TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

# LEISTUNGSVERGLEICH VON FPGA, GPU UND CPU FÜR ALGORITHMEN ZUR BILDBEARBEITUNG

## PROSEMINAR INF-B-610

Dominik Weinrich
dominik.weinrich@tu-dresden.de

Dresden, 30.11.2017

# Gliederung

- Motivation
- Aufbau und Hardware
- 2D Filter
- Stereo Vision
- k-Means
- Diskussion der Ergebnisse

# Motivation

- Hohe Parallelität bei Bildbearbeitungsalgorithmen
- CPU bietet SIMD Befehle
  - Mehrere Operationen in einem Taktzyklus
  - Sehr hohe Taktrate, aber limitiert durch Größe des Caches
- FPGA bietet auf Grund der Flexibilität eine hochgradig optimierbare Parallelisierbarkeit
  - Für jede Anwendung kann der bestmögliche Schaltkreis realisiert werden
  - Auf Grund niedriger Betriebsfrequenz muss Anzahl der Operationen minimal gehalten werden

# Motivation

- Moderne GPU hat eine hohe Anzahl an Kernen
  - Langsamere Taktrate als CPU, aber schneller als FPGA
  - Höhere Parallelität als CPU
  - Kerne sind gruppiert und Datenaustausch zwischen Gruppen ist sehr langsam und Speicher jedes Kerns begrenzt
    - GPUs können nicht die selben Algorithmen wie FPGAs ausführen
    - Um hohe Leistung der GPUs zu nutzen, müssen die Algorithmen so entwickelt werden, dass deren Limitierungen umgangen werden

# Aufbau und Hardware

- Vergleich veröffentlicht auf internationaler Konferenz zu Field Programmable Logic and Gate Arrays (FPL) 2009
- RAM: 4GB
- Zeiten um Bild aus RAM zu laden werden nicht berücksichtigt
- Ergebnis ist Mittel aus 1000 Läufen
- CPU: Intel Core 2 Extreme QX6850 (3GHz, 4 Kerne, 8MB L2 Cache, 8 Operationen auf 128 Bit Daten gleichzeitig)
- FPGA: Xilinx XC4VLX160 (auf 100MHz begrenzt)
- GPU: XFX GeForce GTX 280 1024MB DDR3

