Reconfigurable Computing for Embedded Systems, FPGA Devices and Software Components

Graham Bardouleau and ***James Kulp***
Mercury Computer Systems, Inc.
Phone: 978-967-1653
Email Addresses: {gpb, jek}@mc.com

In recent years the size and capabilities of field-programmable gate array (FPGA) devices have increased to a point where they can be deployed as adjunct processing elements within a multicomputer environment. This enables these devices to become an element within a reconfigurable system performing processing of high data rate streams of data. Conventionally, these devices have performed basically fixed-function processing at the input to a system. However, through the use of component-based programming models, it is possible to view these devices as general-purpose processing accelerators where the need arises within a system.

A common approach to using FPGA devices in systems at present is based on the use of a dedicated driver or software proxy mechanism. The driver or proxy is responsible for controlling the flow of data between the FPGA and other elements within the system. This approach works, but often the driver or proxy requires intimate knowledge of the algorithm running within the FPGA device.

As the drive toward reconfigurable computing platforms continues, the need for a standardized middleware that can be implemented and supported on all forms of processing elements increases. Through the use of such a middleware it would be possible to interface any form of processing element, including microprocessors, digital signal processors (DSPs), FPGAs and even application specific signal processors (ASSPs) and application specific integrated circuits (ASICs). The middleware would define the mechanism by which data would be transferred between processing elements and the associated signaling necessary to ensure data integrity within the system.

Through the implementation of a middleware such as that mentioned above can provide a framework that supports a component-based application model by relieving the application implementation engineers of data movement and signaling issues. Various additional benefits are visible through the use of such a framework, including processor independence, fabric independence, and platform independence. The development of such a middleware and associated framework is ongoing at Mercury Computer Systems.

This paper describes the approach taken at Mercury to develop such a middleware and framework that supports the execution of components on PowerPC microprocessors as well as Xilinx FPGA devices, treating them as peers in a system of heterogeneous processing resources. We will discuss also how this approach maximizes the portability of FPGA functional code in software radio environments.