MAPPING HIGH LEVEL ALGORITHMS ONTO MASSIVELY PARALLEL RECONFIGURABLE HARDWARE

Issam Damaj
Centre for Applied Formal Methods.
London South Bank University
103 Borough Road
London, SE1 0AA. U.K.
damajiw@sbu.ac.uk

John Hawkins
Centre for Applied Formal Methods
London South Bank University
103 Borough Road
London, SE1 0AA. U.K.
john.hawkins@sbu.ac.uk

Ali Abdallah
Centre for Applied Formal Methods
London South Bank University
103 Borough Road
London, SE1 0AA. U.K.
A.Abdallah@sbu.ac.uk

Abstract

The main focus of this paper is on implementing high level functional algorithms in reconfigurable hardware. The approach adopts the transformational programming paradigm for deriving massively parallel algorithms from functional specifications. It extends previous work by systematically generating efficient circuits and mapping them into reconfigurable hardware. The massive parallelisation of the algorithm works by carefully composing "off the shelf" highly parallel implementations of each of the basic building blocks involved in the algorithm. These basic building blocks are a small collection of well-known higher order functions such as map, fold, and zipwith. By using function decomposition and data refinement techniques, these powerful functions are refined into highly parallel implementations described in Hoare's CSP. The CSP descriptions are very closely associated with Handle-C program fragments. Handle-C is a programming language based on C and extended by parallelism and communication primitives taken from CSP. In the final stage the circuit description is generated by compiling Handle-C programs and mapping them onto the targeted reconfigurable hardware such as the Celoxica RC-1000 FPGA system. This approach is illustrated by a case study involving the generation of several versions of the matrix multiplication algorithm.

Keywords

Reconfigurable Architectures, FPGAs, Rapid Development, Handle-C, Functional Specification, CSP.

INTRODUCTION

At one extreme of the computing spectrum, computing systems based on the traditional von Neumann model provide a single and generic computational medium for applications with diverse characteristics. These systems are known as general purpose processors (GPPs). At the other end of the spectrum there are systems with architectures customized for particular applications. These systems are built around one or more Application Specific Integrated Circuits or ASICs. In certain application areas, software executed on a single sequential processor no longer meets our ever increasing efficiency requirements. Besides, the direct architecture algorithm mapping restricts the range of applicability of ASIC-based systems. Consequently, this led to the introduction of reconfigurable computing (RC) combining the flexibility of general-purpose processors and the high performance of ASICs [1-9]. Field Programmable Gate Arrays (FPGAs), an instance of RCs, has recently enabled RC chips with Millions of gates (Xilinx) affording more scalability and cost effectiveness due to hardware reuse. FPGAs offer much flexibility for the design of integrated circuits (ICs) chips for parallelism.

Generally, parallelism and implementation in hardware provide us with two alternatives that can often deliver very dramatic improvements in efficiency. With the emergence of such reconfigurable hardware chips, the presence of a rapid development environment for these scalable hardware circuits is very useful. Moreover, it would constitute the cornerstone solution for the ever-increasing need for more: efficiency, scalability and flexibility in realizing massively parallel algorithms for a wide area of applications [2].

The proposed rapid development model (RDM) adopts the transformational programming approach for deriving massively parallel algorithms from functional specifications (See Figure 1) [1-4]. The functional notation is used for specifying algorithms and for the reasoning about them. This is usually done by carefully combining small number of high order functions (like map, zip and fold) to serve as the basic building blocks for writing high-level programs. The systematic methods for massive parallelisation of algorithms work by carefully composing "off the shelf" massively parallel implementation of each of the building blocks involved in the algorithm. The emphasis in this method is on correctness, scalability and reusability.

To describe parallelism we use Hoare's CSP that allows issues of immense practical importance (such as data distribution, network topology, and locality of communications) to be carefully reasoned about [12]. Relating the Functional Programming and CSP fields gives the ability to exploit a well-established functional programming paradigms and transformation techniques in order to develop efficient CSP processes.

0-7803-7983-7/03/$17.00 ©2003 IEEE