# Hardware – Nvidia GTX 280

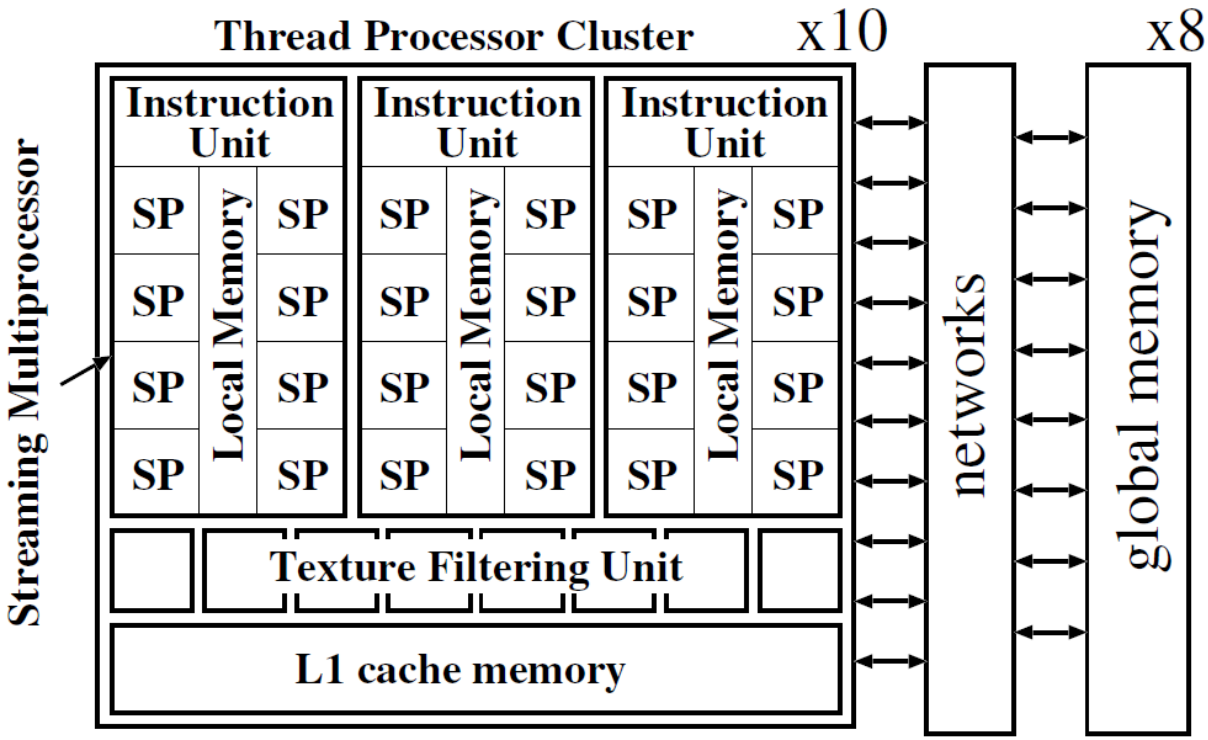


**Abb. 1:** Übersicht über die Nvidia GTX 280 [1]

# 2D Filter

- Filter auf zweidimensionaler Datenmenge (z.B. Weichzeichner)
- Berechnung für 2D Filter:

$$S(x,y) = \sum_{dx=-w}^{w} \sum_{dy=-w}^{w} I(x+dx, y+dy) \cdot G(dx, dy)$$

- Komplexität für 2D Filter: ***O*(w²)**

# 2D Filter

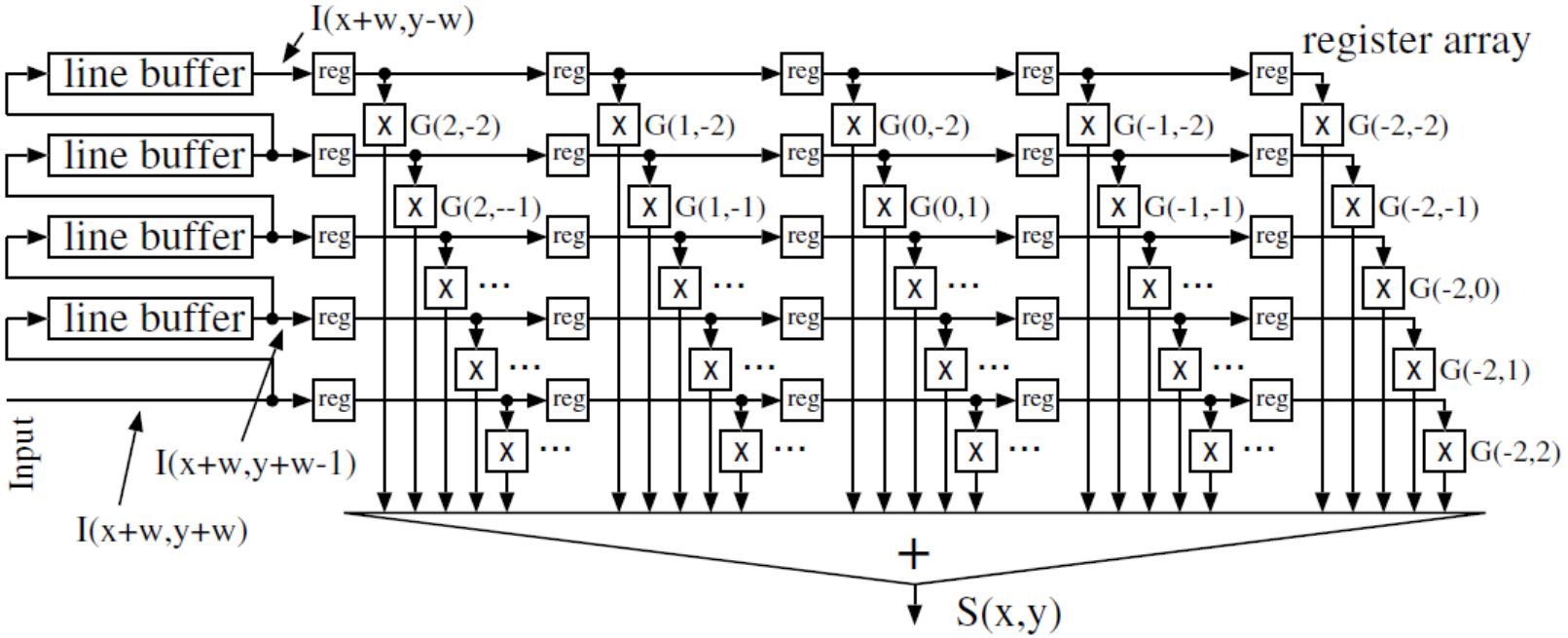


**Abb. 2:** Schaltkreis für einen 2D Filter [1]

# 2D Filter

- Parallelisierung durch gleichzeitige Berechnung von
  - **2w+1** Multiplikationen auf FPGA
  - **4** Pixeln mit **4** Kernen auf CPU
  - **8** Multiplikationen pro Kern auf CPU mit SIMD-Operationen
- GPU kann für maximale Performance **960 S(x, y)** parallel berechnen, da diese voneinander unabhängig sind

