

Übung 1

Dipl.-Inform. Leonard Masing
Dr.-Ing. Oliver Sander

Institutsleitung

Prof. Dr.-Ing. Dr. h. c. J. Becker

Prof. Dr.-Ing. E. Sax

Prof. Dr. rer. nat. W. Stork

Institut für Technik der Informationsverarbeitung (ITIV)



Hardware/Software Co-Design

Agenda

- Vorstellung der Übung
- Wiederholung
- Vorlesungsnahe Forschungsthemen
- Gruppenarbeit
- Vorstellung der Lösung

Kontakt – Übung

■ Dipl.-Inform. Leonard Masing

- Institut für Technik der Informationsverarbeitung
- Email: Leonard.Masing@kit.edu
- Telefon: 0721-608-46500
- Engesser Str. 5, Raum 226.1



■ M.Sc. Fabian Marc Lesniak

- Institut für Technik der Informationsverarbeitung
- Email: Fabian.Lesniak@kit.edu
- Telefon: 0721-608-42504
- Engesser Str. 5, Raum 226.1



Organisation

- Ilias-Plattform als Vorlesungshomepage
 - Magazin » Organisationseinheiten » Fakultät für Elektrotechnik und Informationstechnik » WS 18/19 » [2311620] Hardware/Software Co-Design (WS 18/19)
 - Für aktuelle Informationen bitte der Vorlesung beitreten
 - Kurspasswort: hsc1819
- Alle Vorlesungsfolien zum Download verfügbar
 - Kein zusätzliches Skript
- Übungsblätter, Lösungen & Zusatzfolien ebenfalls verfügbar
 - Ausgewählte Themen werden anhand von Beispielen wiederholt
 - Übungsblätter werden in der Übung in Kleingruppen bearbeitet
 - interaktive Veranstaltung, erhöhter Lerneffekt

Ablauf der Übung

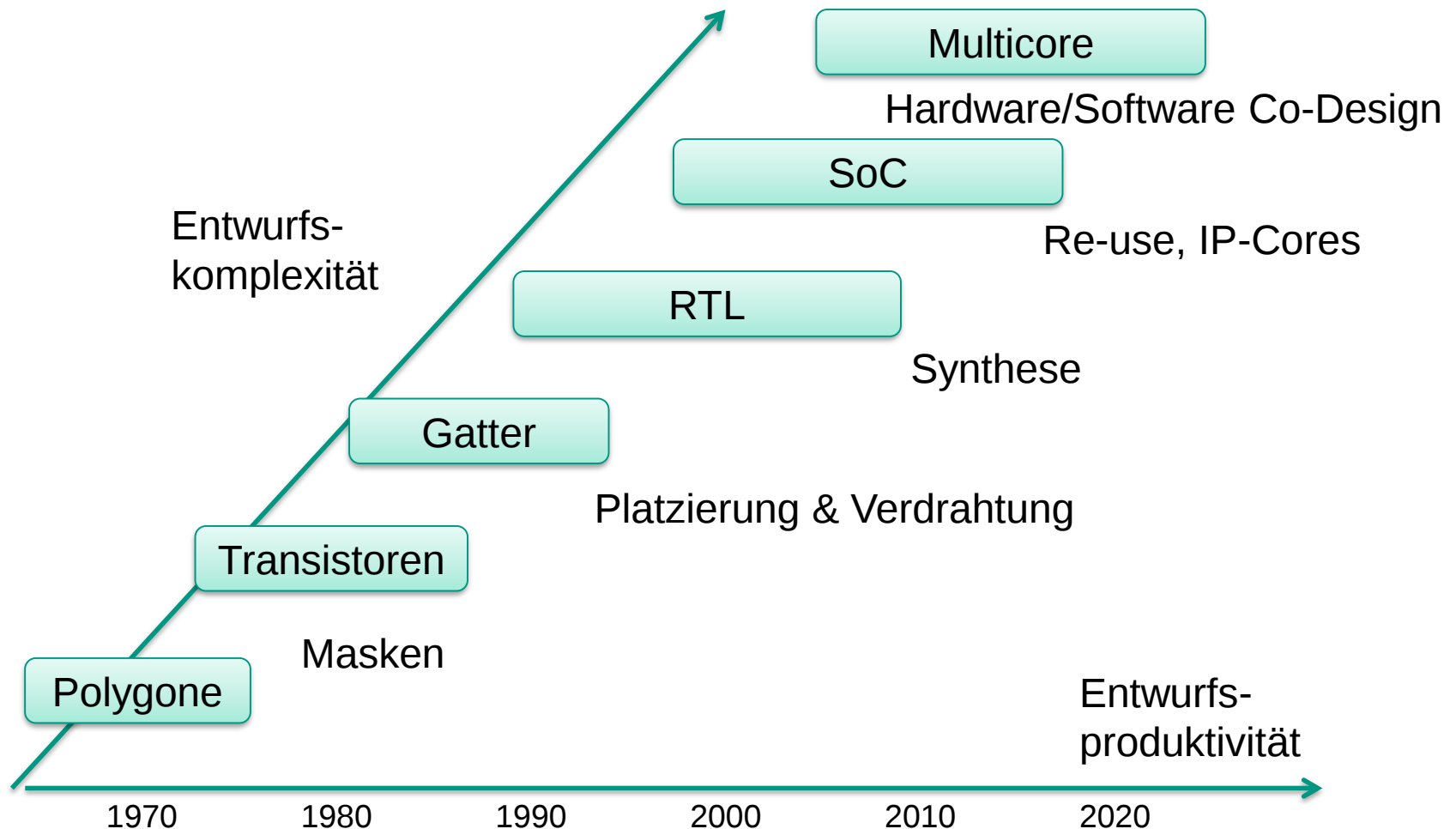
- Wiederholung von ausgewähltem Vorlesungsstoff
 - Vertiefung prüfungsrelevanter Themen

