Forschungswerkstatt 1

Laufzeitoptimierung heterogener Multiprozessorsysteme im automotive Bereich

Moritz Höwer
22.02.2021

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

HAW HAMBURG

Agenda

- Motivation
- Grundlagen Rechnerarchitekturen
- Herausforderungen
- Funktionale Sicherheit (ISO 26262)
- Fazit & Ausblick

Motivation

Motivation

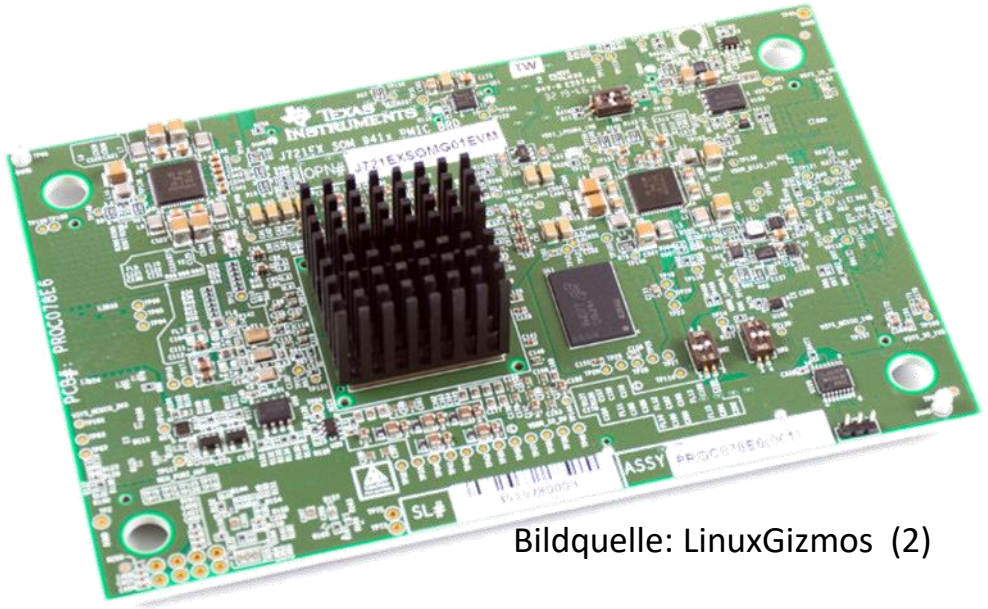
- In Bachelorarbeit bereits Fahrspurerkennung entwickelt
  - Optimiert auf NVIDIA Tegra GPU
- Neue Zielplattform TDA4VM
  - Keine GPU, dafür DSPs
  - Nochmals weniger Leistungsaufnahme
  - Weiterhin Echtzeitanforderungen
- Vorbereitung auf Serienentwicklung
  - Beachtung von Funktionalen Sicherheitsanforderungen
- In der Forschungswerkstatt 1 theoretische Grundlage schaffen

Bildquelle: Bachelorarbeit (1)

Motivation

- Der TDA4VM von Texas Instruments
  - Nach ISO 26262 zertifiziert
  - Verschiedene RISC-Prozessoren
  - Mehrere DSPs
  - Mehrere Spezialbeschleuniger
  - ~~GPU~~ *(nicht zertifiziert)*
  - Leistungsaufnahme ca. 20 W

- Heterogenes Multiprozessorsystem



Bildquelle: LinuxGizmos (2)

Grundlagen Rechnerarchitekturen

Forschungswerkstatt 1

Moritz Höwer

HAW HAMBURG

Nebenläufigkeit auf drei Ebenen

- Function / Task Level Parallelism
  - Mehrere Funktionen gleichzeitig ausführen
  - Threads
- Data Level Parallelism
  - Dieselbe Instruktion auf verschiedenen Daten gleichzeitig ausführen
  - Vektorisierung / Single Instruction Multiple Data (SIMD)
- Instruction Level Parallelism
  - Mehrere Instruktionen gleichzeitig ausführen
  - Superskalar / Very Long Instruction Word (VLIW)

