Eingebettete Prozessorarchitekturen

11. Architekturen und Entwurf für Adaptive Computer

Andreas Koch

FG Eingebettete Systeme und ihre Anwendungen
Informatik, TU Darmstadt

Wintersemester 2010/2011

Zwischenstand

- Einführung Adaptive Computersysteme (ACS)
- Anwendungen
- ACS-Komponenten

A. Koch

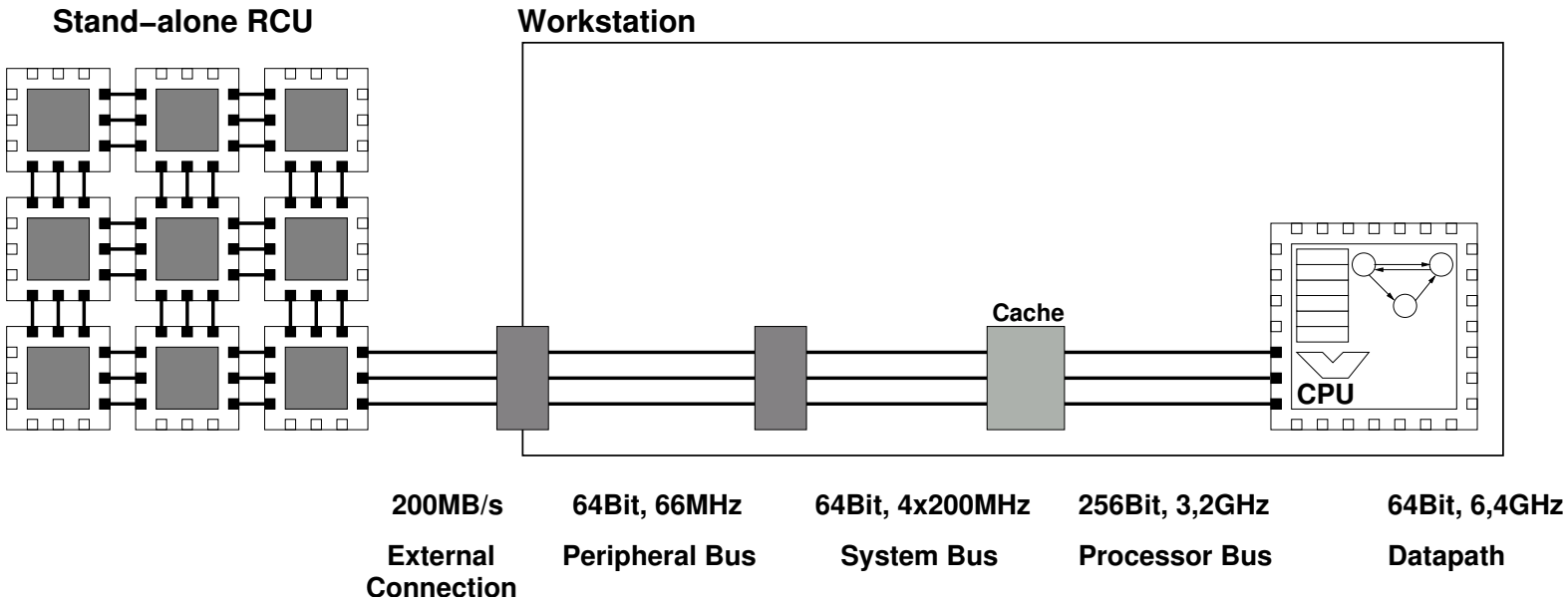
Systemarchitektur

ACS-Komponenten

- Reconfigurable Compute Unit (RCU)
  - Besteht aus Reconfigurable Area (RA)
  - Verschiedene Ausprägungen
  - Unterschiedliche Logikblock-Granularität
- Software-programmierter Prozessor (SPP)
  - Kann auch langsameres Modell sein
  - Nur für Grundlast der Rechenleistung verantwortlich
- Speicher
  - Gemeinsam zwischen RCU und ISP nutzbar
  - Optional: Dedizierter Speicher
- Optional: Schnelle I/Os direkt an RCU

➥Wie Komponenten verknüpfen?