- Bearbeitung der Übungsaufgaben in Kleingruppen
 - 2-3 Personen pro Gruppe
 - Interaktive Diskussion in der Gruppe
 - Vorlesungsfolien hilfreiches Nachschlagewerk
 - Fragen an Übungsleiter

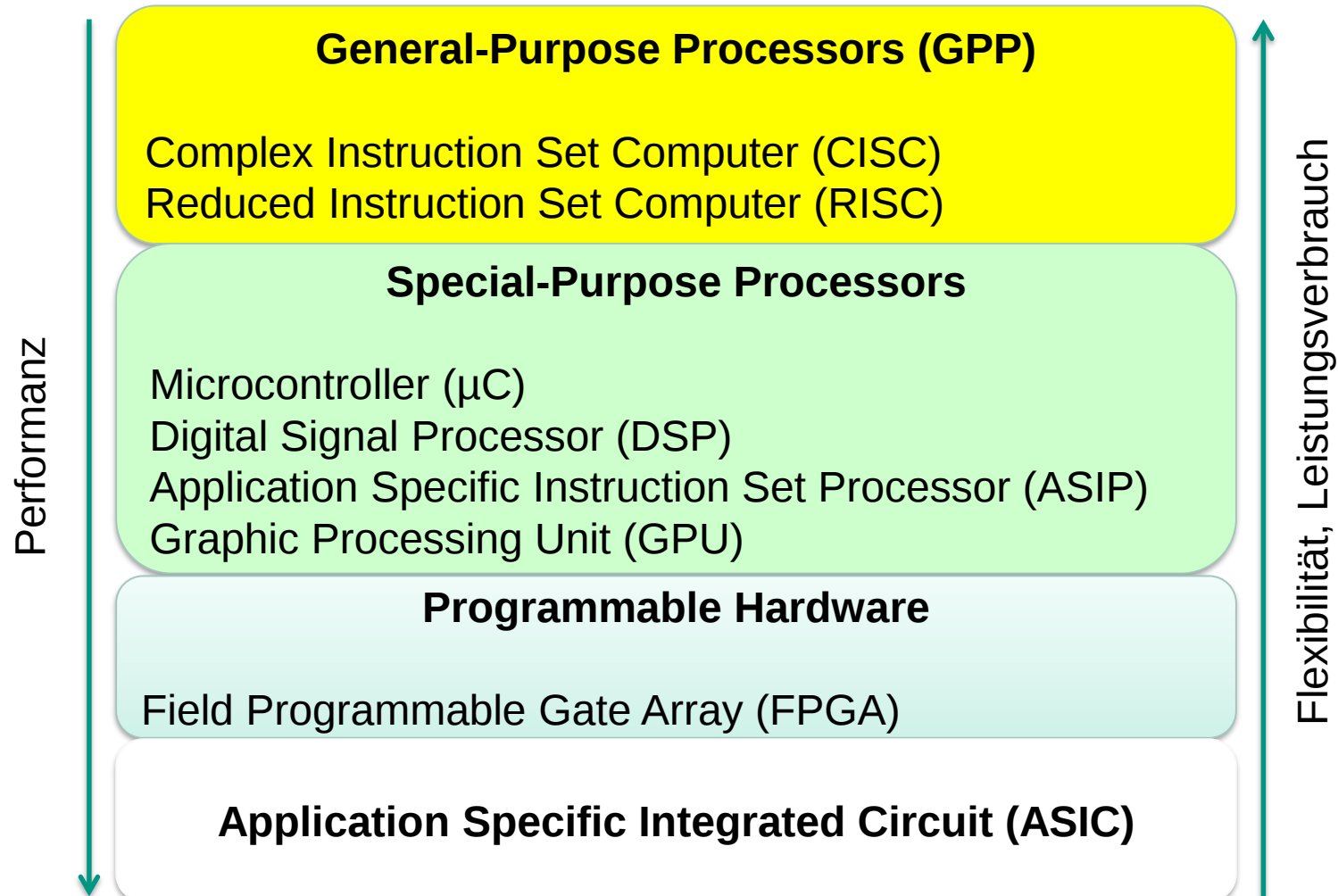
- Vorstellung ausgewählter Ergebnisse
 - Lösungsblätter und Folien werden auf Ilias Lernplattform bereitgestellt