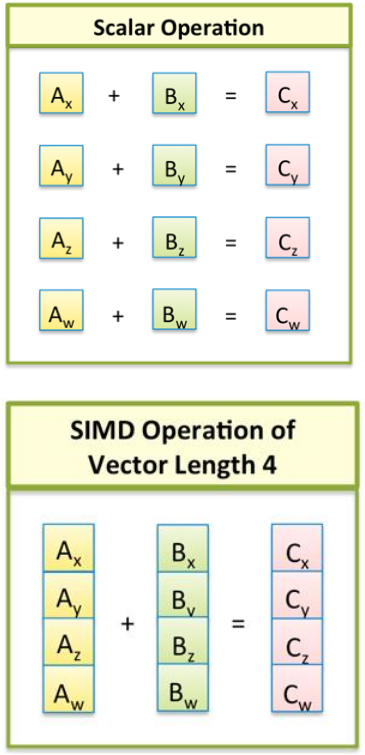
Single Instruction Multiple Data (SIMD)

- Prozessor hat SIMD-Einheiten
  - z.B. Arithmetic Logic Unit (ALU)

- Typische Programmstruktur: Schleife, um Vektor zu addieren
  - Schleifendurchläufe machen immer dasselbe
  - Schleifendurchläufe sind unabhängig voneinander (Ausnahme: Schleifenzähler, also Adresse der Daten)

  → Schleife ist vektorisierbar

  → Mehrere Additionen können gleichzeitig ausgeführt werden



Intel® Architecture currently has SIMD operations of vector length 4, 8, 16

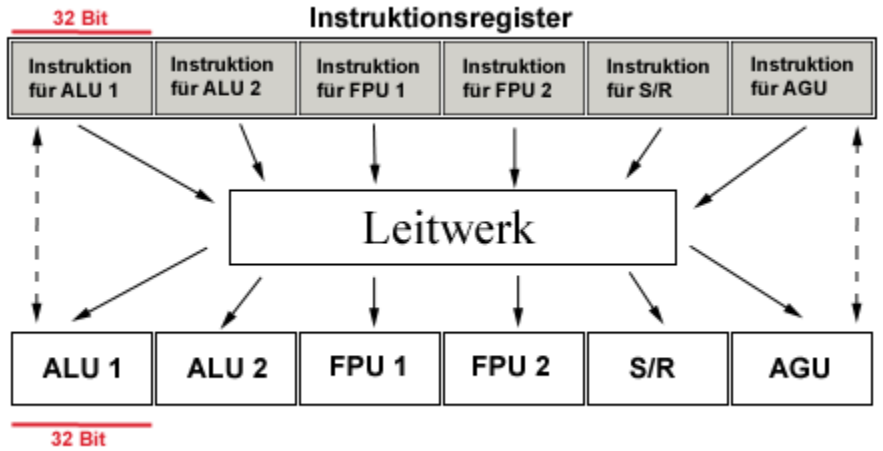
Bildquelle: Medium (3)

Very Long Instruction Word (VLIW)

- Prozessor hat mehre Funktionseinheiten

- Eine Instruktion pro Funktionseinheit pro Taktzyklus

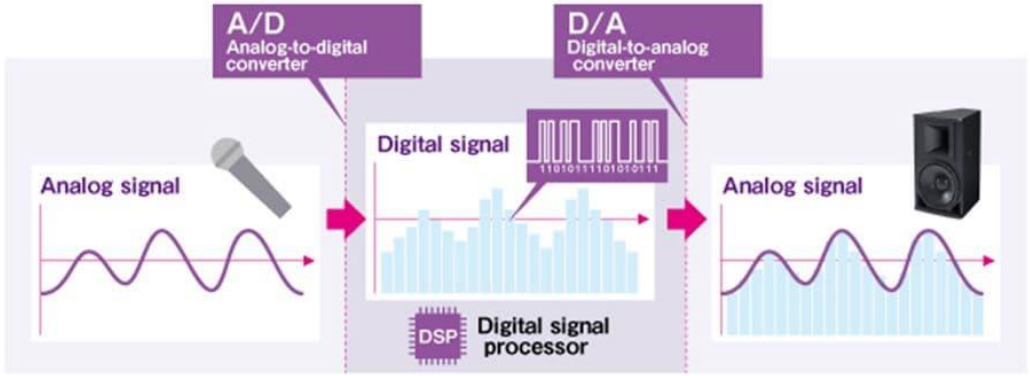
- Einfache Hardware
  - Compiler garantiert Unabhängigkeit
    → Keine Abhängigkeitsprüfung in Hardware nötig
    → `NOP` wenn keine passende Instruktion vorhanden
  - Feste Position im Instruktionswort
    → kein Routing von Instruktionen zu Funktionseinheiten nötig



