Toward a Run-time Support System for Molen Hardware Organization

Mojtaba Sabeghi, Koen Bertels

Computer Engineering Lab, Delft University of Technology, Delft, the Netherlands

---

ABSTRACT

In this paper we propose a runtime support system for the Molen Hardware Organization. This runtime system is responsible for task and resource management based on the runtime changing conditions. We present a resource allocation strategy as well as a few decision metrics. Furthermore, we discuss some of the implementation issues on the Molen platform.

KEYWORDS: Runtime System; Molen Hardware Organization; Resource Management

1. Introduction

With development of reconfigurable computers [1] containing FPGAs with millions of systems gates, it is now feasible to consider the possibility of serving multiple concurrent applications executing on a shared logic area. This will improve the resource utilization and reduce the costs. However, it will increase the degree of complexity in order to manage the shared resources. Needless to say, dynamic and partial reconfiguration is an important factor in sharing the FPGA logic area. Run-time reconfiguration provides the ability to change the configuration not only between applications, but also within a single application.

Considering multiple application on a system which are executing concurrently, there must be mechanisms and policies that manage the competition for resources between different applications and resolve the conflicts. However, current efforts for designing reconfigurable systems are not flexible as they manage the hardware with static and design time decisions based on fixed rules.

In the context of the Molen hardware platform [2] and using Delft Workbench [3] tool chain, the FPGA allocation is currently static and is being done by the Molen compiler. As it was mentioned before, the main problem of this approach is the management of the reconfigurable hardware in a multi-tasking environment.

In this paper we propose a runtime system that efficiently operate the system and resolve the conflicts between executing tasks. In fact, this system is responsible for task and resource management. The reminder of the paper is organized as follows. In section 2, we briefly present the Molen hardware organization. In section 3, we discuss the runtime system specification and implementation requirements. And, we conclude the paper in section 4.

2. Molen Hardware Organization

Within the Molen concept, a general purpose core processor controls the execution and reconfiguration of reconfigurable coprocessors (RP), tuning the latter to various application specific algorithms. An operation, executed by the RP, is divided into two distinct phases: *set* and *execute*. In the set phase, the RP is configured to perform the required operation and in the execute phase the actual execution of the operation is performed. This decoupling allows the set phase to be scheduled well ahead of the execute phase, thereby hiding the reconfiguration latency. [4]

Molen hardware organization as it is depicted in figure **1** consists of two parts; the general purpose processor (GPP) and the reconfigurable processor (RP) usually implemented on an FPGA. Another key component is the Arbiter which performs a partial decoding of the instructions received from the instruction fetch unit and issues them to the appropriate processor (GPP or RP). The Exchange Registers (XREGs) are used for data communication between the Core Processor and Reconfigurable Processor. Parameters are moved from the register file to the XREGs and the results stored back from the XREGs in the register file. [2]

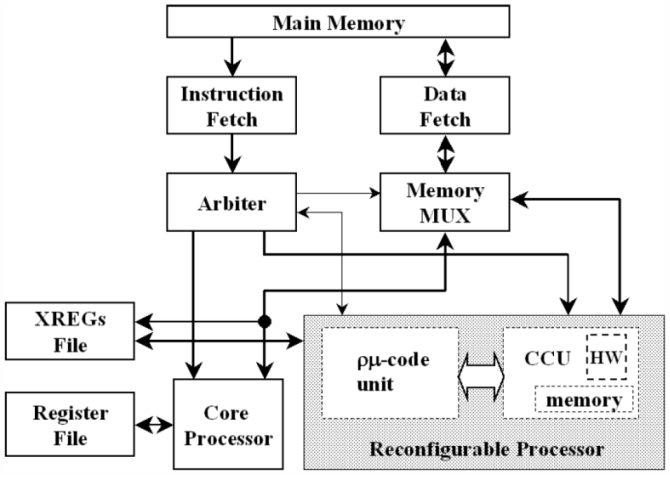

Figure 1 Molen Hardware Organization

3. Runtime System Specification and Implementation

Currently, in the context of Delft Workbench (DWB) [3], FPGA area allocations are being done at design and compile time using the Molen compiler [5]. The profiler identifies the computing intensive parts of the application(s) and annotates the source code. These computing intensive parts can be compiled to VHDL manually or using a C to VHDL compiler called DWARV. Then, the Molen compiler at compile time decides how to allocate the area based on the profile information and also area availability. However as it was mentioned before, in a multitasking system with the dynamic changing environment it is very difficult to manage and allocate the resources at runtime.