Produktivitätsgrenze (I)

- Komplexe eingebettete Systeme erfordern neue Entwurfsmethoden

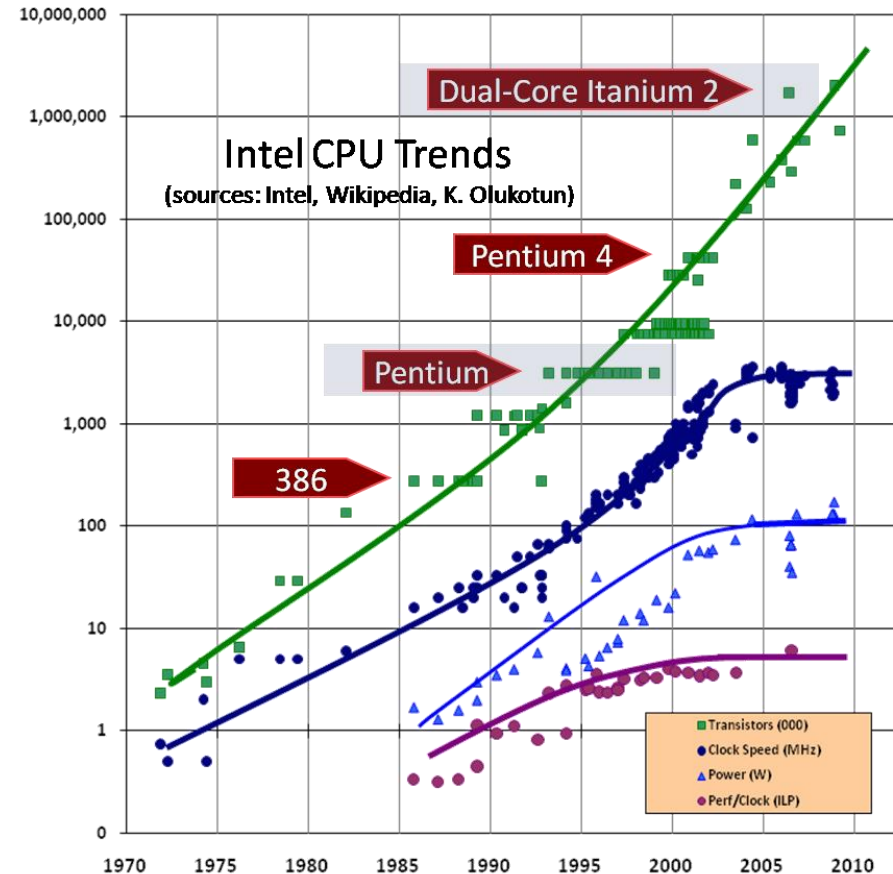


Vergleich der Zielarchitekturen



Free Lunch is over

- Power, Performance & Area (PPA) Trade-Offs im Chip Design
- Frequenz skaliert nicht mehr
 - Keine 10 GHz Systeme
- Fläche skaliert bald nicht mehr
 - Dark silicon
 - Deep sub-micron Effekte
- Verlustleistung
 - Statische und dynamische Verlustleistung vergleichbar
- Ausweg:
 - Nebenläufigkeit der Software
 - Multi-/Many-Core
 - Spezialisierte Architekturen



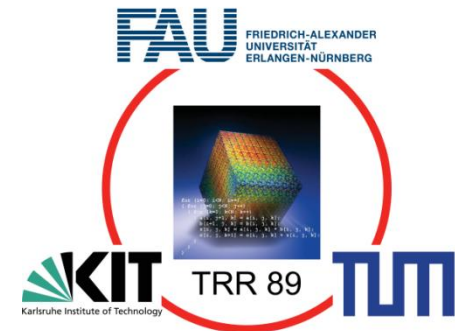
