A lightweight approach for embedded reconfiguration of FPGAs

Brandon Blodget
Xilinx Research Labs
2100 Logic Drive, San Jose, CA, 95124, USA
brandon.blodget@xilinx.com

Scott McMillan
Xilinx Inc.
2100 Logic Drive, San Jose, CA, 95124, USA
scott.mcmillan@xilinx.com

Patrick Lysaght
Xilinx Research Labs
2100 Logic Drive, San Jose, CA, 95124, USA
patrick.lysaght@xilinx.com

Abstract

This paper presents a lightweight approach for embedded reconfiguration of Xilinx Virtex II^{tm} series FPGAs. A hardware and software infrastructure is reported that enables an FPGA to dynamically reconfigure itself under the control of a soft microprocessor core that is instantiated on the same array. The system provides a highly integrated, lightweight approach to dynamic reconfiguration for embedded systems. It combines the benefits of intelligent control, fast reconfiguration and small overhead.

1. Introduction

Dynamic reconfiguration and self-reconfiguration are two of the more advanced forms of FPGA reconfigurability. Dynamic reconfiguration implies that an active array may be partially reconfigured, while ensuring the correct operation of those active circuits that are not being changed. Self-reconfiguration extends the concept of dynamic reconfigurability. It assumes that specific circuits on the array are used to control the reconfiguration of other parts of the FPGA. Clearly the integrity of the control circuits must be guaranteed during reconfiguration, so by definition self-control is a specialized form of dynamic reconfiguration.

Both dynamic reconfiguration and self-reconfiguration rely on an external reconfiguration control interface to boot an FPGA when power is first applied or the device is reset. Once initially configured, self-control requires an internal reconfiguration interface that can be driven by the logic configured on the array. On Xilinx Virtex II^{tm} parts, this interface is called the internal reconfiguration access port (ICAP). We have interfaced the ICAP to Xilinx's MicroBlaze, a 32-bit RISC soft microprocessor core, to provide intelligent control of device reconfiguration at runtime. The integration of this functionality is especially attractive for embedded systems. This lightweight approach maximizes flexibility while minimizing additional external circuitry.

In the next section, we review the details of the ICAP and the reconfiguration mechanisms of the Virtex II^{tm} FPGAs. This provides the background necessary for an appreciation of the hardware and software subsystems of the reconfiguration control infrastructure that are described in sections 3 and 4. Section 5 describes a novel application of the new controller in telecommunications switching. Section 6 concludes the paper.

2. Virtex II^{tm} Reconfiguration Mechansim

Virtex II^{tm} devices are configured by loading application specific data into configuration memory. On the Virtex II^{tm} the configuration memory is segmented into "frames". Virtex II^{tm} devices are partially reconfigurable and a frame is the smallest unit of reconfiguration. The number of frames and the bits per frame is different for the different devices in the Virtex II^{tm} family. The number of frames is proportional to the CLB width of the device. The number bits per frame is proportional to the CLB height of the device. For example an XC2V40 has 404 frames and 104 bytes per frame. The XC2V6000 has 2508 frames and 984 bytes per frame.

The Virtex II^{tm} has an internal reconfiguration access port (ICAP). ICAP provides an 8 bit input data bus and an 8 bit output data bus which can be used by internal logic to reconfigure and readback configuration memory. As an example to reconfigure/readback a LUT requires 2 frames of data. On a XC2V40 at 50 MHz this would take 4us. On a larger device like the XC2V6000 it would take 40us.

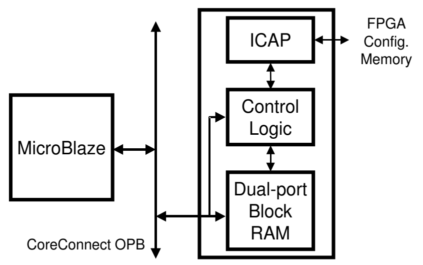

Figure 1. Block diagram of the hardware components of the system

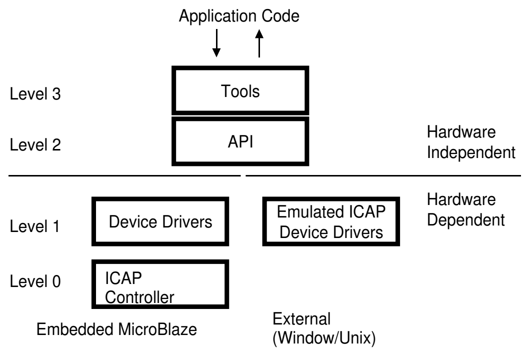

Figure 2. Software subsystem architecture

2.1. Hardware Subsystem

The hardware subsystem of our reconfiguration control infrastructure sits on the on-chip peripheral bus (OPB). The MicroBlaze microprocessor communicates with this peripheral over the OPB bus. The hardware peripheral is designed to provide a lightweight solution to reconfiguration. In order to do this it employees a read/modify/write strategy. Only one frame of data is worked on at one time. This way external memory is not needed to store a complete copy of the configuration memory. *Figure 1* shows a block diagram of this peripheral. The MicroBlaze program request a specific frame, then the control logic of the peripheral uses the ICAP to do a readback and loads the configuration data into a dual-port block RAM. One block RAM can hold an XC2V8000 data frame easily. When the readback is complete the MicroBlaze program directly modifies the configuration data stored in the bram. Finally the ICAP is used to write the modified configuration data back to the device.

2.2. Software Subsystem

The software subsystem is implemented using a layered approach. This allows us to change the implementation of the lower layers without affecting the upper layers. This layered approach proved useful for debugging the Level 2 API in that we were able to test the API in a purely software environment to make sure the partial reconfiguration packets were getting created correctly.

The Level 2 API provides useful functions for embedded applications requiring reconfiguration. There are functions for downloading partial bitstreams stored in external memory. There are functions for copying regions of configuration memory, and pasting it to a new location. Finally there are JBits like API calls for reconfiguring select FPGA resources.

It is foreseen that a Level 3 could sit between the Level 2 API and the embedded application. This level could enforce user defined design rules. For example the user could define certain regions that are static and should not be reconfigured.

2.3. Work in progress

Several extensions and applications of the system are in progress. The first of these is to produce a variant of the original system on a Virtex VII Pro device in which a PowerPC, hard microprocessor, will replace the MicroBlaze soft processor. The new system will be used to control the operation of a dynamically reconfigurable crossbar switch, reported previously. The result will be the first, fully integrated, dynamically reconfigurable, crossbar switch on an FPGA.

2.4. Conclusions

We have described an intelligent subsystem for lightweight reconfiguration of Xilinx Virtex II^{tm} FPGAs in embedded systems. The system enables self-reconfiguration under software control within a single FPGA. The reconfiguration subsystem has a layered hardware and software architecture that permits a variety of different interfaces to maximize flexibility and ease-of-use. The first application of the controller is to integrate the control of the operation of a dynamically reconfigurable crossbar switch for telecommunications applications.

2.5. Acknowledgements

It is a pleasure to acknowledge the contributions made to this work by Xilinx Research Labs members Phil James-Roxby, Eric Keller and Prasanna Sundararajan.