Freistehende RCU

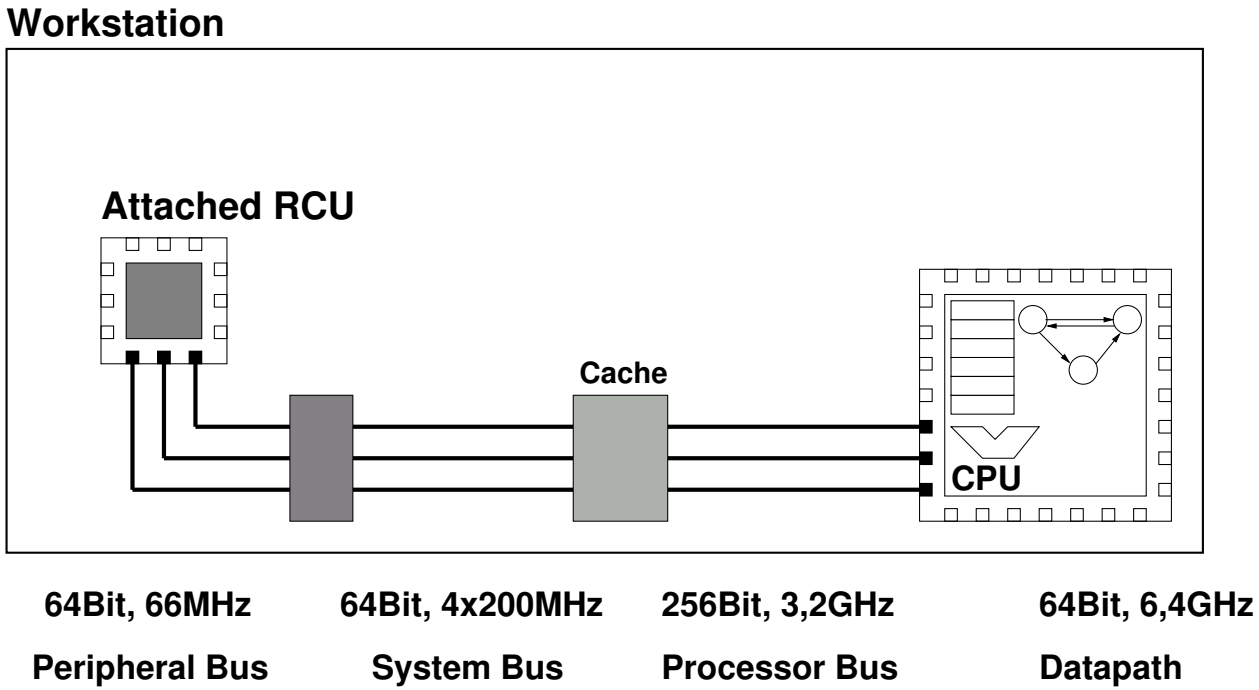


A. Koch

- Benutzt z.B. bei ASIC-Emulation
- Große Kapazität ($>$ 500 Mio. Gatter)
- Aber wenig allgemein verwendbar

