High-level programming of coarse-grained reconfigurable architectures

Zain-ul-Abdin

In Proceedings of 19th International Conference on Field Programmable Logic and Applications (FPL), Czech Republic, August 31 - September 2, 2009.

High-level Programming of Coarse-Grained Reconfigurable Architectures

Zain-ul-Abdin
Centre for Research on Embedded Systems (CERES),
Halmstad University, Halmstad, Sweden.

1. Introduction

Embedded signal processing systems are facing the challenges of increased computational demands. Reconfigurable architectures, which can be configured to form application-specific hardware, offer not only high degree of parallelism but also the possibility to dynamically allocate the resources during run-time. This allows the user to implement applications which are otherwise too large to be handled by a particular device. The reconfigurable computing devices have evolved over the years from the gate-level arrays to a more coarse-grained composition of highly optimized functional blocks or even program controlled processing elements, which are operated in a coordinated manner to improve performance and energy efficiency.

However, developing applications that employ reconfigurable architectures is constrained by the need for mastering low-level structural description languages and their lack of support for expressing dynamic reconfiguration. Traditionally, system developers have instead relied on using advanced synthesis tools and automatic parallelization techniques to lower the development costs, but these techniques lag in terms of achieved runtime performance.

2. Thesis Statement

We propose that, in order to meet high computational demands, the application development has to be based on suitable models of computations that will lead to scalable and reusable implementations. The models should enhance the understanding of the application and at the same time enable the developer to organize the computations so that they can be efficiently mapped to the target reconfigurable architecture. The goal of the thesis is to propose methods to program future coarse-grained reconfigurable architect-tures in a productive manner in such a way as to achieve energy efficient mapping with improved performance.

3. Research Approach

The approach used in our research is to first understand the class of coarse-grained reconfigurable architectures and identify the emerging trends, and then perform a study of relevant software models of computation. Based on the findings, we intend to perform some experiments by adopting selected well-known computation models to program some example coarse-grained architectures. We present our preliminary results obtained from one such target architecture, where we generate a single configuration at compile-time. Further, we suggest a framework based on the selected programming model that allows the developer to express the dynamic mode of reconfiguration.

4. Architectures and Computation Models for Coarse-Grained Reconfigurable Computing

We have surveyed the field of coarse-grained reconf-igurable computing [1], taking into account the arch-itectural characteristics of granularity, reconfigurability, and interconnection network. We have classified the coarse-grained reconfigurable architectures into four broad categories, i.e., Hybrid architectures, Arrays of functional units, Arrays of processors, Arrays of soft processors.

The study reveals a trend across the categories to keep reconfigurability at the interconnection network level and have heterogeneous programmable cores implemented with instruction streaming. As the number of cores increases, the complexity of distribution of clock increases too. Following the ideas of the VLSI community, the negative impacts of the global clock distribution in the processor arrays can be minimized by adopting the globally asynchronous locally synchronous (GALS) prin-ciple at a coarse-grained level. One of the emerging exam-ples of such an implementation is Ambric's massively parallel MIMD style array of RISC processors comm-unicating over a fabric of asynchronous message passing channels [2]. Therefore we chose Ambric to be the first target architecture for our experimental framework.

The computation models that are studied have the inherent ability to express communication explicitly, separated from the computations, which makes them suitable for exposing parallelism. The models that we consider are Stream Processing [3], Communicating Sequential Processes (CSP) [4], and Kahn Process Networks (KPN) [5]. The streaming model has the characteristics of being synchronous, while CSP and KPN are asyn-chronous. CSP uses unbuffered channels with message passing for describing communication between processing nodes, while channels between processes in KPN have unbounded buffering capability.

5. Programming Reconfigurable Processor Arrays in occam-pi

We propose CSP to be a suitable model of computation for the coarse-grained reconfigurable processor arrays that are based on the GALS principle because of its asynchronous nature and its available programming model such as occam. As a proof of concept we performed an experiment by using occam-pi [6] for programming the Ambric array of processors after implementing a compiler [7]. Occam-pi is an extension of classical occam that includes the mobility features of the pi-calculus. For each process in the occam-pi source code, the backend of the developed compiler produces Ambric's proprietary aJava and aStruct code, the latter to describe the communication with other processes. This work is part of our ongoing research whose initial studies are presented here.