Bildquelle: Planet3Dnow (4)

Digitaler Signalprozessor (DSP)

- Spezialisierter Prozessor für digitale Signalverarbeitung



Bildquelle: Yamaha (5)

- Typische Eigenschaften
  - Wiederkehrende Berechnungen (Filter)
  - Kontinuierliches Eingangssignal → Echtzeitanforderung
- Aufbau
  - Kein Cache für die Daten, sondern On-Chip-Memory für Zwischenergebnisse
  - Simple Hardware (keine Sprungvorhersage, etc.)
- Außerdem: günstig und energiesparend

Herausforderungen

Was steht zur Verfügung?

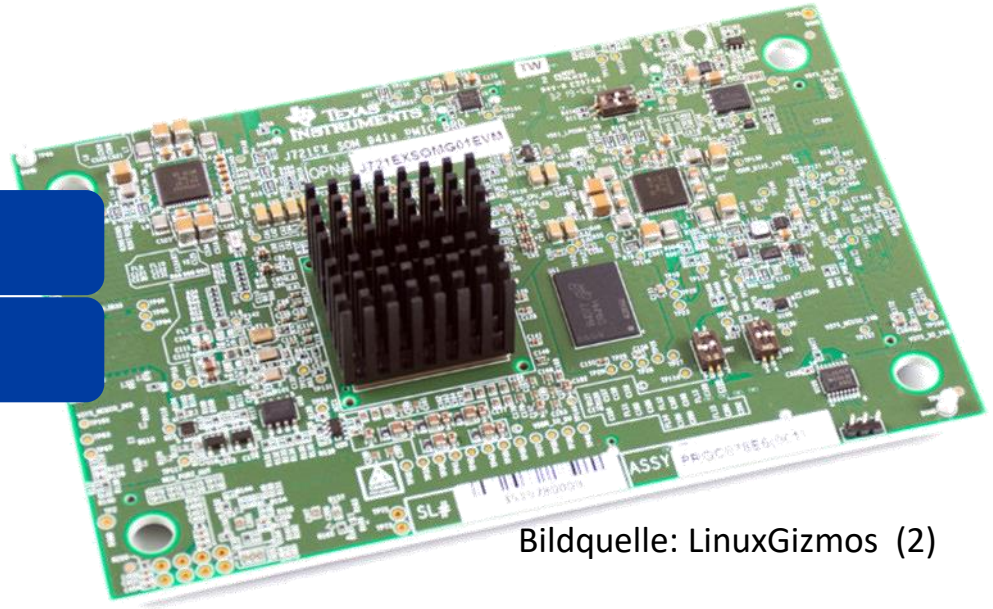
- Der TDA4VM von Texas Instruments
  - Nach ISO 26262 zertifiziert
  - Verschiedene RISC-Prozessoren
  - Mehrere DSPs
  - Mehrere Spezialbeschleuniger
  - GPU *(nicht zertifiziert)*
- Heterogenes Multiprozessorsystem

SIMD

VLIW

Vision Processing Accelerator (VPAC)

Matrix Multiply Accelerator (MMA)



Bildquelle: LinuxGizmos (2)

Grundlegende Problemstellung

Wie kann der Algorithmus auf die verschiedenen Rechenkerne verteilt werden?

Welche Möglichkeiten zur Optimierung bieten die verschiedenen Rechenkerne?

Architecture Mapping & Offloading

- Ausgangspunkt: Alles läuft auf dem Hauptkern

- Rechenintensive Abschnitte ermitteln
  - Prüfen, ob Spezialbeschleuniger vorhanden
    - Beispiel: Bild glätten und verkleinern (Gauß Pyramide) → VPAC
    - Beispiel: Matrixmultiplikation → MMA
  - Prüfen, ob DSP geeignet ist
    - Beispiel: Fourier Transformation
  - Prüfen, ob Offloading sinnvoll
    - Kommunikationsoverhead beachten!

