Inhalt

* Deckblatt
* Eidesstaatliche Erklärung
* Preface (Vorwort)
* Abstract (Kurzfassung)
* Introduction
  + Zielsetzung
    - Was zeigt dieses Dokument?
      * GPGPU Beschleunigung mit OpenCL
      * Beispielimplementierungen von Scan, Sort und Matrix Mul.
    - Was soll bei meiner Arbeit rauskommen?
      * Welche Algorithmen profitieren von GPU Beschleunigung
        + In konkreten Fällen?
        + Immer?
      * Mehraufwand einer GPU Implementierung im Vergleich mit CPU?
    - Wie komme ich zu diesem Ziel?
      * Algorithmen ausgewählt
      * Bestehende Implementierungen eingebunden
      * Teilweise Algorithmen selbst implementiert
      * Referenzimplementierungen für die CPU erstellt
      * Benchmarks auf verschiedener Hardware
  + Motivation
    - Warum ist dieses Ziel wichtig? (für alle?)
    - Was ließe sich dadurch verbessern?
      * Um wie viel ließe es sich verbessern?
    - Wem würde das helfen?
      * Speziellen Unternehmen
      * Spezielle Domänen/Anwendungsgebiete?
      * Allen?
  + Kapitelübersicht
* Einführung in OpenCL
  + Historischer Rückblick
    - Entwicklung von Grafikhardware
    - Shader Programmierung (als erste Möglichkeit eingeschränkt für GPU's zu programmieren, Focus auf Grafikberechnungen)
    - GPGPU Ansätze (Missbrauch von Shadern für allgemeine Probleme, vgl. Sort mit GLSL Vertex Program)
    - Technologien (Cg, CUDA, OpenCL, DirectCompute)
  + Was ist OpenCL?
    - Offener Industriestandard (keine Implementierung/Bibliothek)
    - Khronos (spezifiziert API)
    - Vendors (implementieren API)
    - Komponenten
      * API Spezifikation
      * OpenCL C Language
  + Anforderungen
    - GPU (and CPU) Hardware
    - Treiber
    - SDK (Header, Bibliotheken (C/C++))
    - Bindings für andere Programmiersprachen
  + Hardware Architekturen
    - Grafikkarten-/Prozessorarchitektur
    - Unterschiede/Gemeinsamkeiten
  + API
    - Aufbau eines OpenCL Programs
    - Abstraktion der Hardware in OpenCL
      * Platforms, Devices, CommandQueues, Buffer, Programs, Kernels
    - Minimalistisches Codebeispiel (ohne Kernel)
  + Kernel execution
    - Was ist ein Kernel?
    - Wie werden Kernel ausgeführt?
      * Work groups/items
        + Hardware threads/Synchronisierung
      * Wahl der richtigen work group size
        + Auswirkungen auf Performance
        + Konsequenzen für die Programmierung
  + Memory model
    - Speicherbereiche auf der GPU
      * Global, local, constant, private
      * Performance
      * Wie werden diese angesprochen?
      * Bedeutung für die Kernel Programmierung
      * Generelle Designentscheidungen
        + Paralleler Speicherzugriff
        + Shared memory zum Cachen/Synchronisieren
        + Bench conflicts
    - Datentransfer
      * Wie werden Daten zur GPU befördert?
  + Begriff von Hardwareherstellern (AMD, NVIDIA)
* State of the Art
  + Welche Programme/Produkte/Unternehmen setzen OpenCL ein?
  + Wie weit ist OpenCL verbreitet?
  + Anwendungsgebiete
    - Von OpenCL
    - Von GPGPU Beschleunigung
    - WebCL
* Implementierung
  + Matrix multiplication
    - Naiv
  + Parallel prefix sum (scan)
    - Apple
    - GPU Gems
  + Sorting
    - AMD (Radix sort)
    - Bealto (Bitonic sort)
    - Bekannte CPU Algorithmen (Quicksort, Radixsort)
* Benchmarks
  + Vergleiche zwischen den Implementierungen
    - Laufzeit
    - Speicherkomplexität
    - CPU vs. GPU
  + Vergleiche auf verschiedener Hardware
* Conclusion
  + Wo kann OpenCL sinnvoll eingesetzt werden?