

AMD Stream Computing

AMD Stream Computing Environment (SCE)

Ein kurzer Überblick

Übersicht

- ▶ **Einleitung**
 - ▶ AMD Stream SDK
 - ▶ Brook+
 - ▶ CAL
 - ▶ zusätzliche Tools
- ▶ **AMD GPU Architektur (R600 Serie)**
 - ▶ doppelte Genauigkeit
- ▶ **AMD GPU Architektur (R700 Serie)**
- ▶ **Vergleich mit CUDA**
- ▶ **Fazit**
- ▶ **Beispiel: Matrix Multiplikation (Brook+ - Variante)**

AMD Stream SDK

- ▶ aktuelle Version: 1.01 BETA
- ▶ beinhaltet:
 - ▶ Brook+
 - ▶ Compute Abstraction Layer (CAL)
- ▶ unterstützte Betriebssysteme:
 - ▶ Windows XP (32-bit/64-bit)
 - ▶ Linux (32-bit/64-bit)
- ▶ in Zukunft geplant:
 - ▶ Windows Vista (von nVidia CUDA bereits unterstützt) und Apple Mac OS Support
- ▶ benötigt:
 - ▶ ATI Grafikkarte
 - ▶ ATI Radeon HD 2xxx Serie
 - ▶ ATI Radeon HD 3xxx Serie
 - ▶ ATI FireStream Serie (Konkurrenzprodukt zu nVidia Tesla)
 - ▶ aktueller Grafikkartentreiber
 - ▶ C/C++ Compiler
- ▶ Einstieg über Manuals (teilweise fehlerhaft), AMD Developer Forum und Developer Kontakt per E-mail

Begriffsdefinition

- ▶ **SIMD (Multi-)Prozessor**
 - ▶ oft als SIMD Array bezeichnet
 - ▶ besteht aus einem Verbund von 16 Shaderprozessoren (SPU's)

Brook+



- ▶ basiert auf der Programmiersprache Brook, die von der Stanford University entwickelt wurde
- ▶ von AMD verbesserte und mit der
 - ▶ AMD GPU Architektur
 - ▶ AMD CAL runtimeabgestimmte Variante, um die AMD Streaming Prozessoren (SIMD Prozessoren) optimal zu nutzen
- ▶ Vorteile:
 - ▶ high-level language
 - ▶ stark angelehnt an C/C++ Syntax und einfach zu erlernen
 - ▶ Entwicklern wird von AMD empfohlen Brook+ zu verwenden

Brook+



- ▶ Brook+ Code muss in C++ Code umgewandelt werden über den Aufruf von `brcc.exe` (Release Version) oder `brcc_d.exe` (Debug Version)
 - ▶ C/C++ Code wird dabei unverändert übernommen
- ▶ Einbinden von include- und library-Verzeichnissen nicht vergessen
- ▶ zusätzliche Abhängigkeiten:
 - ▶ `brook.lib` (Release Version)
 - ▶ `brook_d.lib` (Debug Version)

- ▶ **lower-level runtime API**
 - ▶ ermöglicht hardwarenahe Programmierung des Kernel in Assemblersprache IL
- ▶ **Vorteile:**
 - ▶ mehr Möglichkeiten zur Optimierung im Vergleich zu Brook+
- ▶ **Nachteile:**
 - ▶ Einarbeitung sehr zeitaufwendig

zusätzliche Tools

- ▶ **GPU Shader Analyzer**

- ▶ bietet u.a. die Möglichkeit einen Brook+ Kernel in IL Assemblercode umzuwandeln