Architecture Mapping & Offloading

- Offloading dynamisch
  - Zur Laufzeit nach Ressourcenverfügbarkeit entscheiden, was wo läuft
  - Kompletten Programmcode („Kernel“) auf den Beschleuniger kopieren und ausführen

  (+) Flexibel und einfach erweiterbar

  (-) Laufzeit nicht vorhersagbar und komplex

- Offloading statisch
  - Jeweils ein eigenes Programm pro Beschleuniger inklusive aller möglichen „Kernel“
  - Zur Laufzeit nur Daten kopieren und Kernel anstarten

  (+) Laufzeiten vorhersagbar und vergleichsweise simpel

  (-) Nicht ohne weiteres einfach so erweiterbar

Laufzeitoptimierung

- Laufzeitoptimierung geht immer bis ans Limit

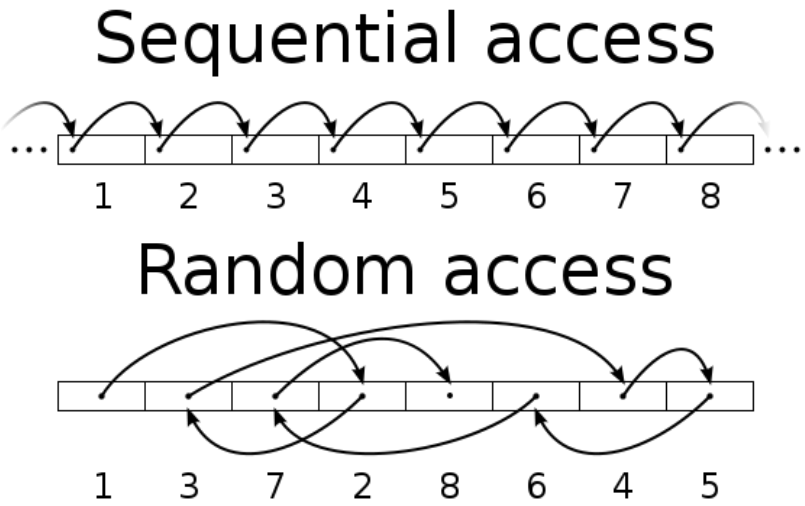
- Begrenzender Faktor typischerweise Speicherzugriff
  - ungünstiges Speicherlayout
  - ineffiziente Nutzung von Caches

- Seltener: zu hohe Auslastung der Recheneinheiten

- Ebenfalls beachten: Vektorisierungsverhältnis
  - Verhältnis genutzter Operanden zur Vektorlänge
  - Ähnliche Idee auch bei VLIW im Bezug auf Instruktionen

Speicherzugriffsmuster

- Schlecht: Zufälliger Zugriff
  - Nicht vorhersagbar
- Besser: Sequentieller Zugriff
  - Vorhersagbar
  - Hardware kann schon das nächste Datum laden („Prefetching“)
- Ideal: Kontinuierlicher Zugriff (Schrittweite 1)

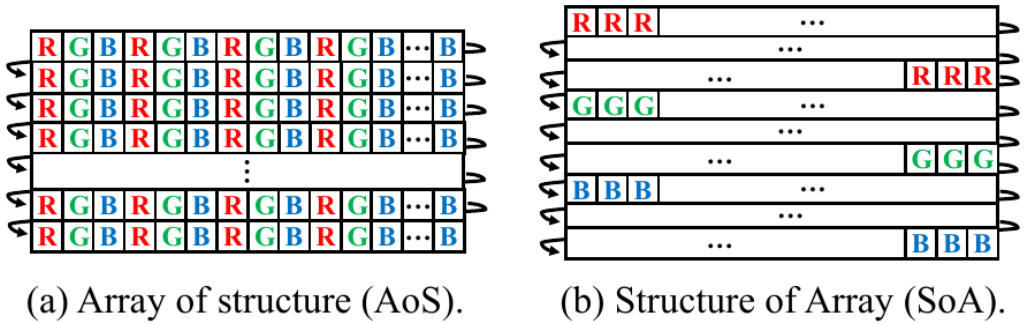


Bildquelle: Wikimedia (6)

