Uebersetzung funktionaler Sprachen für den ADAM-Parallelrechner

Diss. ETH Nr. 9880

Uebersetzung funktionaler Sprachen für den ADAM-Parallelrechner

ABHANDLUNG
zur Erlangung des Titels
DOKTOR DER TECHNISCHEN WISSENSCHAFTEN
der
EIDGENÖSSISCHEN TECHNISCHEN HOCHSCHULE
ZÜRICH

vorgelegt von
Stephan B. Murer
Dipl. Informatik-Ing. ETH
geboren am 26. Februar 1963
von Beckenried (NW) und Zürich

angenommen auf Antrag von
Prof. Dr. A. Kündig, Referent
Prof. Dr. H. Burkhart, Korreferent
1992

Abstract

In this dissertation the necessary concepts to generate code for a new class of parallel computers, i.e. coarse-grain dataflow machines (or multi-threaded architectures, as they are called more recently), are demonstrated on the basis of a simple functional programming language. In a first, introductory chapter, an overview is given over some general requirements for parallel computers (scalability, programmability, general-purpose applicability). The analysis of these requirements has led to the concept of coarse-grain dataflow machines or multi-threaded architectures, as they have been called more recently. In the next two chapters the ADAM architecture as the target and the functional language MFL as the source for the compilation are defined as far as necessary. Finally, the concept of hierarchical acyclic dataflow graphs as intermediate code in the compiler is presented.

In the central part of the thesis we discuss the several novel code generation methods for coarse-grain dataflow computers. In this context the following problems have been solved:

1. The choice of a reasonable trade-off between parallelism and overhead for management and synchronization of parallel activities.

2. The efficient implementation of large data structures for functional languages on a parallel machine.

3. The choice of an optimum instruction sequence to fully exploit the potential parallelism of the machine.

Chapters in this part share the same structure: First the problem is introduced, then methods to solve it are shown and finally the solutions are validated with simulation experiments and compared to similar results in the literature.

In the following chapter, we discuss the implementation of the MFL compiler and its integration into a new kind of programming environment. After some conclusions and a number of proposals for further research, three appendices may be found, containing the syntax of MFL in the usual form, all source codes of the MFL benchmark programs and an example assembler code of a runtime routine for the management of large data structures.

Keywords: functional languages, data-flow architectures, parallel processors, multi-threaded architectures, code generation, optimization

Kurzfassung

In dieser Dissertation werden auf der Basis einer einfachen funktionalen Programmiersprache die notwendigen Konzepte aufgezeigt, um Code für eine neuartige Klasse von Parallelrechnern - die grobgranularen Datenflussmaschinen - zu generieren. In einem ersten, einführenden Kapitel wird ein Ueberblick über allgemeine Anforderungen an Parallelrechner (Skalierbarkeit, Programmierbarkeit und allgemeine Anwendbarkeit) gegeben. Die Analyse dieser Anforderungen hat zum Konzept der grobgranularen Datenflussrechner geführt. Die ADAM-Architektur als Zielarchitektur und die funktionale Sprache MFL als Quellensprache für die Uebersetzung werden soweit nötig beschrieben. Es wird dann das Konzept hierarchischer, azyklischer Datenflussgraphen als Zwischencode für funktionale Programme eingeführt.

Im zentralen Teil der Arbeit wird eine Reihe von neuen Codegenerierungstechniken für grobgranulare Datenflussrechner diskutiert. In diesem Kontext wurden die folgenden Probleme gelöst:

1. Die Wahl eines vernünftigen Kompromisses zwischen Parallelität und dem notwendigen Zusatzaufwand zur Verwaltung und Synchronisation paralleler Aktivitäten,

2. Die effiziente Implementation grosser Datenstrukturen für funktionale Sprachen auf einer parallelen Maschine,

3. Die Wahl einer optimalen Sequenz von Instruktionen, so dass das Parallelitätspotential der Maschine möglichst gut ausgenutzt werden kann.

In allen Kapiteln dieses Teils werden am Anfang die Probleme eingeführt, dann die Methoden zu deren Lösung gezeigt, die Lösungen anhand von Simulationsexperimenten validiert und schliesslich mit ähnlichen Arbeiten aus der Literatur verglichen.

Im nächsten Kapitel wird die Implementation des MFL-Compilers und seine Integration in eine neuartige Programmierumgebung diskutiert. Nach einigen Schlussfolgerungen und Hinweisen auf mögliche Folgearbeiten folgen noch drei Anhänge mit einer Syntax von MFL in der üblichen Notation, den MFL-Quellenprogrammen aller in den Experimenten verwendeten Beispielen und dem Assembler-Code für einen Teil des Laufzeitsystems.

Schlüsselbegriffe: Funktionale Sprachen, Datenfluss-Architekturen, Parallelrechner, Codegenerierung, Optimierung