- ▶ **GPU CodeAnalyzer**

- ▶ bietet Unterstützung bei der Performancesteigerung

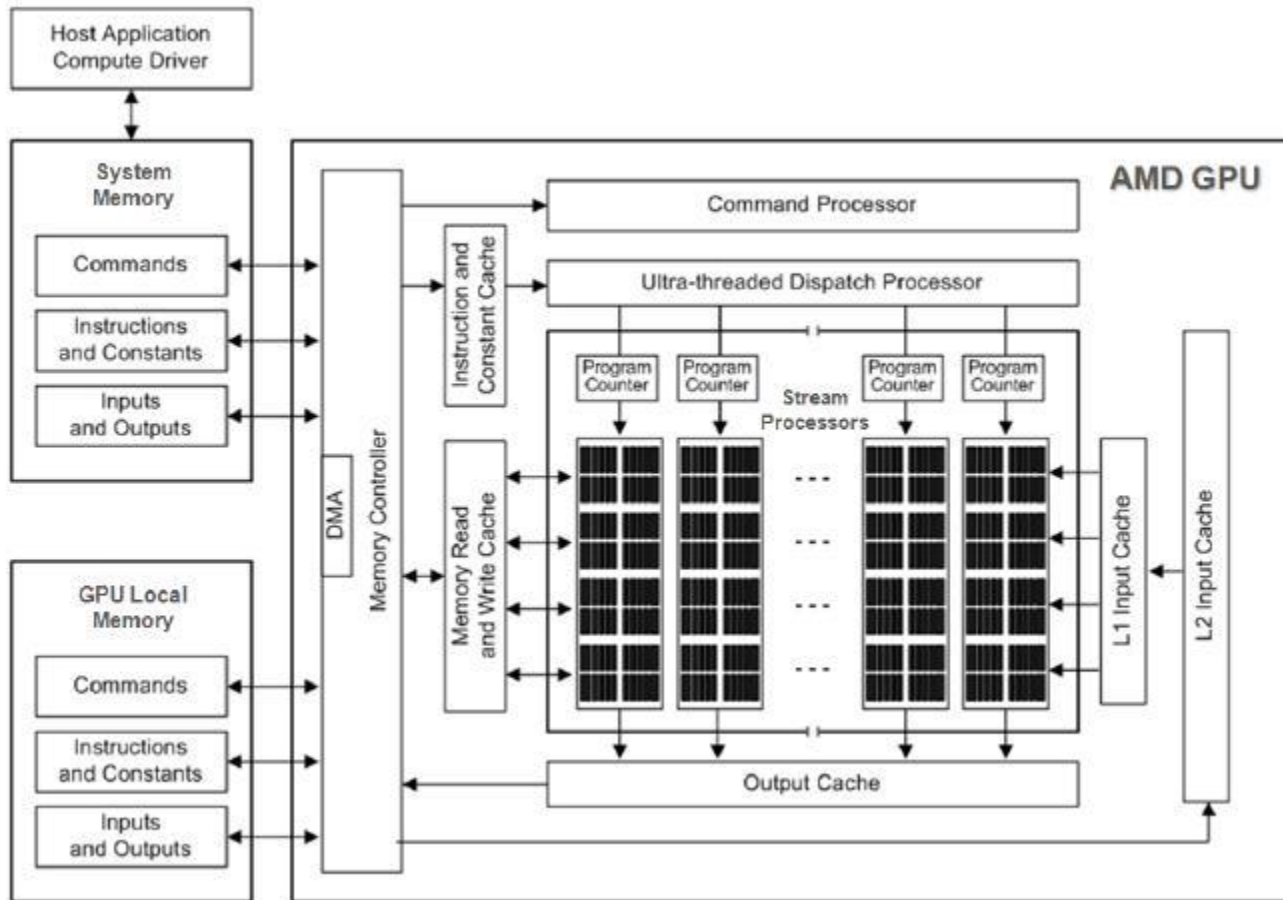
- ▶ **ACML Bibliothek**

- ▶ von AMD bereitgestellte Bibliothek, die einige effizient implementierte Methoden bereitstellt, die für AMD64 Systeme optimiert wurden
- ▶ Beispiel: FFT

Begriffsdefinitionen

- ▶ **Thread**
 - ▶ Instanz eines Kernels der auf einer SPU ausgeführt wird
- ▶ **Block**
 - ▶ Zusammenfassung von mehreren Threads
- ▶ **Domain**
 - ▶ eine Art Karte, die Threads im Outputstream einen Platz zuordnet

AMD GPU Architektur (R600 Serie)



AMD GPU Architektur (R600 Serie)