# 2D Filter – Vergleich

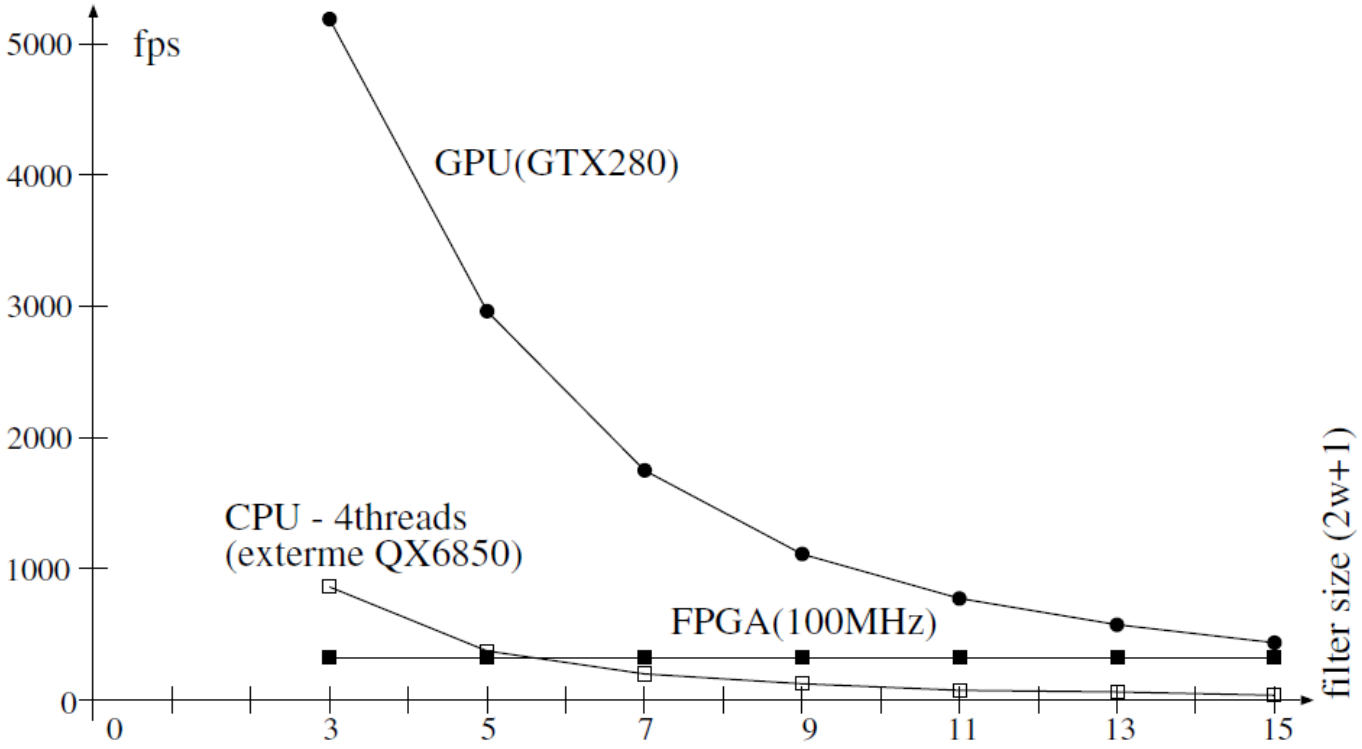


**Abb. 3:** Vergleich für 2D Filter bei einem 640x480 Pixel großen Graustufenbild [1]

# Stereo Vision

- Vergleich von zwei Bildern aus verschiedenen Blickrichtungen
- Berechnen der Entfernungen über Verschiebung
- Vergleich über „sum of absolute difference“ (SAD)

