An FPGA-based *Soft* Multiprocessor System for IPv4 Packet Forwarding

Yujia Jin Kaushik Ravindran Nadathur Satish Kurt Keutzer
University of California at Berkeley, CA, USA
{yujia, kaushikr, nrsatish, keutzer}@eecs.berkeley.edu

Introduction

Modern network applications require devices that are flexible enough to support different services while providing performance at gigabit processing rates. However, prohibitive design costs and shrinking market windows restrict the number of IC design starts. The FPGA is a cost-effective platform; however, programming in HDL is time-consuming and less attractive to software designers. An alternative is a *soft multiprocessor system* on an FPGA. The soft multiprocessor is a configurable network of programmable processors and peripherals crafted out of processing elements, logic blocks and memories on an FPGA. For example, Xilinx supports soft multiprocessor development on the Virtex family of FPGAs, integrating the IBM PowerPC 405 cores on chip, soft MicroBlaze cores, and customizable peripherals.

Motivation for Soft Multiprocessors

Common system solutions for network applications are: (a) software on ASSPs, and (b) HDL for FPGAs. One would intuitively expect soft multiprocessors to perform lower than ASSPs or HDL solutions. But soft multiprocessors trade-off performance for advantages in product costs and development time.

First, technology scaling towards smaller process geometries is driving the IC design cost into the $20 million ranges. In turn, product revenues need to reach $200 million to repay the investment. FPGA designs obviate pressures due to prohibitive costs and IC turnaround times. Second, software development time is significantly less than HDL design time. Also, embedded applications have a general tradition of software implementation. Hence, the productivity associated with software implementations favor ASSP and soft multiprocessor solutions.

Soft multiprocessors achieve lower costs due to the underlying FPGA platform, while reducing development time due to software implementation. They provide a mechanism to easily explore and empirically evaluate various architectures, without ever paying the price to cast a design into silicon. Finally, soft multiprocessors could open the FPGA market to the larger world of embedded software designers.

Objectives of our Work

This work attempts to quantify the position of soft multiprocessors in the performance-cost-productivity trade-off space for network applications. The key questions are: (a) How much performance do soft multiprocessors lose compared to the other two system solutions? (b) Is the loss in performance worth the gains in lowering product cost and development time?

We empirically evaluate the effectiveness of soft multiprocessors for the data plane of the IPv4 packet forwarding application. We construct a 2-port 2 Gbps router as a soft multiprocessor on the Xilinx Virtex-II Pro FPGA . We then extend this multiprocessor solution for packet forwarding to incorporate Network Address Translation (NAT). The performance, development time and cost trade-offs for soft multiprocessor solutions are evaluated with respect to the two alternate system solutions: (a) a software implementation on the Intel IXP1200 network processor, and (b) an HDL design for the Xilinx Virtex-II Pro FPGA.

Preliminary Results and Extensions

Our preliminary multiprocessor for packet forwarding uses 14 MicroBlaze soft processors and 2 PowerPC cores arranged as multiple pipelined arrays. It is clocked at 100 MHz and achieves a forwarding rate of 1.8 Gbps. In terms of normalized performace, the multiprocessor loses 4X compared to HDL solutions and 2X compared to the IXP1200 network processor. However, the development time is significantly less compared to HDL solutions. The ease of implementation make it an ideal solution for prototyping new applications, while meeting modest performance requirements.

The next step is to automate the design space exploration process for soft multiprocessors. For an application specified in some domain specific language, the idea is to compose the architecture from processors and hardware blocks. The task allocation and scheduling problems are then solved to map application components to the multiprocessor.