Ph.D. Research Abstract: A methodology for Java/FPGA co-design

Philippe Faes, Ghent University
http://www.elis.UGent.be/~pfaes

Introduction

In recent years many methodologies have been proposed for co-designing Field Programmable Gate Array (FPGA) and software systems. In these systems the FPGA is used to accelerate highly parallelisable and time critical parts of the system, while an instruction set processor (ISP) is used for the sequential parts.

Existing systems require a specific programming style of the software developer. This makes it virtually impossible to use legacy software. Some systems require a message-passing interface between threads, and allow threads to migrate between software and FPGA [7]. Other systems give access to the hardware using a specific hardware driver and communication libraries [6]. Yet other systems provide a special compiler to enable hardware acceleration [11]. We propose a system where the original Java source code and even the compiled Java bytecode are left unmodified, and the Java Virtual Machine (JVM) is used for handling the communication with the FPGA. [4], [3]

System Description

In this research we take advantage of a JVM [2] for intercepting method calls. We leave the Java bytecode unmodified, and merely inform the JVM of which methods can be accelerated. Whenever one of these methods is called, the JVM intercepts the call and decides if it should be executed in hardware or software. For methods delegated to the FPGA, the primitive parameters are passed by value and the object parameters (such as arrays) are passed by reference, as the Java specification prescribes. These references can be used to fetch data from the main memory through Direct Memory Access (DMA) or they can be passed as arguments to other methods.

We regard the FPGA as a full partner of the instruction set processor. It can access main memory and call the JVM to execute methods and constructors on behalf of the FPGA. It can even perform synchronization operations on objects (lock and unlock).

In the future we would like to allow the JVM to reconfigure the FPGA, based on the current needs of the application. Programs with a phased behavior will benefit from this, since they can use the full FPGA for executing a certain algorithm in one phase, and a different algorithm in a second phase of their execution [10].

We will also introduce non-homogeneous memories. Different memory chips can be mounted on different circuit boards in a system, interconnected by some fabric. Some memories will have a better interconnection with the ISP, others will be more easily accessible by the FPGA. We will investigate where the data should be stored that is used both by the ISP and the FPGA. This decision is far from trivial in systems where data is produced by one routine on the ISP, and later read and modified by both the FPGA and ISP. We would like to minimize the communication cost, while preserving a transparent view on the memories.

Application and measurement data

As a first demo application, we have accelerated a Java application for comparing protein sequences. We use a Java implementation of the Smith-Waterman-Gotoh [5], [9] algorithm, based on [8]. We calculated the cost function for the alignment of one fixed protein sequence with each sequence in a database of 1000. This operation took 49.35s on an AMD Athlon MP 2600+ machine for a Java implementation on the JikesRVM 2.3.5 (a JVM formerly known as Jalapeño [1]). When we accelerate the application on the same machine with an Altera PCI Development Board with a Stratix 1s25 FPGA, we can execute the same operation in 1.24s, which is a performance gain of almost a factor 40.

We have shown that a standard, bidirectional and fully transparent interface between Java and reconfigurable hardware is feasible without the need to modify the Java bytecode, and that large performance gains can be achieved. We will continue to work on memory *management* and the communication protocol between Java and the hardware.

References

[1] Alpern, B., Attanasio, C. R., Barton, J. J., Burke, M. G., Cheng, P., Choi, J.-D., Cocchi, A., Fink, S. J., Grove, D., Hind, M., Hummel, S. F., Lieber, D., Litvinov, V., Mergen, M. F., Ngo, T., Russell, J. R., Sarkar, V., Serrano, M. J., Shepherd, J. C., Smith, S. E., Sreedhar, V. C., Srinivasan, H., and Whaley, J. The Jalapeño virtual machine. *IBM Systems Journal 39*, 1 (2000), 211–238.

[2] Arnold, K., and Gosling, J. *The Java Programming Language*. Addison Wesley, 1996.

[3] Faes, Ph., Christiaens, M., Buytaert, D., and Stroobandt, D. FPGA-aware garbage collection in java. In *2005 International Conference on Field Programmable Logic and Applications (FPL)* (Tampere, Finland, 1 2005), T. Rissa, S. Wilton, and P. Leong, Eds., IEEE, pp. 675–680.

[4] FAES, PH., CHRISTIAENS, M., AND STROOBANDT, D. Transparent communication between Java and reconfigurable hardware. In *Proceedings of the 16th IASTED International Conference Parallel and Distributed Computing and Systems* (Cambridge, MA, USA, 11 2004), T. Gonzalez, Ed., ACTA Press, pp. 380–385.

[5] GOTOH, O. An improved algorithm for matching biological sequences. *Journal of Molecular Biology 162*, 3 (1982), 705–708.

[6] HA, Y., VANMEERBEECK, G., SCHAUMONT, P., VERNALDE, S., ENGELS, M., AND DE MAN, H. Virtual Java/FPGA interface for networked reconfiguration. In *Proceedings of the ASP-DAC 2001 Asia and South Pacific Design Automation Conference* (Jan. 2001), pp. 558–563.

[7] MIGNOLET, J.-Y., VERNALDE, S., VERKEST, D., AND LAUWEREINS, R. Enabling hardware-software multitasking on a reconfigurable computing platform for networked portable multimedia appliances. In *ERSA 2002 Las Vegas* (June 2002).

[8] MINNAERT, B. Ontwerp van een hardwareversneller voor de vergelijking van DNA-sequenties. Master's thesis, Gent, 2005.

[9] SMITH, T. F., AND WATERMAN, M. S. Identification of common molecular subsequences. *Journal of Molecular Biology 147*, 1 (1981), 195–197.

[10] TRIMBERGER, S. Scheduling designs into a time-multiplexed FPGA. In *International Symposium on FPGAs* (1998), ACM, pp. 153–160.

[11] VASSILIADIS, S., WONG, S., GAYDADJIEV, G. N., BERTELS, K., KUZMANOV, G., AND PANAINTE, E. M. The Molen polymorphic processor. *IEEE Transactions on Computers* (November 2004), 1363–1375.