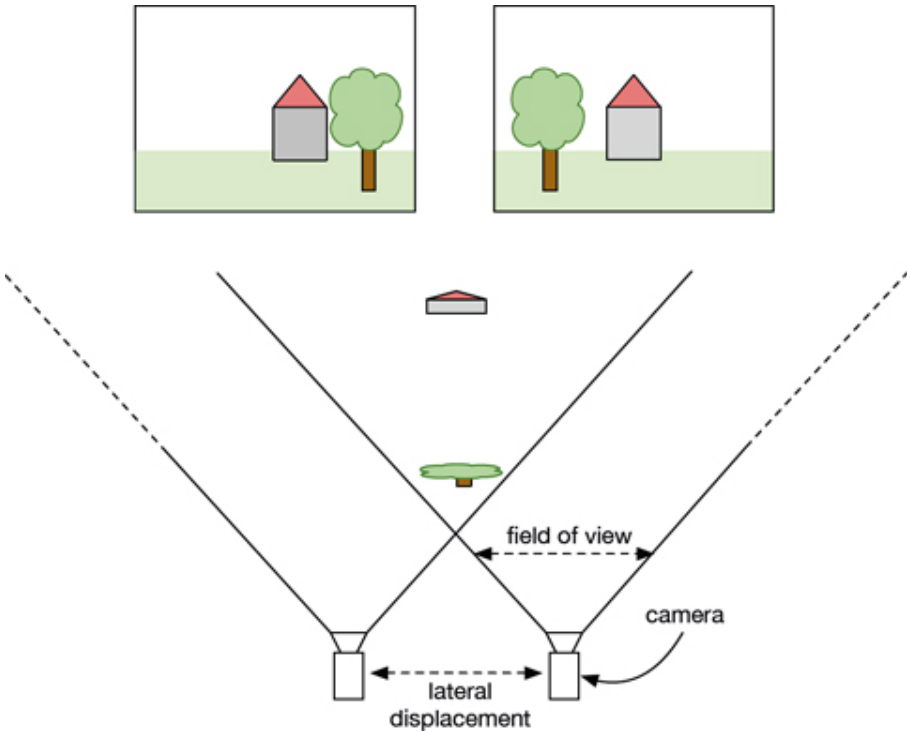


**Abb. 4:** Stereovision [5]

# Stereo Vision

$$SAD_{xy}(x, y, d) = \sum_{dx=-w}^{w} \sum_{dy=-w}^{w} \left| I_r(x+dx, y+dy) - I_l(x+d+dx, y+dy) \right|$$

- Bildgröße: **x·y**
- Verschiebung: **d**
- Größe des Fensters: **(2w+1)·(2w+1)**
- Anzahl der zu Vergleichenden Fenster: **D**
- Komplexität des naiven Ansatzes: ***O*(x·y·w²·D)**

# Stereo Vision

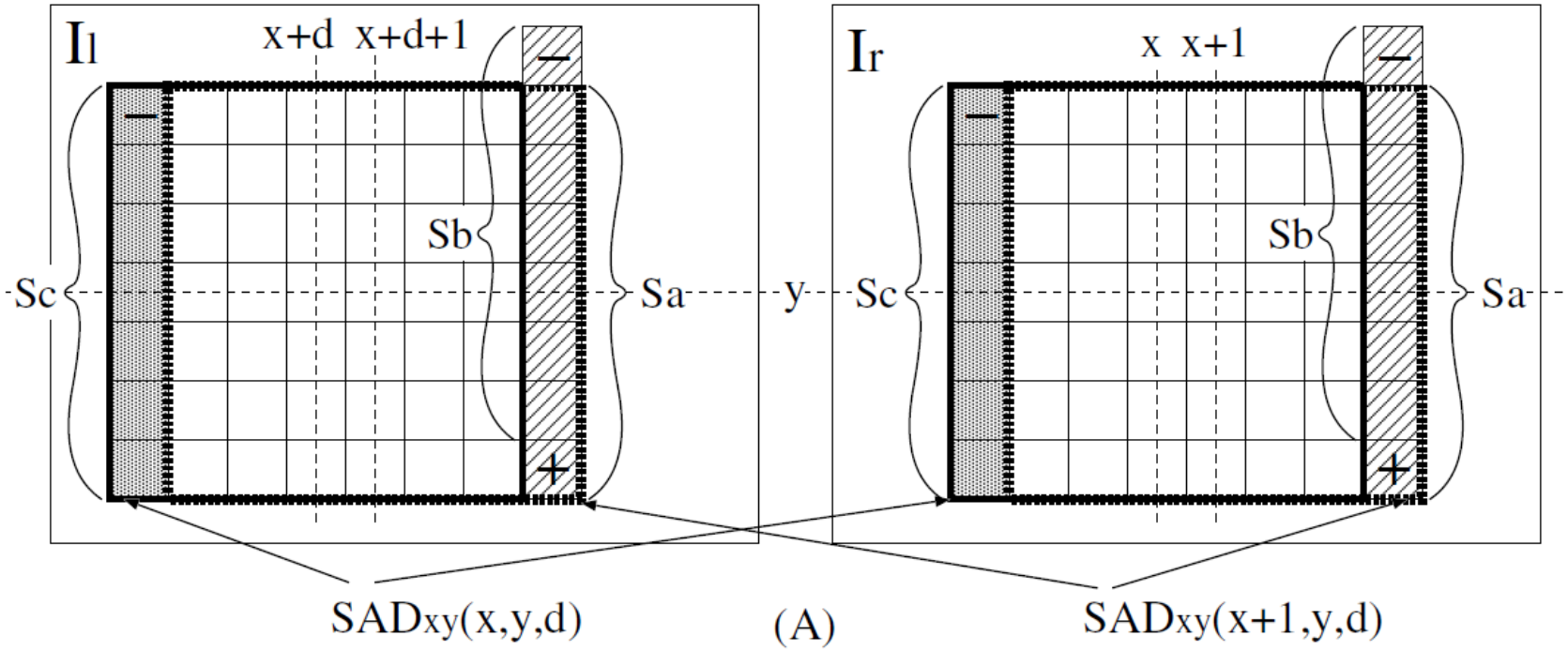


**Abb. 5:** Optimierung für SAD [1]

# Stereo Vision

- Es werden **D** Buffer der Größe **x** benötigt
- Es werden **D** FIFOs der Größe **2w+1** benötigt

