ADRES: An Architecture with Tightly Coupled VLIW Processor and Coarse-Grained Reconfigurable Matrix

Bingfeng Mei[1,2], Serge Vernalde[1], Diederik Verkest[1,2,3], Hugo De Man[1,2], and Rudy Lauwereins[1,2]

[1] IMEC vzw, Kapeldreef 75, Leuven, B-3001, Belgium
[2] Department of Electrical Engineering, Katholieke Universiteit Leuven, Belgium
[3] Department of Electrical Engineering, Vrije Universiteit Brussel, Belgium

Abstract. The coarse-grained reconfigurable architectures have advantages over the traditional FPGAs in terms of delay, area and configuration time. To execute entire applications, most of them combine an *instruction set processor*(ISP) and a reconfigurable matrix. However, not much attention is paid to the integration of these two parts, which results in high communication overhead and programming difficulty. To address this problem, we propose a novel architecture with tightly coupled *very long instruction word* (VLIW) processor and coarse-grained reconfigurable matrix. The advantages include simplified programming model, shared resource costs, and reduced communication overhead. To exploit this architecture, our previously developed compiler framework is adapted to the new architecture. The results show that the new architecture has good performance and is very compiler-friendly.

1 Introduction

Coarse-grained reconfigurable architectures have become increasingly important in recent years. Various architectures were proposed [1][2][3][4]. These architectures often comprise a matrix of functional units (FUs), which are capable of executing word- or subword-level operations instead of bit-level ones found in common FPGAs. This *coarse* granularity greatly reduces the delay, area, power and configuration time compared with FPGAs, however, at the expense of flexibility. Other features include predictable timing, a small configuration storage space, flexible topology, etc. However, the reconfigurable matrix alone is not capable of executing entire applications. Most coarse-grained architectures are coupled with processors, typically RISCs. The execution model of such hybrid architectures is based on the well-known 90/10 locality rule[5], i.e., *a program spends 90% of its execution time in only 10% of the code*. Some computational-intensive kernels are mapped to the matrix, whereas the rest code is executed by the processor. So far not much attention is paid to the integration of the two parts of the system. The coupling between the processor and the reconfigurable matrix is often loose, which is essentially two separated parts connected by a

P.Y.K. Cheung et al. (Eds.): FPL 2003, LNCS 2778, pp. 61–70, 2003.

communication channel. This results in programming difficulty and communication overhead. In addition, the coarse-grained reconfigurable architecture consists of components which are similar to those used in processors. This represents a major resource-sharing and cost-saving opportunity, which is not extensively exploited in traditional coarse-grained architectures.

To address the above problems, in this paper we presents a novel architecture called ADRES (*Architecture for Dynamically Reconfigurable Embedded System*), which tightly couples a VLIW processor and a coarse-grained reconfigurable matrix. The VLIW processor and the coarse-grained reconfigurable matrix are integrated into one single architecture but with two virtual functional views. This level of integration has many advantages compared with other coarse-grained architectures, including improved performance, a simplified programming model, reduced communication costs and substantial resource sharing. Nowadays, new programmable architecture can not succeed without good support for mapping applications. In our previous work, we built a compiler framework for a family of coarse-grained architectures [6]. A novel modulo scheduling algorithm was developed to exploit the loop-level parallelism efficiently[7]. In this paper, we present how this compiler framework can be adapted to the ADRES architecture. In addition, some new techniques are proposed to solve the integration problem of the VLIW processor and the reconfigurable matrix.

The paper is organized as follow. Section 2 describes the proposed ADRES architecture and analyzes its main advantages. Section 3 discusses how the compiler framework is ported to the ADRES architecture and some considerations of the compilation techniques. Section 4 reports experimental results. Section 5 covers related work. Section 6 concludes the paper and presents future work.

2 ADRES Architecture

2.1 Architecture Description

Fig. 1 describes the system view of the ADRES architecture. It is similar to a processor with an execution core connected to a memory hierarchy. The ADRES core(fig 3) consists of many basic components, including mainly FUs and register files(RF), which are connected in a certain topology. The FUs are capable of executing word-level operations selected by a control signal. The RFs can store intermediate data. The whole ADRES matrix has two functional views, the VLIW processor and the reconfigurable matrix. These two functional views share some physical resources because their executions will never overlap with each other thanks to the processor/co-processor model. For the VLIW processor, several FUs are allocated and connected together through one multi-port register file, which is typical for VLIW architecture. Compared with the counterparts of the reconfigurable matrix, these FUs are more powerful in terms of functionality and speed. They can execute more operations such as branch operations. Some of these FUs are connected to the memory hierarchy, depending on available ports. Thus the data access to the memory is done through the load/store operation available on those FUs.