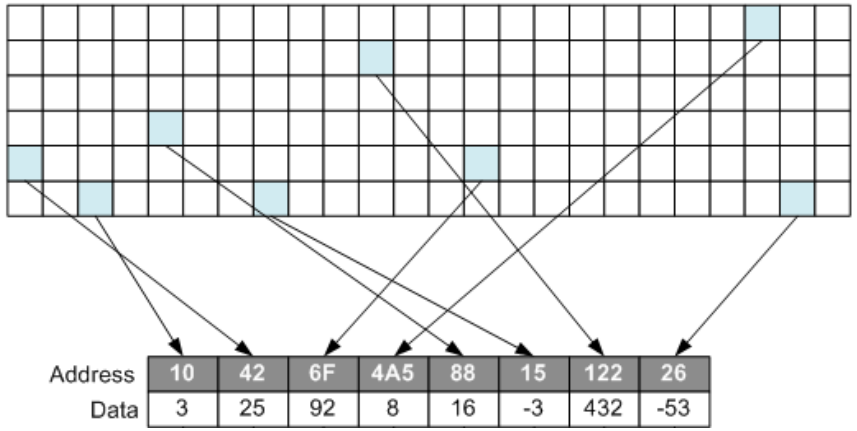
Speicherlayout & Scatter / Gather

- Scatter / Gather
  - Sammeln (Gather) von im Speicher verteilten Daten in einem kontinuierlichen Puffer
  - Verteilen (Scatter) von Daten aus dem kontinuierlichen Puffer in den Speicher

- Besser: Wahl der Datenstruktur passend zum Speicherzugriffsmuster
  - ggf. vorher transformieren



Bildquelle: Maeda Et al. (7)



Bildquelle: EEjournal (8)

Funktionale Sicherheit (ISO 26262)

Forschungswerkstatt 1
Moritz Höwer

HAW HAMBURG

Der ISO 26262

- „Functional Safety and Road Vehicle Standard”
  - Spezialisierung der Funktionalen Sicherheitsnorm IEC 61508 für den Automobilbereich

- Regelt Produktsicherheitslebenszyklus von Entwicklung bis Außerbetriebnahme
  - Betrachtet sporadische Hardwarefehler im Betrieb und Fehler bei der Entwicklung (dürfen nicht zu Gefahr für Mensch oder Maschine führen)
  - Betrachtet nicht böswillige Manipulationen oder Erfüllung von Kundenanforderungen

- Stand der Wissenschaft und Technik
  → muss in Serienprodukten eingehalten werden

Der ISO 26262

- Basierend auf Gefährdungspotential und Risiko wird „Automotive Safety Integrity Level“ (ASIL) vergeben
- Bei geringem Risiko reicht Qualitäts-management des Unternehmens
- Anforderungen steigen mit ASIL
- Über zwei unabhängige, redundante Systeme kann ASIL verringert werden („ASIL-decomposition“)

Decreased Controllability

Increased Severity

ASIL (Automotive Safety Integrity Level		Controllability (C)			
Severity (S)	Expopsure (E)	C0 Controllable in General	C1 Simply Controllable	C2 Normally Controllable	C3 Difficult to Control or Uncontrollable
S0 No Injuries	–	QM			
S1 Light and Moderate Injuries	E0 – Unusual	QM			
	E1 – Very Low Probability	QM			
	E2 – Low Probability	QM			
	E3 – Medium Probability	QM			ASIL A
	E0 – High Probability	QM		ASIL A	ASIL B
S2 Severe Injuries, Possibly Life Threatening	E0 – Unusual	QM			
	E1 – Very Low Probability	QM			
	E2 – Low Probability	QM			ASIL A
	E3 – Medium Probability	QM		ASIL A	ASIL B
	E0 – High Probability	QM	ASIL A	ASIL B	ASIL C
S3 Life Threatening or Fatal Injuries	E0 – Unusual	QM			
	E1 – Very Low Probability	QM			ASIL A
	E2 – Low Probability	QM		ASIL A	ASIL B
	E3 – Medium Probability	QM	ASIL A	ASIL B	ASIL C
	E0 – High Probability	QM	ASIL B	ASIL C	ASIL D