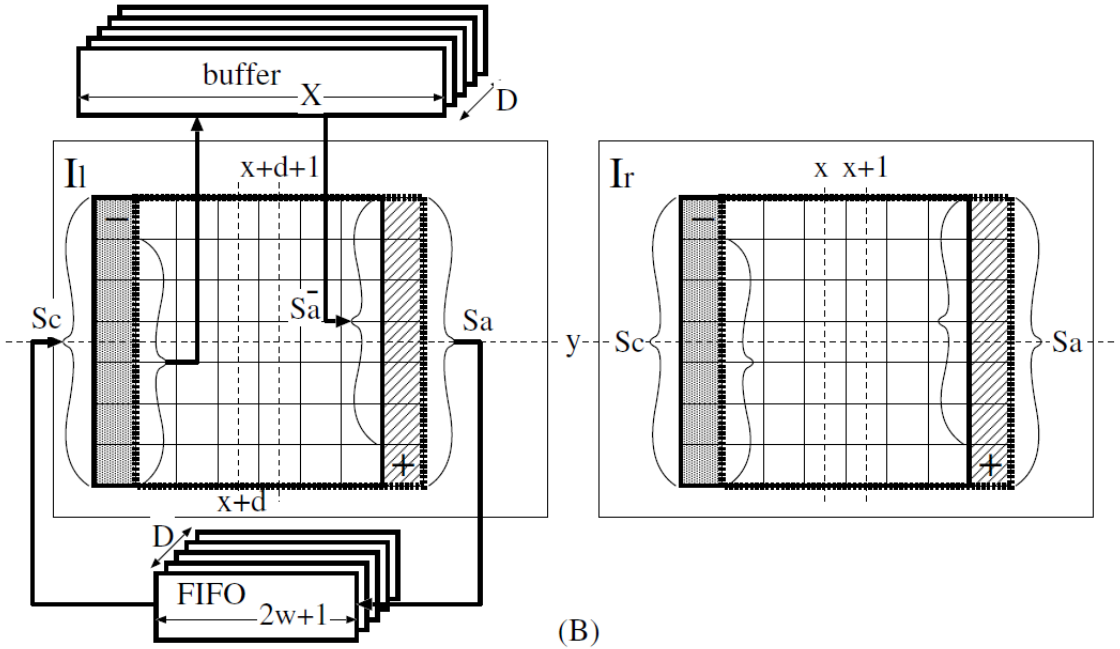


**Abb. 6:** Berechnungsmethode für SAD [1]

# Stereo Vision

- Komplexität nach Reduzierung: ***O*(x·(y+w)·D)**
- Parallelisierung durch gleichzeitige Berechnung von
  - **D** SADs auf FPGA
  - **8** SADs auf CPU mit SIMD-Operationen
  - **N** Pixeln, bei Aufteilung d. Bildes in **N** Teilbilder auf FPGA
  - **4** Teilbildern auf **4** Kernen auf CPU
- Nicht möglich für GPU, da lokaler Speicher zu klein und Zugriffszeiten auf globalen Speicher zu groß für Buffer und FIFOs

# Stereo Vision

- Berechnung jeder SAD in eigenem Thread
- Threads laufen parallel und teilen sich Local Memory eines Streaming Multiprozessors
- Optimierung durch Zusammenlegen von Threads
  - Für 512 < X < 1024 können zwei Threads zusammengelegt werden
  - Min. 32 Threads pro Streaming Multiprozessor

