Compiling F# Functions for GPUs

Harald Steinlechner, 0825851

Kurzfassung

Die rasante Weiterentwicklung von Hard- und Software im Bereich der Grafikprozessoren führte zu verschiedenen Ansätzen, die hohe Rechenleistung dieser Chips auch für andere Zwecke als Realtime-Rendering zu verwenden. Leider unterscheidet sich die Architektur der GPUs (Graphics Processing Units) grundlegend von weit verbreiteten Desktop-Prozessoren, was die Verwendung erschwert.

Programmierung dieser Architekturen ist zur Zeit mittels spezieller Sprachen möglich - aus Software-Engineering-Sicht erweist sich dies häufig als problematisch. Zum einen durch die hohe Komplexität paralleler Algorithmen, zum anderen durch den Mangel ausgereifter Entwicklungsumgebungen für eben diese Sprachen. Außerdem ist die Wiederverwendbarkeit gegenüber bestehendem und zukünftigen Code sehr eingeschränkt, da sich die APIs ständig ändern. Dies sind Gründe dafür, dass das Potential der Hardware selten ausgeschöpft wird. Funktionale Programmierung bietet hohe Wiederverwendbarkeit, Abstraktion und motiviert Parallelisierung. Diese Arbeit beschäftigt sich mit der Übersetzung funktionaler Sprachen auf imperative Sprachen, welche von GPUs ausgeführt werden können um die Vorteile funktionaler Hochsprachen mit der hohen Leistung von GPUs zu vereinen.

Abstract

Rapid improvement in programmability and hardware flexibility unleashed the power of Graphics Processing Units (GPUs) to a broad field of applications exploiting parallel program execution. Parallel programming, inherently a difficult problem in computer science coupled with a vast number of architectures, languages and lack of classical software development tools causes troubles to software engineers - eventually withholding developers from actually using GPUs effectively. Enhancing GPUs programmability or even exploiting its performance implicitly by automatic code generation is a very active research topic currently. Special purpose languages often suffer flexible development toolchains, parallelization libraries often lack functionality and flexibility.

Also from and economoic point of view code reusage and stability plays a crucial role in software development. Functional programming offers composability, abstraction and is well suited for parallel programming. We present an efficient F# compiler targeting GPU programs like shaders - nevertheless our approach is general and also supports other imperative languages as well. Our work motivates a large set of applications, libraries and programming models supporting developers to unleash the power of GPUs.