Bildquelle: Analog Dialogue (9)

ASIL-Einstufung der Fahrspurerkennung

- Teil von „Advanced Driver Assistance System“ (ADAS)
  - Autopilot, Staupilot oder ähnlich

- Bei Ausfall des ADAS…
  - Hohe Wahrscheinlichkeit tödlicher Verletzungen
  - Fahrzeug schwer zu kontrollieren (Fahrer ist abgelenkt)

  → ASIL-D

- Fahrspurerkennung als Teilsystem (kann redundant und unabhängig gebaut werden)

  → ASIL-B (D)

ASIL-B – Anforderungen an die Software

- Vorgaben an Architektur und Implementierung, u.a.
  - Beschränkung auf ein als sicher geltendes Subset der verfügbaren Sprachfeatures (z.B. MISRA-C++)
  - Keine Nutzung von dynamischem Speicher ohne Überprüfung zur Laufzeit
  - Keine impliziten Typumwandlungen
  - Bei Tests wird Statement und Branch coverage gefordert
  - Es müssen Hardware in the Loop (HiL)-Tests durchgeführt werden

- Einhaltung der Vorgaben gilt für alle eingesetzten Softwarekomponenten
  - Compiler und Tools
  - Drittanbieter Bibliotheken

ASIL-B – Anforderungen an die Software

- Zertifizierte Compiler und Tools kann man kaufen ✔

- Komplexe OpenSource Bibliothek meist nicht zertifizierbar ✘
  (nicht mit vertretbarem Aufwand)
  - ➔ Für Serienentwicklung nicht ohne weiteres nutzbar

- Mögliches Vorgehen:
  - Drittanbieter Bibliotheken für Prototypen nutzen
  - Tatsächlich benötigte Funktionalität ermitteln
  - Eigene Bibliothek entwickeln und zertifizieren

Fazit & Ausblick

Fazit

- Zertifizierung nach ISO 26262 größte Hürde und Einschränkung

- Anforderung in der Literatur:
  - Portabilität
  - Generischer Ansatz

- Anforderung in der Serienentwicklung:
  - Spezifisch auf ein System

➔ Viele Interessante Ansätze, aber nur für Prototypen geeignet

Ausblick

- Grundprojekt:
  - Erste Schritte mit TDA4VM
- Hauptseminar + Hauptprojekt:
  - Aufbauen auf Forschungswerkstatt 1 + Grundprojekt
  - Erste Implementierung der Fahrspurerkennung (Fokus Echtzeitfähigkeit)
- Masterarbeit:
  - Verfeinerung der Implementierung (Echtzeitfähigkeit und Zertifizierbarkeit)

Fragen?

HAW HAMBURG

Bildquellen

(1) Höwer, Moritz: Effiziente GPU-basierte Klassifizierung von Fahrspuren auf eingebetteten Echtzeitsystemen, Hamburg: Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, Juni 2019

(2) http://linuxgizmos.com/linux-based-dev-kits-unlock-tis-new-jacinto-7-automotive-socs/

(3) https://medium.com/@pixelstab/the-simd-experience-data-parallelism-on-my-game-engine-13711054ed6e

(4) https://www.planet3dnow.de/cms/20378-doping-fuer-cpus-moeglichkeiten-der-leistungssteigerung/8/

(5) https://de.yamaha.com/de/products/contents/proaudio/docs/better_sound/part1_06.html

(6) https://en.wikipedia.org/wiki/File:Random_vs_sequential_access.svg

(7) Maeda, Yoshihiro ; Fukushima, Norishige ; Matsuo, Hiroshi: Taxonomy of Vectorization Patterns of Programming for FIR Image Filters Using Kernel Subsampling and New One. In: Applied Sciences 8 (2018), Nr. 8. – URL https://www.mdpi.com/2076-3417/8/8/1235. – ISSN 2076-3417

(8) https://www.eejournal.com/article/20170209-scatter-gather/

(9) https://www.analog.com/en/analog-dialogue/articles/dual-amr-motor-position-sensor-for-safety-critical-applications.html