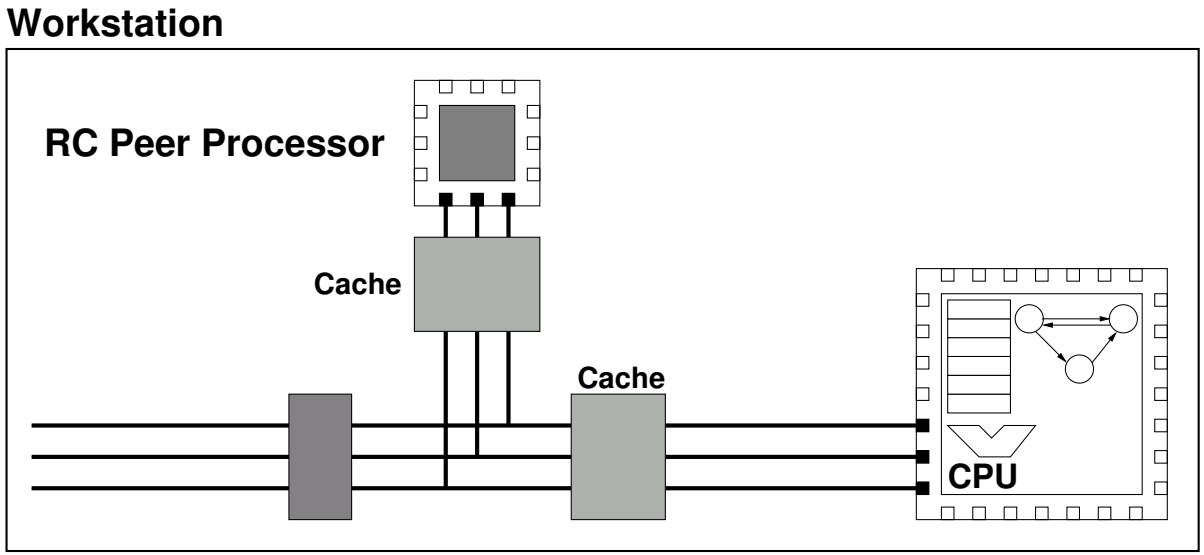
Angeschlossene RCU

A. Koch



- An Peripheriebusse angeschlossen
- PCI, VME, SBus, PCI Express
- Standardbusse, RCU damit leicht anschliessbar
- Üblichste Methode der RCU-Integration
- Aber immer noch langsame Kommunikation
  - PCI schreiben: 10 Takte, lesen: 30 Takte