To solve this problem, we proposed a consistent, protected, low overhead interface which decides how to allocate the hardware at run-time based on the status of the system.

To decide, this layer needs some information. A part of this information can be provided from the design time ─for example the Configuration Call Graph (CCG)─ and embedded in to the binary or provided in a separate file. The Configuration Call Graph (CCG) is a directed graph that can be used by the runtime system to perform the allocation of the hardware reconfigurations. Its nodes are the operations that can be run in the hardware and the edges represent the dependencies.

The allocation is based on configuration latency, total execution time in hardware and software, the configuration order and availability of the hardware. Some heuristics can also be beneficial among which we can mention most frequency used configuration, best speed up configuration, lowest power consumption configuration and even a combination of them. Using the CCG, we can also look at the future and obtain a near optimal schedule.

For each operation annotated by the profiler, the Molen compiler put a call to the hardware besides the software implementation of that operation. This can be done using Molen set and execute instructions. At runtime, the runtime system can choose to execute either the software version or the hardware call. By means of this mechanism we can avoid runtime binary transformation.

The Molen set and execute instruction are very important for the runtime system. Since, we need a mechanism to inspect and instrument the code to recognize the computation intensive parts. And, this is exactly what Molen set and execute instructions do. However, these instructions do not configure or execute anything on the hardware. The set just informs the runtime system of a possible future call to a hardware and the execute instruction is a signal to system to run the hardware if that is appropriate.

The runtime system receives all the sets and executes form all the applications and based on the allocation strategy and allocation metrics decides which one should be run on hardware and which one in software. It has to keep track of the already configured modules and whether they are being used or not. This can be achieved by a simple data structure called Hardware Control Block (HCB). This is a table containing the relevant information of each hardware module and the process to which they belong. Also, it can contain some extra information about the usage trends of that hardware module. They can be used in the heuristic part of the decision procedure.

Another important issue that must be addresses by the runtime system is the XREGs. The XREGs are being used for data exchange between hardware and software. The runtime system is responsible for allocating them to each process. On the other hand, as it is not known before which operation will be run on the hardware, the runtime system most also take an active role in the parameter transfer through the XREGs.

4. Conclusion and Future Works

In this paper we presented a runtime system for Molen hardware platform. The resource allocation strategy as well as some decision metrics was discussed. In the future, we will work on a detailed specification of this runtime system. Furthermore, we will compare it with current static and compile time approaches and also with other runtime techniques.

5. References

[1] Katherine Compton, Scott Hauck, Reconfigurable computing: a survey of systems and software, *ACM Computing Surveys,* vol. 34, no. 2, pp 171-210, June, 2002

[2] Stamatis Vassiliadis, Stephan Wong, Georgi Gaydadjiev, Koen Bertels, Georgi Kuzmanov, Elena Moscu Panainte, The MOLEN Polymorphic Processor, *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1363-1375, November, 2004.

[3] Koen Bertels, Stamatis Vassiliadis, Elena Moscu Panainte, Yana Yankova, Carlo Galuzzi, Ricardo Chaves, Georgi Kuzmanov, Developing applications for polymorphic processors: the Delft Workbench, Technical Report, pp. 7, January, 2006

[4] Stamatis Vassiliadis, Georgi Gaydadjiev, Koen Bertels, Elena Moscu Panainte, The Molen Programming Paradigm, *Proceedings of the Third International Workshop on Systems, Architectures, Modeling, and Simulation, pp. 1-10, July 2003*

[5] Elena Moscu Panainte, Koen Bertels, Stamatis Vassiliadis, The Molen Compiler for Reconfigurable Processors, *ACM Transactions in Embedded Computing Systems*, vol. 6, no. 1, February, 2007.