<http://www.gotw.ca/publications/concurrency-ddj.htm>

Invasive Computing (*Invas/C*) – Motivation

- Number of processing cores will increase in future architectures
- How will such many core architectures ...
 - ... be managed ?
 - ... programmed ?
 - ... look like ?
- Funded by the German Research Foundation (DFG)
 - Transregional Collaborative Research Center "Invasive Computing" (SFB/TR 89)
- Life span of the project:
 - Phase I: 01.07.2010 – 30.06.2014
 - Phase II: 01.07.2014 – 30.06.2018
 - Phase III: 01.07.2018 – 30.06.2022

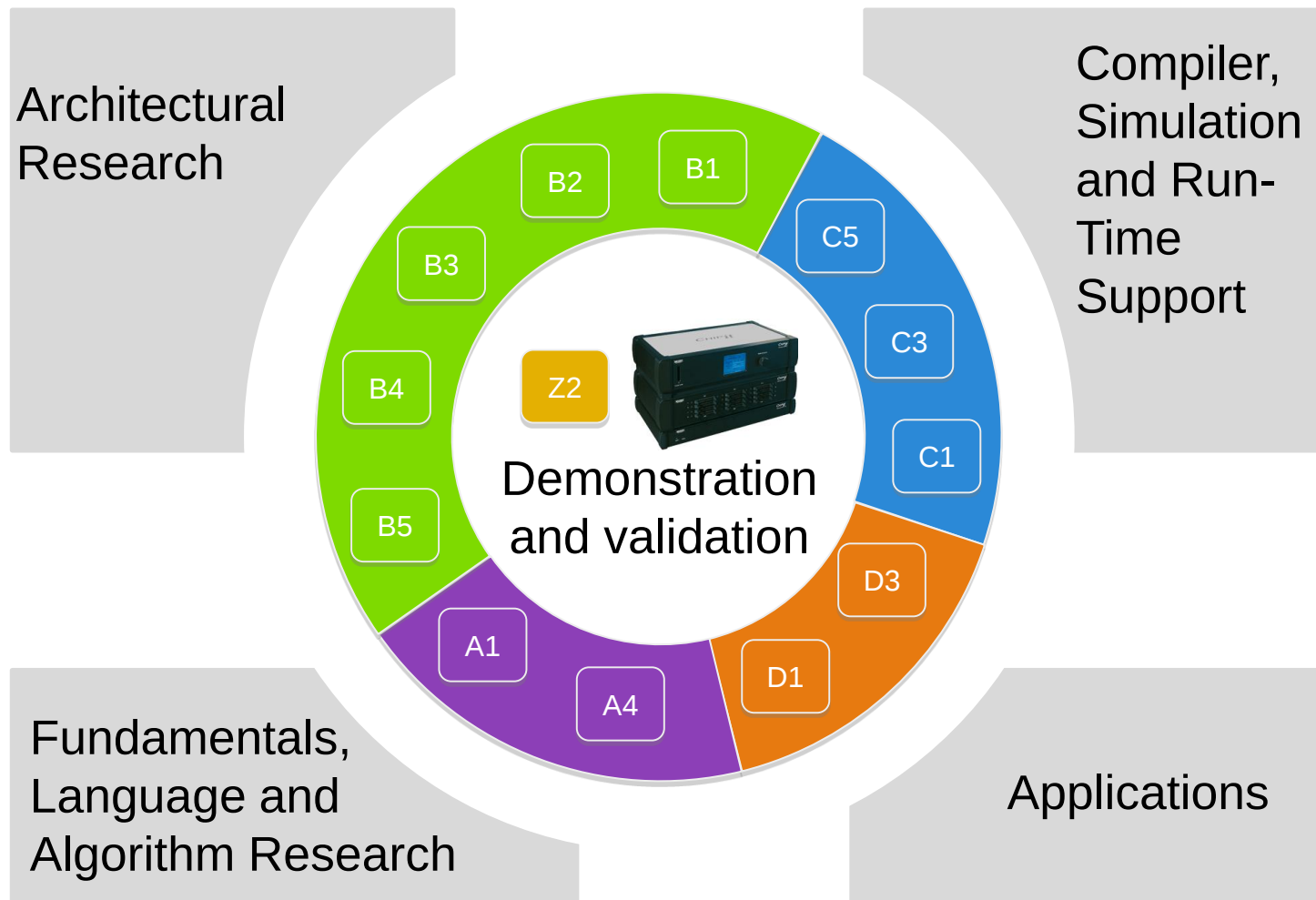
Invasive Computing (*Invas/C*) – Project Goals