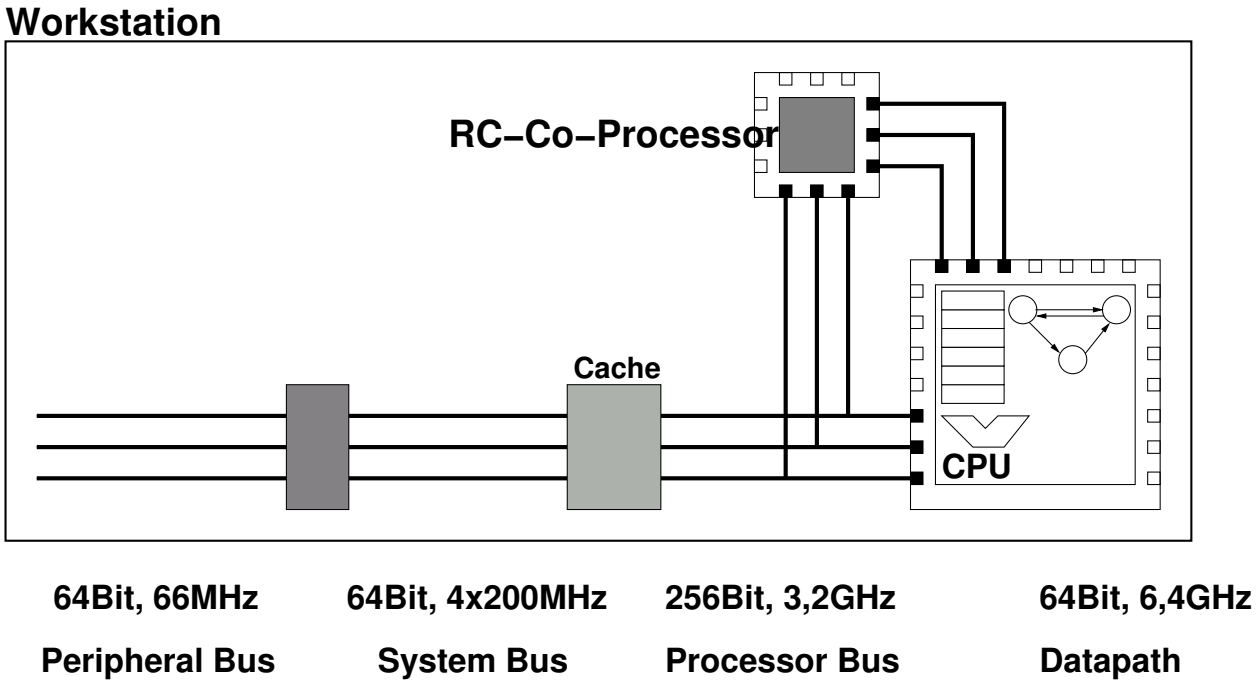
Multiprozessorsystem mit RCU



A. Koch

- RCU nun gleichberechtigter Partner
- Erhöhter Datendurchsatz, reduzierte Latenz
  - Schwierigkeiten: Z.B. Cache-Kohärenz
- Lange angedacht, mittlerweile verfügbar
  - Hypertransport/HTX bis zu 3.2 GB/s
  - FSB bis zu 8.5 GB/s
  - QPI (aktuelle Entwicklung) bis zu 25.6 GB/s

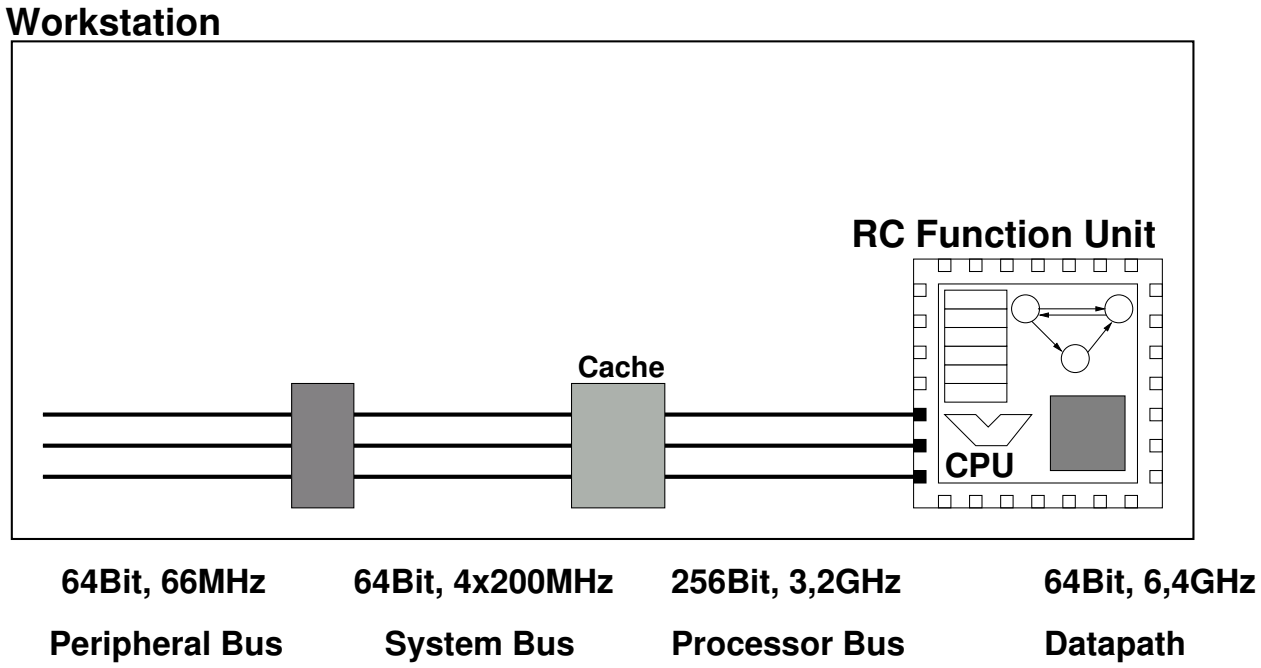
RCU als Koprozessor



A. Koch

- Anschluss an internen Prozessorbus
- Keinen Zugriff auf CPU-Register
- Aber oft gemeinsamer Cache (i.d.R. L2-Ebene)
- Weniger Kohärenz-Schwierigkeiten
- Noch mehr Bandbreite bei weiter reduzierter Latenz
- Experimentell: UC Berkeley GARP (MIPS + RCU)