An application study is performed in which the results of three different implementations of the 1D-DCT algorithm are compared on the basis of performance versus resource requirements [7]. The comparison reveals that the most parallelized version, which uses 34 processors, produces 27 times higher throughput when compared with the serialized implementation. In terms of lines of code metrics, the occam-pi source code is three times shorter than its corresponding aStruct and aJava code.

6. Discussion and Future Work

In the past the techniques used to program the studied architectures involve either low-level micro-coding or generating data-flow graphs from C-language source code. In our view, a better approach will be to base the application development on concurrent computation models such as CSP, KPN, or Stream processing. These models provide a representation that enhances the task of parallelization because of their exposure of explicit concurrency and strong encapsulation. We are focusing on computation models that support concurrent execution and offer possibilities for expressing reconfigurations, for instance the combination of CSP with the pi-calculus. These models allow separation of computations from the communication, so that the computational kernels can then be scheduled for reconfiguration without any data dependencies. Our experience so far is that these models allow optimizations and support correct design.

A significant property of reconfigurable architectures is their ability to undergo run-time reconfiguration. So far we have not dealt with making use of this property in our experimental framework. A natural continuation of this work is to extend the compiler platform to use the mobility features of occam-pi for implementing run-time reconfiguration in reconfigurable processor arrays. The mathematical modeling of the mobility features of occam-pi is based on pi-calculus. The mobility concept of the pi-calculus enables the movement semantics during assignment and communication, which means that the respective data has moved from the source to the target and afterwards the source loses the possession of the data. Mobile communication is introduced in the form of mobile channel types, and the data communicated on mobile channels has to be of the mobile data type. Channel type variables behave similarly to the other mobile variables. Once they are allocated, their communication means moving the channel-ends around the network. In terms of pi-calculus it has the same effect as if passing the channel-end names as messages. For expressing run-time reconfiguration, dynamic invocation of processes is necessary. In occam-pi concurrency can be introduced by not only using the classical PAR construct but also by dynamic parallel process creation using forking. We propose to use the extensions in the occam-pi language, such as channel direction specifiers, mobile data and channel types, dynamic process invocation, and process placement attribute, to express run-time reconfiguration of hardware resources in the programming model.

We would also like to evaluate our ideas further by providing a platform which can be used to achieve efficient implementations with respect to performance, energy and engineer efficiency on a range of coarse-grained reconfigurable architectures. The proposed framework includes extensions of the occam-pi language to support the use of heterogeneous reconfigurable processing elements and implementation of partitioning techniques in the compiler to adapt the generated code in a way that is best suited for the target architecture. Finally the framework is to be tested by implementing a number of signal processing applications of our interest and evaluating the results on different heterogeneous architectures.

7. References

[1] Zain-ul-Abdin, and B. Svensson, "Evolution in architectures and programming methodologies of coarse-grained reconfigurable computing", *Microprocessors and Microsystems*, Vol. 33, 2009, pp. 161-178.

[2] M. Butts, A.M. Jones, and P. Wasson, "A structural object programming model, architecture, chip and tools for reconfigurable computing", *FCCM*, 23rd April, 2007.

[3] R. Stephens, "A survey of stream processing", *Acta Informatica*, Vol. 34(7), 1997, pp. 491-541.

[4] C.A.R. Hoare, *Communicating sequential processes*, Prentice-Hall, 1985.

[5] G. Kahn, "The semantics of a simple language for parallel programming", *Information Processing*, North-Holland Publishing Company, 1974, pp. 471-475.

[6] P.H. Welch, and F.R.M. Barnes, “Communicating mobile processes: introducing Occam-pi”, *LNCS*, Springer Verlag, Vol. 3525, April 2005, pp. 175-210.

[7] Zain-ul-Abdin, and B. Svensson, “Using a CSP based programming model for reconfigurable processor arrays”, *ReConFig’08*, 3-5 December, 2008.