- Novel paradigm for designing and programming future parallel computing systems with hundreds of cores
- Decentralized and self adaptive hardware and software
- Ability of applications to invade resources, spread workload and release them after execution
- Invasive programming supports resource-aware computing through:
 - Language & compiler support
 - Operating system support
 - Hardware extensions



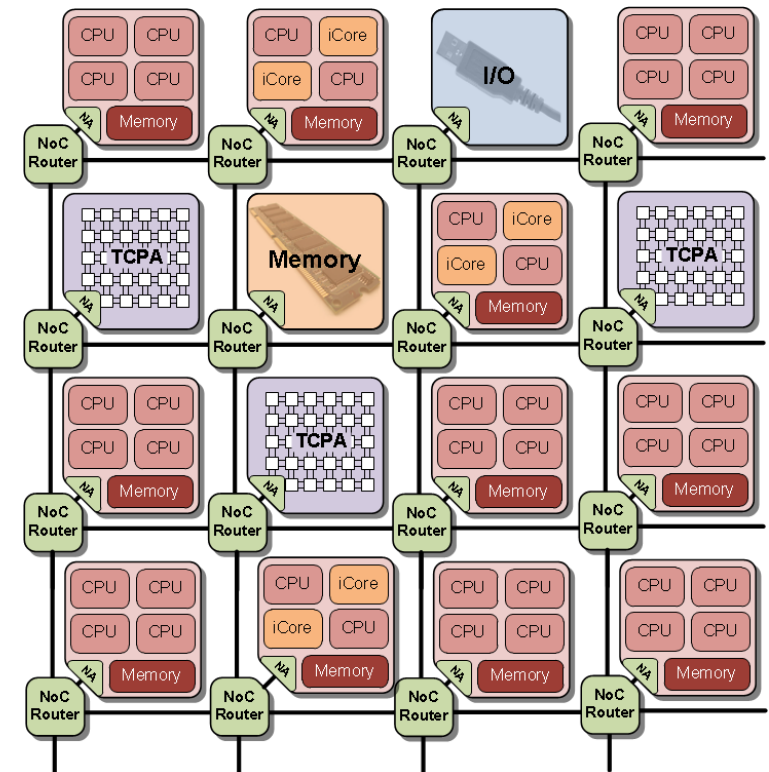
Invasive Computing (*Invas/C*) – Overview