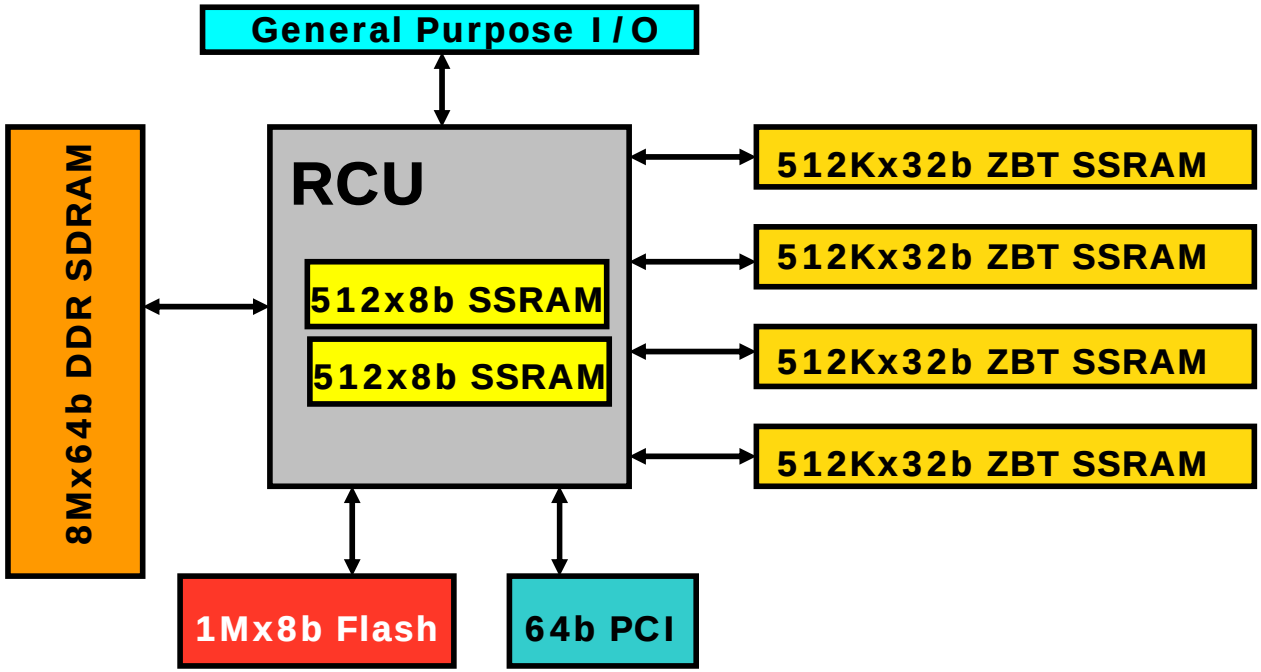
RCU als Funktionseinheiten in CPU



A. Koch

- Direkt in Prozessor integriert (RFU)
- Sehr niedrige Latenz
- Problem: Beschränkter Datendurchsatz
  - Rechnet i.d.R. auf 2-3 Registern pro Instruktion
  - Nur einfache Berechnungen je Instruktion
  - I.d.R. hat RFU niedrigere Taktfrequenz als CPU
- Benötigt konfigurierbaren Prozessorkern als Basis

