Software for Reconfigurable Computing

Steven A. Guccione
Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124 (USA)
Steven.Guccione@xilinx.com

April 10, 1997

1 Introduction

In the mid-1980s, a new approach to hardware design was pioneered by Xilinx. With the advent of the Field Programmable Gate Array (FPGA), hardware designers were given some of the flexibility formerly reserved only for software designers.

The FPGA permits hardware to be dynamically configured. Logic gates and their interconnection can be specified and modified, in-system if necessary. This has several immediate benefits.

The ability to customize hardware allows designers to replace collections of small standard parts, the "glue" logic, with a single package. The cost savings of smaller boards with fewer packages made these devices immediately popular.

While the manufacturing advantages of FPGAs are immediately obvious, there is another benefit. A design could be improved and upgraded without having to replace existing hardware. The circuitry in the FPGA could be reprogrammed, extending the life of existing hardware and inventories.

Reconfigurability also provide a safety factor for engineers designing with FPGAs. A hardware bug does not require a complete re-fabrication of a piece of hardware. Again, reprogramming of the FPGA could repair the design, even for hardware already in the field.

Finally, the ability to dynamically reconfigure the hardware while in a functioning system provides the potential for reconfigurable multifunction hardware. Here, a system can dynamically change its hardware to implement an almost infinite variety of functions.

It is this last feature of FPGA hardware that is only recently been exploited. Dynamic reconfiguration permits any number of tasks previously performed by custom hardware to be done by a single general purpose reconfigurable hardware resource.

Perhaps more significantly, this opens the door for any design, even one previously implemented in software, to be implemented in hardware. The large financial commitment to build custom hardware is no longer necessary. Once established, the reconfigurable hardware platform may be used for nearly any purpose.

The benefits of implementing a system or algorithm in hardware as opposed to software are well known. Speedups of several orders of magnitude are common. As the density of FPGA devices has increased, so has the size and number of functions possible. This has led to a corresponding increase in the interest in reconfigurable hardware for algorithm acceleration.

While the hardware has continued to increase in speed and density, software to support this new approach to computation has been slower to develop. It is widely accepted that software support for reconfigurable hardware will be one of the keys to its widespread acceptance.

2 Reconfigurable Hardware

One of the first commercially successful uses for reconfigurable hardware was, perhaps not surprisingly, the emulation of hardware. In the late 1980s several companies manufactured and sold large arrays of FPGAs to emulate custom designs. These were immediately embraced by makers of complex integrated circuits, particularly microprocessor manufacturers. Now the logical correctness of a large integrated circuit could be verified at hardware speeds. This represented several orders of magnitude increase in speed over software simulations.

Software for these machines was, for the most part, commercially available circuit design CAE tools. What is perhaps most significant is that reconfigurable hardware had replaced large general purpose computers running circuit simulation software.

At this time, researchers were also beginning to use reconfigurable hardware for other computationally intensive tasks. One popular application was neural networks. A least three reconfigurable systems to accelerate neural networks were built. Again, these provided several orders of magnitude increase in performance over software based approaches.

As the success of these special purpose reconfigurable machines became known, it became clear that there were few limitations on the use of such dynamically programmable hardware to accelerate algorithms. The possibility of hardware comparable in size and cost to a workstation running at custom hardware speeds and outperforming supercomputers was recognized by several researchers. Projects experimenting with reconfigurable systems began, more or less simultaneously, at several Universities and research labs around the world. In 1993, the IEEE Workshop on FPGAs for Custom Computing Machines met and for the first time brought these researchers together.

3 Software Tools

With the success of special purpose reconfigurable hardware, several groups developed more general purpose hardware. Along with this hardware, research into new software techniques began.

The software development tools for the first systems were limited to available hardware design CAD tools available from the FPGA device vendor. This required that users of the system be proficient in both hardware and software design. In addition, there was little support for interfacing this customized hardware to software running on a host workstation or PC.