- 13 Subprojects in 4 Project Areas



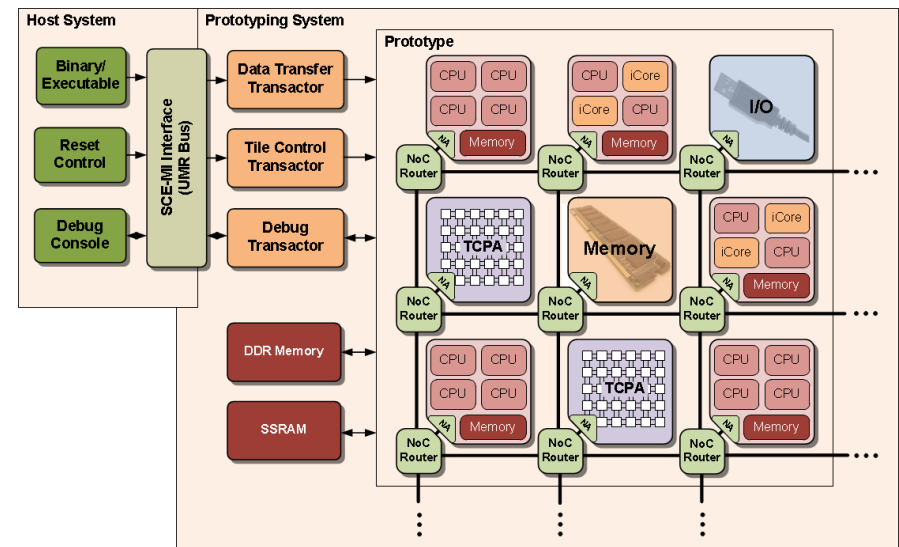
Invasive Hardware Architecture

- **Heterogeneous hardware architecture:**
 - Different processing tiles
 - Memory and I/O tiles
- **Interconnected through the invasive Network on Chip**
 - Distributed Routing
- **HW/SW-Co-Design Challenges for Interconnect:**
 - Scalability & Flexibility
 - Self-Organization and Self-Optimization



Invasive Computing (*Invas/C*) – Demonstrator and Validation

- FPGA based prototyping
- Demonstration of complex interaction between hardware and operating system
- Prototyping platform: proFPGA and chipit
 - High speed connection to Host PC



ITIV Themen

■ Was?

- Abschlussarbeiten, Hiwi Tätigkeiten oder Kombination aus Beidem
- Für Informatiker und Elektrotechniker gleichermaßen
- Vielfältige Hardware/Software Co-Design Themen von High-Level Modellierung bis Low-Level Hardware

■ Wann?

- Am besten so früh wie möglich!
 - First come, first serve

■ Wie?

- Email schreiben
 - Leonard.Masing@kit.edu
- nach der Übung melden



WE WANT YOU!

Arbeitsphase

- Aufgabe 1.01: Hardware/Software Co-Design
 - Diskussion von Vor- und Nachteilen von Hardware/Software bezüglich verschiedener Kriterien

- Aufgabe 1.02: Architekturen
 - Wichtige Kriterien bei der Realisierung von Zielarchitekturen
 - c) ist Bonusaufgabe

- Diskutieren Sie die Vor- und Nachteile in der Implementierung eines Systems in Hardware bzw. Software bzgl. folgender Kriterien:
 - a) Entwurfszeit bzw. time-to-market
 - b) Performanz
 - c) Kosten
 - d) Leistungsverbrauch
 - e) Wartbarkeit bzw. Änderbarkeit
 - f) Testbarkeit
 - g) Sicherheit

Lösung Aufgabe 1.01a: Hardware/Software Co-Design - Entwurfszeit bzw. Time-to-market

- HW --
 - Lange Entwicklungszeiten für full custom ASICs und Boards; Langer externer Fabrikationsweg (z.B. für Waver); „First time right“; Hoher Testaufwand
- HW -
 - Einsatz von off-the-shelf components/module, dadurch i.a. komplizierteres Board
- SW ++
 - Erste Lösungen sind sehr schnell fertig; Fertigstellung kann in Stufen erfolgen
- SW +
 - Softwarefehler können Post-Fabrication durch Softwareupdates beseitigt werden. (WLAN)
- HW +
 - Einsatz von konfigurierbarer Hardware (FPGAs). Post-Fabrication HW Updates
- HW/SW +
 - Gleichzeitige Entwicklung möglich