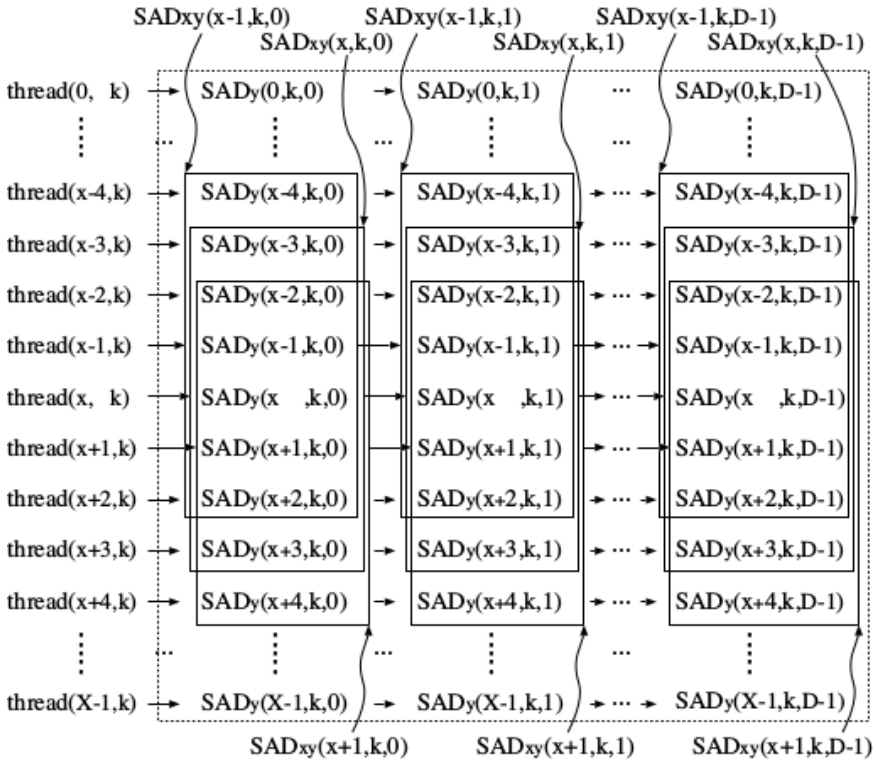


**Abb. 7:** Berechnungsmethode für GPU [1]

# Stereo Vision – Vergleich

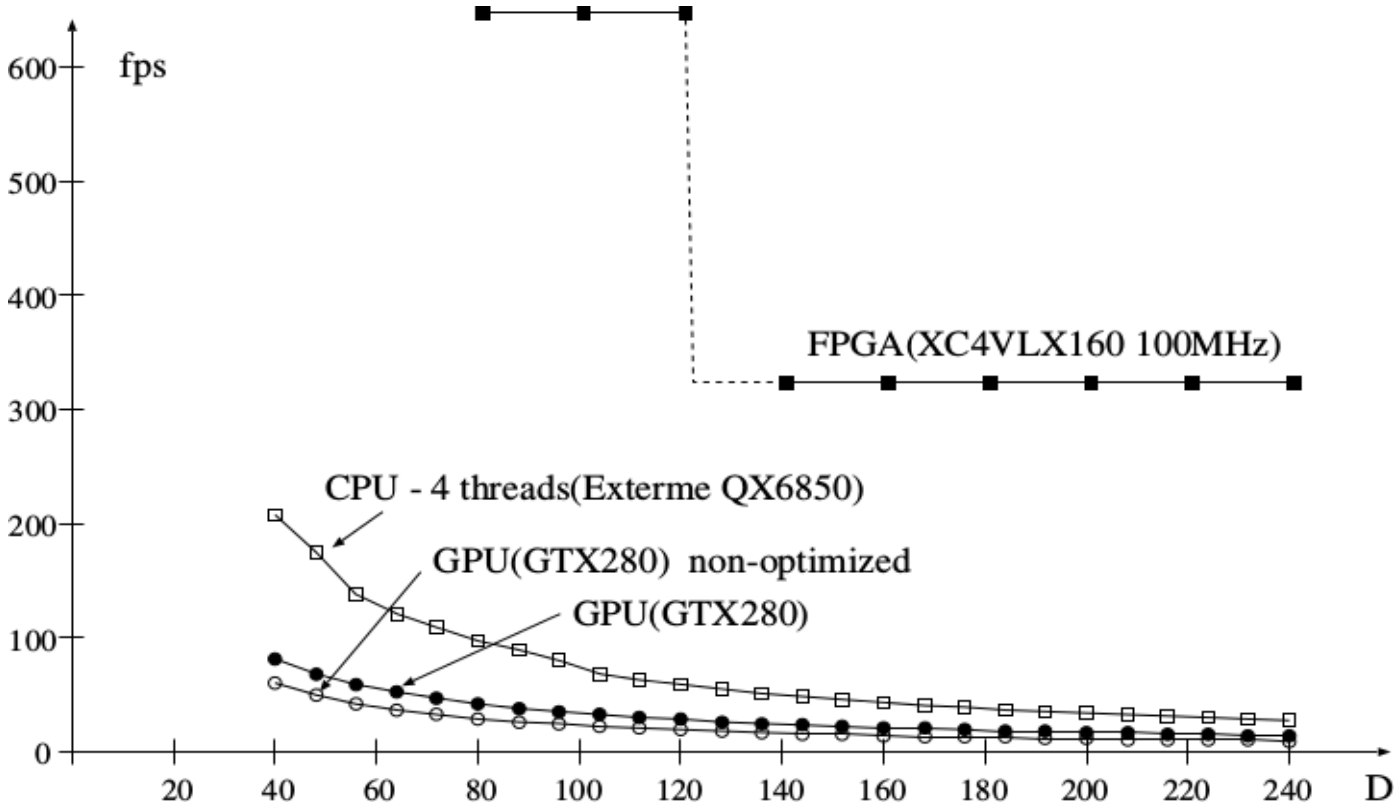


**Abb. 8:** Vergleich für Stereo Vision bei einem 640x480 Pixel großem Graustufenbild mit einer Fenstergröße von 7x7 [1]

# k-Means

- Ziel: Aufteilung der Datenpunkte in k Teilmengen

**Abb. 9:** k-Means auf Bild angewandt mit k=8 [6]

# k-Means

$$E = \sum_{i=1}^{K} \sum_{x \in C_i} \left( x - center_i \right)^2 \qquad center_i = \sum_{x \in C_i} x / \left| C_i \right|$$

- Die Punkte sollen in **K** Mengen $\mathbf{C_i}$ mit **i=1…K** eingeteilt werden, sodass **E** minimal ist
  1. Jeder Punkt wird dem Cluster mit dem kleinsten Abstand zugeteilt
  2. Die Zentren der Cluster werden neu berechnet
