description of the radio chain and optionally a C++ description of the decision engine. The *Composer* takes that radio description and creates an efficient FPGA implementation consisting of some radio components implemented on the PowerPC processor and other radio components implemented in the FPGA logic fabric. Thus, the output of the composer is a partial bitstream that must be downloaded to the partially reconfigurable region defined in the FPGA (see Figure 1).

As run-time system we use *Implementing Radio In Software (IRIS)*, a software radio system developed at Trinity College Dublin [2]. IRIS is a modular, highly flexible and highly reconfigurable software radio system, which as part of this research project, has been ported to run on the embedded PowerPC of a Xilinx FPGA using the embedded Linux Operating System (OS) distribution described in [3]. It instantiates the hardware and software DSP components based on the output generated by the composer, and runs the radio. The user defined decision engine can reconfigure the radio at run-time, possibly causing a FPGA partial reconfiguration.

We have successfully demonstrated a live audio transmission application with the transmitter running on the Xilinx University Program (XUP) board [4] using the proposed system architecture shown in Figure 1. This live audio transmission demonstrator will be shown at the *Collaborative International Software-defined Radio Workshop*.

3 Conclusion

High-level design tools are a key technology to extend the use of FPGAs among radio designers. These tools should hide to the radio designers all the FPGA implementation details. The proposed Cognitive Radio (CR) framework allows radio designers to benefit from the embedded processor as well as from the logic fabric of Xilinx FPGAs without hardware design experience. It is a possible answer to the ongoing discussion whether to use GPPs or FPGAs for the implementation cognitive radios.

References

[1] P. Lysaght *et al.*, "Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAs," in *International Conference on Field Programmable Logic and Applications (FPL)*, Madrid, Spain, Aug. 2006.

[2] P. MacKenzie, "Software and reconfigurability for software radio systems," Ph.D. dissertation, University of Dublin, Trinity College, Ireland, 2004.

[3] Xilinx Git repository. Xilinx, Inc. [Online]. Available: git://git.xilinx.com

[4] *Xilinx University Program Virtex-II Pro Development System – Hardware Reference Manual*, Xilinx, Inc., Mar. 2005. [Online]. Available: http://www.xilinx.com/univ/XUPV2P/Documentation/XUPV2P_User_Guide.pdf