Lösung Aufgabe 1.01b: Hardware/Software Co-Design - Performanz

- HW ++
 - Nichts ist schneller als ein full-custom ASIC

- HW +
 - FPGAs sind sehr schnell, vor allem wenn Parallelität ausgenutzt werden kann

- SW --
 - Bestimmte Operationen können nur von dezidiertter Hardware ausgeführt werden (z.B. restriktive Echtzeit-Bedingungen)

- SW +
 - Die Rechenleistung kann auf mehrere Prozessoren verteilt werden

- SW +
 - Optimierung durch komplexere Algorithmen, die nur teilweise in HW realisierbar sind

Lösung Aufgabe 1.01c: Hardware/Software Co-Design - Kosten

- SW ++
 - Programmierkenntnisse sind weit verbreitet; Compiler sind ausgereift und leicht zu bedienen; Änderungen an der Sprache sind relativ selten; Geringe Tool-Kosten
- SW ++
 - Off-the-shelf Mikrokontroller und Zubehör sind bedingt durch die hohen Stückzahlen billig
- SW ++
 - Gefahr eines totalen Re-Designs ist klein
- HW --
 - Hohe Tool und Personalkosten durch lange Entwicklungszeiten
- HW --
 - ASICs lohnen sich nur für große Stückzahlen
- HW +
 - Rekonfigurierbare Hardware minimiert die Entwurfskosten im Vergleich zum ASIC-Entwurf
- HW +
 - Rekonfigurierbare Hardware kann benötigte Chipfläche reduzieren (dyn. Rekonfiguration, Compute in „Time & Space“)

Lösung Aufgabe 1.01d: Hardware/Software Co-Design - Leistungsverbrauch

- HW ++
 - Hardware oft speziell auf das Design angepasst, verbraucht nur die absolut notwendige Leistung

- HW ++
 - Nicht benötigte Teile können abgeschaltet werden

- SW --
 - Hohe Taktraten für die Erfüllung von Echtzeitbedingungen notwendig

- SW --
 - Software verbraucht in der Regel mehr Leistung als Hardware. (dynamisches Steuerwerk, Programmspeicher, Caches, etc.)

- HW +
 - Rekonfigurierbare Hardware vergleichsweise viel Energie
 - Abhilfe: Low Power Flash FPGAs (Igloo FPGA von Altera)

Lösung Aufgabe 1.01e: Hardware/Software Co-Design - Wartbarkeit / Änderbarkeit

- HW --
 - ASICs sind oft nicht-modulare Einzellösungen (→ IP Re-use)

- HW --
 - Ein ASIC-Layout ist fest, keine nachträgliche Änderung

- HW ++
 - FPGAs sind zur Laufzeit konfigurierbar (rekonfigurierbar)

- SW ++
 - Fast beliebige Änderungen möglich

- HW/SW +
 - Rekonfigurierbare HW ermöglicht Änderung von SW & HW

Lösung Aufgabe 1.01f: Hardware/Software Co-Design - Testbarkeit

- HW ++
 - CAD-Tools helfen sehr bei Entwurf und Synthese von korrekten Schaltungen
- HW +
 - Zusatzlogik zum Testen und Debuggen kann direkt in die Schaltung integriert werden (JTAG Interface)
- HW --
 - Simulation der HW sehr Zeitaufwändig und Ressourcenhungrig
- SW ++
 - Sehr Aufgereifte Entwicklungsumgebungen zum Debuggen, Testen und Simulieren vorhanden.
- SW -
 - Sehr komplexe Programme möglich; Oft kein sauberes Design (→ Software-Engineering).
- HW/SW +
 - Rapid-Prototyping mit rekonfigurierbarer Hardware möglich
 - HiL (Hardware in the Loop)
- HW/SW --
 - Oft keine geeigneten Hilfsmittel für den Test von gemischter Hardware/Software
 - Aktuelles Forschungsgebiet: Co-Design, Co-Simulation