  3. Die Operationen werden solange wiederholt bis es zu keiner Verbesserung für **E** kommt

# k-Means

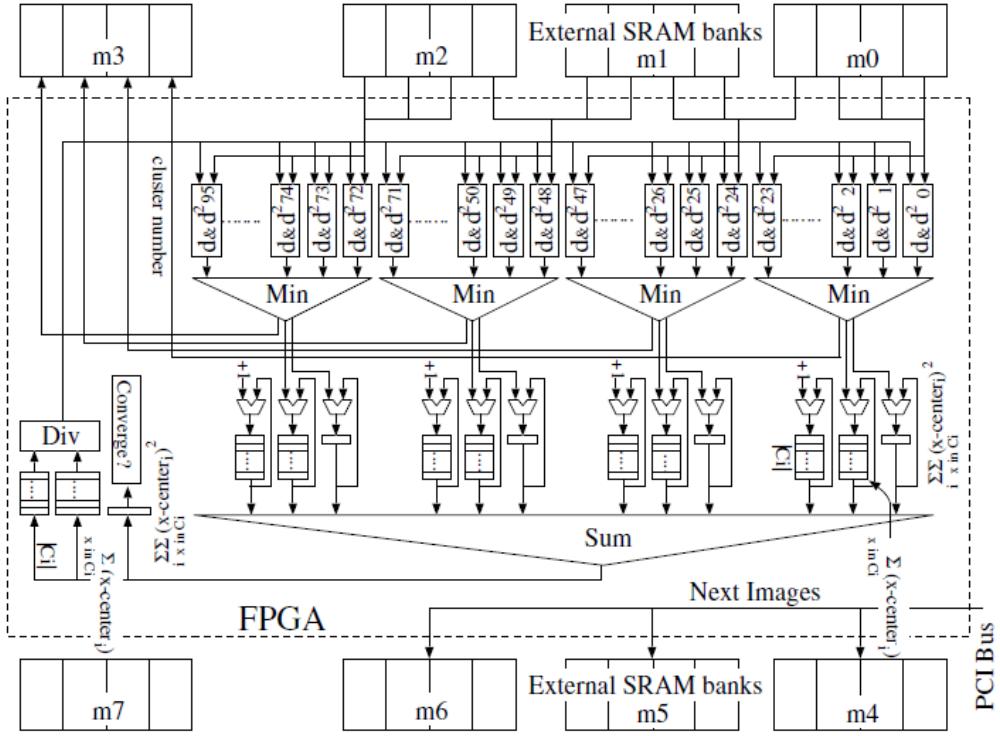


**Abb. 10:** Schaltkreis für k-Means Algorithmus für 24 Bit RGB-Bilder [1]

# k-Means

- Parallelisierung durch gleichzeitige Berechnung von
  - **96** Abständen (24 pro Pixel * 4 Pixel) auf FPGA
  - **8** Abständen pro Kern auf CPU mit SIMD-Operationen
  - **4** Pixeln auf **4** Kernen auf CPU
  - **N** Threads für **N** Pixel in einer Zeile/Spalte berechnen sequentiell **K** Abstände auf GPU (960 Threads für maximale Performance)
    - Summenbildung für **E** in globalen Speicher
    - 32 Threads pro Streaming Multiprozessor müssen parallel auf lokalen Speicher schreiben → Zugriffskonflikte