True high-level language support quickly became the Holy Grail of the emerging field of reconfigurable computing. Here, the goal was to take standard "dusty deck" code previously written for traditional processors and automatically compile it into circuits and support software for a reconfigurable system. Segments of the code benefiting from hardware acceleration would be automatically identified and mapped to the reconfigurable logic. The remaining code would execute on the host workstation or PC, and interface to the reconfigurable logic where necessary.

One compilation approach is to treat the reconfigurable logic as a coprocessor which augments the instruction set. This approach can trace its roots to experiments with writable instruction sets on mainframes and minicomputers in the 1970s. This permitted customizable instructions via microcode. With of limitations of compiler technology of the time and the advent of RISC processors the idea of customizable instruction set computers lay dormant for almost a decade.

Using reconfigurable logic as a coprocessor, however, complex operations requiring high-performance hardware could be dynamically configured. The interface would operate much like a math coprocessor in a workstation or PC. Unlike the math coprocessor, however, the instructions available could change from program to program or even from clock cycle to clock cycle.

This compilation approach uses a familiar programming model based on the traditional stored program or von Neumann architecture. This allows standard compilation techniques to be used with little modification. Unfortunately, the analysis of the software and the generation of a customized instruction set is a difficult optimization problem. And once all of this work is performed, the inherently serial model of computation still limits performance.

Another approach eschews the notion of processor instructions and treats the reconfigurable logic as a piece of custom hardware performing parallel processing. Here, data is sent to the reconfigurable logic, either in real-time by an external source or by a host processor. Results are read back to be further manipulated or displayed by the host machine. Such architectures are currently being used for applications such as cryptography and signal and image processing. Again, performance levels of supercomputers or custom hardware are routinely observed on these systems. One application, the searching of a genetics database, reported a speedup of over 40,000 over a workstation.

While performance is maximized, the software task becomes more difficult. Automatically identifying sections of code to be transformed into custom hardware has proved an illusive goal. Today most hardware designs are typically hand-crafted with circuit design tools, then interfaced with host software. Because the hardware and software design are decoupled, interfacing is often difficult. Links between CAD tools and compilers are gradually beginning to appear.

Much of the difficulty in using existing FPGA based machines appears to come from the underlying FPGA architectures. Because these devices have not traditionally been used for on the fly reconfiguration, their architectures are not well suited to reconfigurable processing tasks. The major barriers are slow serial reconfiguration and the inability to easily perform partial reconfiguration of a device.

Xilinx has recently announced the XC6200 device which takes aim at these deficiencies. A parallel microprocessor bus interface and fine-grained partial configuration combine to make the XC6200 an ideal platform for reconfigurable computing experimentation. Additionally, all wires in the XC6200 are unidirectional, so bus contention problems are eliminated. This makes it impossible to damage the device with a bad configuration file. This alone makes the XC6200 welcome in reconfigurable computing circles.

While a traditional FPGA CAD tool suite is available for the XC6200, other tools from researchers are beginning to emerge. Of particular interest in a set of Java libraries that provide full access to the hardware resources of the XC6200. Circuits may be configured and reconfigured and the host IO and processing performed in a single piece of code. This means that the gap between host software and CAD tools has, to some extent, been bridged. A single software development environment, the Java compiler, replaces the CAD tool suite and the host high level language compiler and eliminates the interfacing issues.

4 Conclusions

Software for reconfigurable systems is still in its infancy. The current trend appears to be away from traditional CAD tools such as schematics and hardware description languages. This is not surprising, since these tools are used to specify static hardware designs. Extending them to provide for dynamically changing circuits and interconnect appears to be a difficult, if impossible task.

But the real hope for software appears to be in the underlying device architectures. It is no secret that existing architectures are ill suited to reconfigurable processing. Some new devices appear to be addressing the systems issues, providing features such as unidirectional routing, microprocesor interfaces and partial reconfiguration.

Currently, ease of use of software appears the be the biggest barrier to acceptance of reconfigurable architectures. But the large gains in performance are still attracting new users. As the field accelerates, feedback from experiences with software and tools should help to shape and improve the software and the architectures in much the same way compiler technology influenced the microprocessor in the RISC revolution in the 1980s. As these systems become easier to use, expect to see more and more systems taking advantage of the performance gains available from reconfigurable computing.