Lösung Aufgabe 1.01g: Hardware/Software Co-Design - Sicherheit

- HW ++
 - Redundanz durch Einsatz mehrere gleichartiger Komponenten
- HW ++
 - Hardware ist schwierig zu kopieren (Custom-ASICs, Multilayer Boards)
- SW --
 - Software ist relativ einfach zu kopieren oder verändern, Schutzmechanismen aufwendig (iPhone)
- HW/SW +
 - Software passt nur zur entsprechenden Hardware
- HW/SW ++
 - Realisierung von SIL4 (Safety Integrity Level) durch redundanter Realisierung in HW und SW.

- Hardware/Software Co-Design
 - Entwurfszeit bzw. time-to-market
 - Performanz
 - Kosten
 - Leistungsverbrauch
 - Wartbarkeit bzw. Änderbarkeit
 - Testbarkeit
 - Sicherheit



Aufgabe 1.02: Architekturen

- Überlegen Sie sich Kriterien, welche die Entscheidung der Realisierung einer Spezifikation in Hardware oder Software begünstigen.
 - a) Was sind die wichtigsten Kriterien im Falle einer Zielarchitektur für
 - 1) Steuerung einer Ampel
 - 2) Mobiltelefon
 - 3) System zur Bildverarbeitung
 - 4) Kraftwerksüberwachung?
 - b) Welche unterschiedlichen Optimierungskriterien machen die Entscheidung Hardware/Software aus im Falle einer
 - 1) Ein-Chip Hw/Sw-Lösung
 - 2) Board-Level Hw/Sw-Lösung
 - c) Für welche Anwendungsbereiche erscheinen Ihnen ASIPs (Prozessoren mit anwendungsspezifischem Instruktionssatz) sinnvoll?

Lösung Aufgabe 1.02a: Architekturen

- Kriterien für die Entscheidung für eine Hardware- oder Softwarelösung für einige Beispiele:
 - 1) Steuerung einer Ampel
 - Steuerungsdominant, Sicherheit, Einfache Konfigurierbarkeit, Umweltresistenz, Größe
 - 2) Mobiltelefon
 - Steuerungs- und Datenfluss, Geringer Leistungsverbrauch, Größe, Stückzahl, zu realisierende Dienste (QoS)
 - 3) System zur Bildverarbeitung
 - Datenflussdominant, Performanz, Durchsatz
 - 4) Kraftwerksüberwachung?
 - Steuerungsdominant, absolute Sicherheit, Redundanz, Reaktionszeit, Harte Echtzeitbedingungen

Lösung Aufgabe 1.02b: Architekturen

- Kriterien für die Entscheidung für eine Hardware/Software Implementierung als

1) Ein-Chip Lösung

- Kosten (nur günstig bei großen Stückzahlen), Gewicht, Größe, Zuverlässigkeit (Schirmung, Konnektoren), Leistungsverbrauch, interner vs. externer Kommunikationsaufwand, Kopierschutz

2) Board-Level Lösung

- Erfüllbarkeit (Passt nicht auf einen Chip), Kosten (Standardchips sind günstiger), Entwurfszeit, Flexibilität, Verlässlichkeit

Lösung Aufgabe 1.02c: Architekturen

Punkte, die für den Einsatz eines ASIPs statt einer Standard-CPU sprechen:

- Kostengünstiger bei großen Stückzahlen, Leistungsverbrauch geringer, Operationsverkettung möglich, Parallelität möglich, spezialisierte Funktionen, Anpassung der Wortlänge, optimierte Speicherstrukturen, optimierter Datenpfad, Spezialregister.
- Aufgaben fest umrissen, keine Änderung wahrscheinlich, speziell zugeschnittener Befehlssatz erlaubt erhöhte Performanz durch Einsatz dedizierter Funktionseinheiten (heterogener vs. homogener Registersatz)

- Architekturen
 - Wichtigsten Kriterien für die Auswahl von Zielarchitekturen für verschiedene Anwendungsklassen
 - Unterschiedliche Optimierungskriterien je nach HW/SW Co-Design
 - Anwendungsbereiche für ASIPs