- ▶ 4 separate, unabhängige SIMD (Multi-)Prozessoren
- ▶ Jeder SIMD Prozessor besteht aus 16 Shaderprozessoren (SPU's)
- ▶ Jede SPU kann parallel 4 Threads ausführen im Blockformat 2 x 2, um Outputstream zu berechnen
 - ▶ Jeder SIMD Prozessor kann parallel 64 Threads bzw. 16 Blöcke der Größe 2 x 2 berechnen
- ▶ vordefinierte Reihenfolge der Abarbeitung von Elementen in Streams
 - ▶ Brook+ kann darauf keinen Einfluss nehmen im Gegensatz zur CAL runtime API
- ▶ 8192 32-bit Register vorhanden

AMD GPU Architektur (R600 Serie)

- ▶ GPU verfügt über lokalen high-speed DRAM Speicher (GDDR4)
- ▶ Unabhängige SIMD Prozessoren können entweder auf lokalen Speicher zugreifen, um Daten zu transferieren oder Kernel ausführen
 - ▶ Kein Zugriff auf anderen Speicher möglich!
- ▶ Caches sind vorhanden, die standardmäßig die Daten in der Reihenfolge enthalten, wie sie von den Threads gebraucht werden
 - ▶ Gruppierung wie bei Threads quadratisch
 - ▶ Brook+ kann darauf keinen Einfluss nehmen im Gegensatz zur CAL runtime API
 - ▶ funktioniert, weil Brook+ den Outputstream nach fester Reihenfolge berechnet

doppelte Genauigkeit

- ▶ bei R670 Architektur (und abwärts) werden zwei 32-bit Register belegt
- ▶ Architektur R700 besitzt 64-bit Register
- ▶ im SDK stehen folgende Double-Typ Varianten zur Verfügung
 - ▶ Double (einfacher double Wert)
 - ▶ Double2 (2 dimensionaler Vektor)
 - ▶ Double4 (4 dimensionaler Vektor)

AMD GPU Architektur (R700)

- ▶ aktuelle R700 Architektur besitzt 800 Scalar Prozessor Einheiten (SPU's)
- ▶ verbesserte Speicheranordnung und Bussystem um Transfer zwischen SPU und local memory zu beschleunigen
- ▶ GDDR5 Speicher
- ▶ 64-bit Register
- ▶ nahezu volle Umsetzung des IEEE 754
 - ▶ double precision
 - ▶ „nan“ vorhanden

Vergleich mit CUDA



- ▶ SDK's ermöglichen beide stabile Implementierung und sind sich sehr ähnlich
 - ▶ lower-level API für direkten Zugriff auf Hardware, um Optimierungspotential voll zu nutzen
 - ▶ ebenfalls einfacher Zugang über high-level language möglich
 - ▶ ähnliche Syntax
- ▶ AMD Stream SDK bietet gegenüber CUDA bereits die Möglichkeit doppelte Genauigkeit bei Gleitkommaarithmetik zu verwenden
- ▶ CUDA hat die größere Entwicklergemeinschaft, sodass Einstieg leichter fällt
 - ▶ hilfreiches Forum
 - ▶ gute Dokumentation

Fazit

- ▶ Der Unterschied bei der Softwareumgebung ist minimal (double precision Unterstützung für CUDA im nächsten Release)
- ▶ Entscheidend für die Zukunft wird sein, welche Hardwarearchitektur, die größeren Vorteile aufweist:
 - ▶ Leistung
 - ▶ Threadverwaltung
 - ▶ Speicherverwaltung
 - ▶ IEEE 754 Umsetzung
 - ▶ ...
- ▶ Hardware wird Determinante sein, nicht die Softwareumgebung

Beispiel: Matrix Multiplikation

- ▶ implementiert in Brook+
- ▶ Umwandlung des Brook+ Code in äquivalenten C++ Code über Brook+ Compiler
- ▶ Daten befinden sich im local memory der Grafikkarte
 - ▶ SPU hat schnellen Zugriff auf benötigte Daten über Caches
- ▶ Aufgabe: Multiplikation von Matrizen gefüllt mit double-Werten

Bibliographie

▶ Weblinks

- ▶ www.wikipedia.de
- ▶ <http://ati.amd.com/technology/streamcomputing/>
- ▶ http://www.nvidia.com/object/cuda_home.html#

ENDE