# k-Means - Vergleich

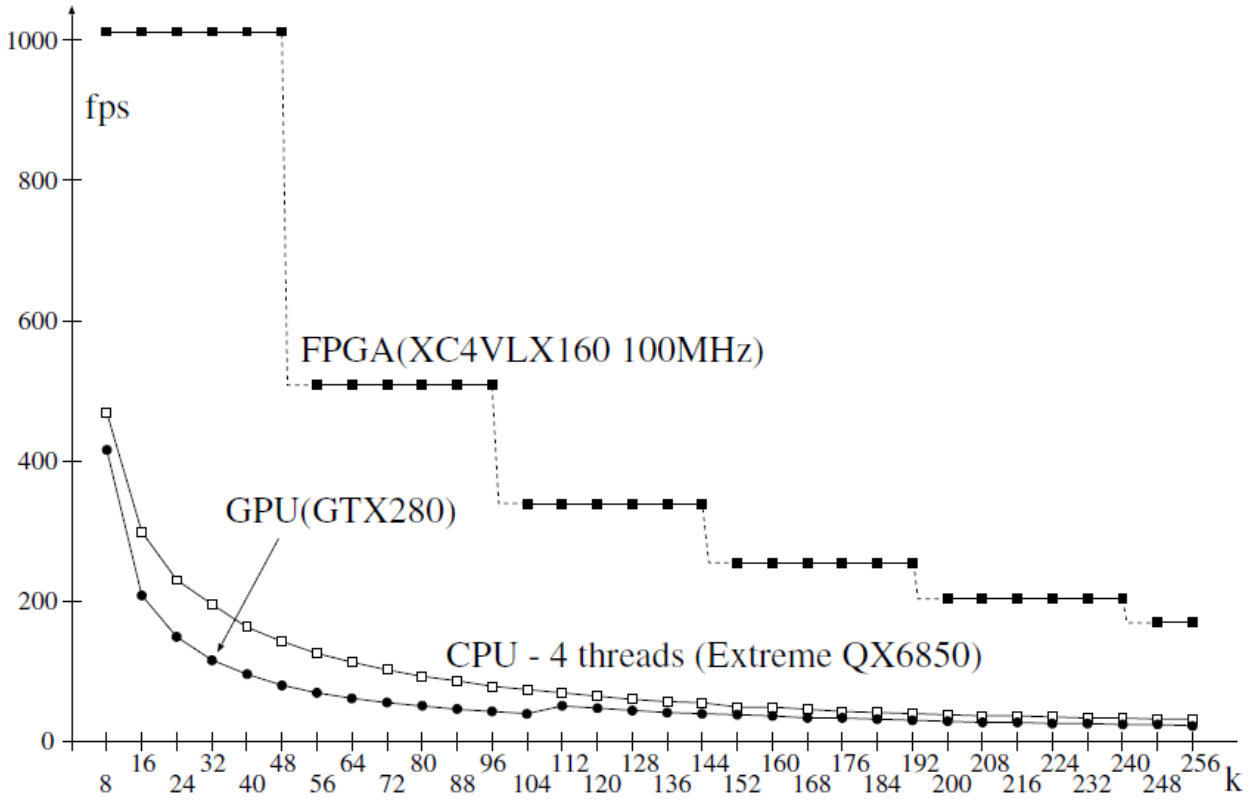


**Abb. 11:** Vergleich für k-Means bei einem 768x512 Pixel großem Farbbild [1]

# Diskussion der Ergebnisse

- Verschiedene Algorithmen laufen unterschiedlich auf verschiedener Hardware
- Performance von GPU sehr gut, bei naiven Algorithmen, bei denen alle Berechnungen unabhängig voneinander sind
- CPU ist schneller bei Algorithmen mit voneinander abhängigen Berechnungen als GPU
- Performance von FPGA nur limitiert durch Größe und Speicheranbindung
- Nur drei Probleme betrachtet und Algorithmen für GPU und CPU wurden nicht vollständig optimiert
- Stromverbrauch und Kosten wurden im Vergleich nicht mit einbezogen

# Quellen


- [1] S. Asano, T. Maruyama, Y. Yamaguchi: „Performance Comparison of FPGA, GPU and CPU in Image Processing“, FPL, 2009, S. 126-131
- [2] T. Saegusa, T. Maruyama, Y. Yamaguchi: „How fast is an FPGA in image processing?“, FPL, 2008, S. 77-82
- [3] T. S. Huang, “Stability of two-dimensional recursive filters,” IEEE Transactions on Audio and Electroacoustics, 1972, S. 158–163
- [4] J. S. Lim, Two-Dimensional Signal and Image Processing, Prentice-Hall International, 1990
- [5] Single-Camera Stereo Vision for Mobile Devices, http://mint.fh-hagenberg.at/?p=1149&doing_wp_cron=1508173850.0824210643768310546875 , 15.10.2017

# Quellen


- [6] k-Means Algorithmus, https://carlosvision.files.wordpress.com/2011/09/kmeans_eucl_rgb1.jpg , 18.10.2017