Schnittstellen und Speicher

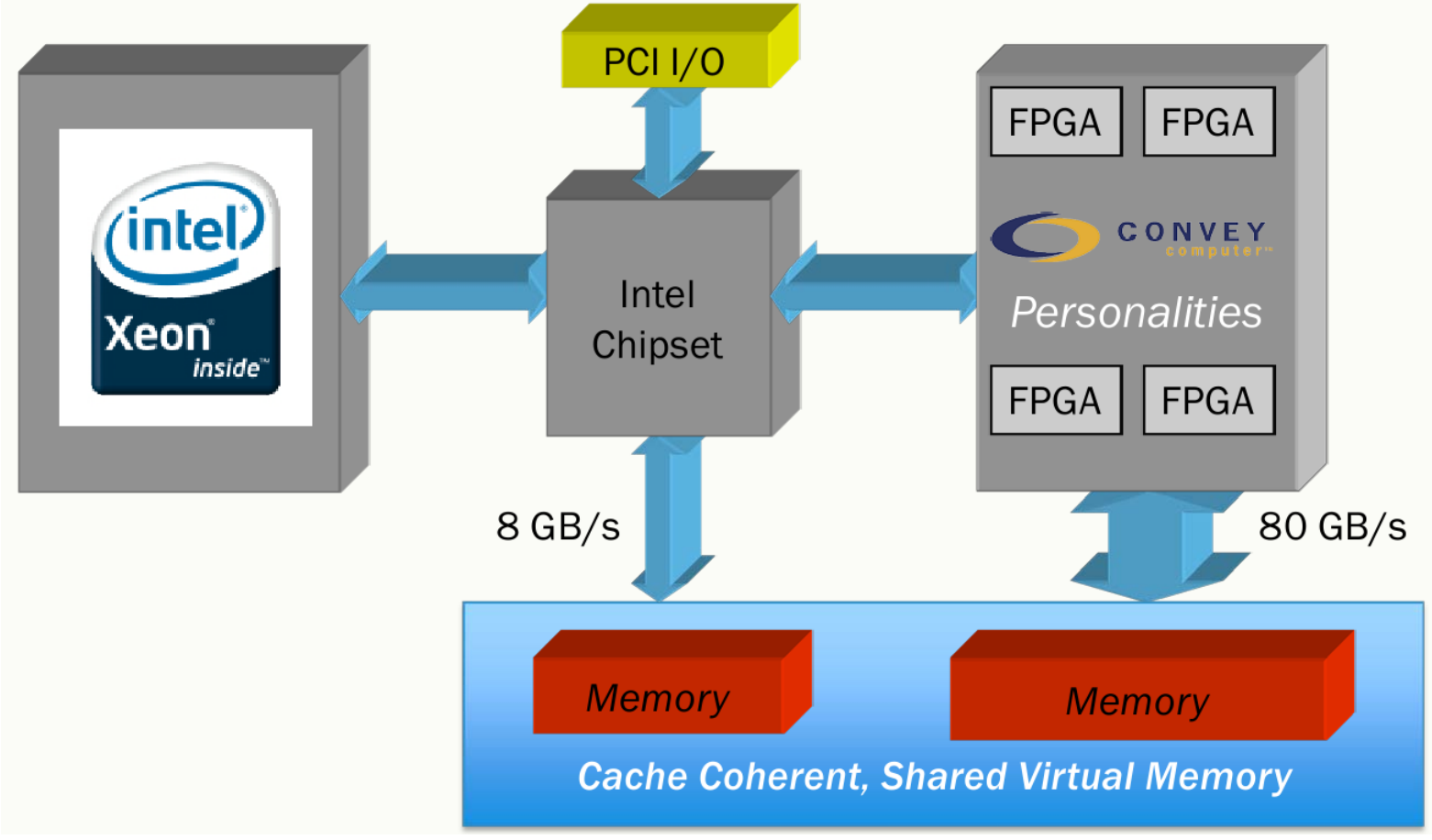


- Heterogene Speicher
  - Ort: On-Chip/Off-Chip
  - Typ: DRAM, SRAM, Flash
  - Variante: DDR1/2/3, QDR, ZBT, RL, ...
- Organisiert als mehrere Bänke
  - Eigene Adressräume, parallel zugreifbar

Beispiel für heterogenen Multiprozessor: Convey HC-1ex

Grundarchitektur

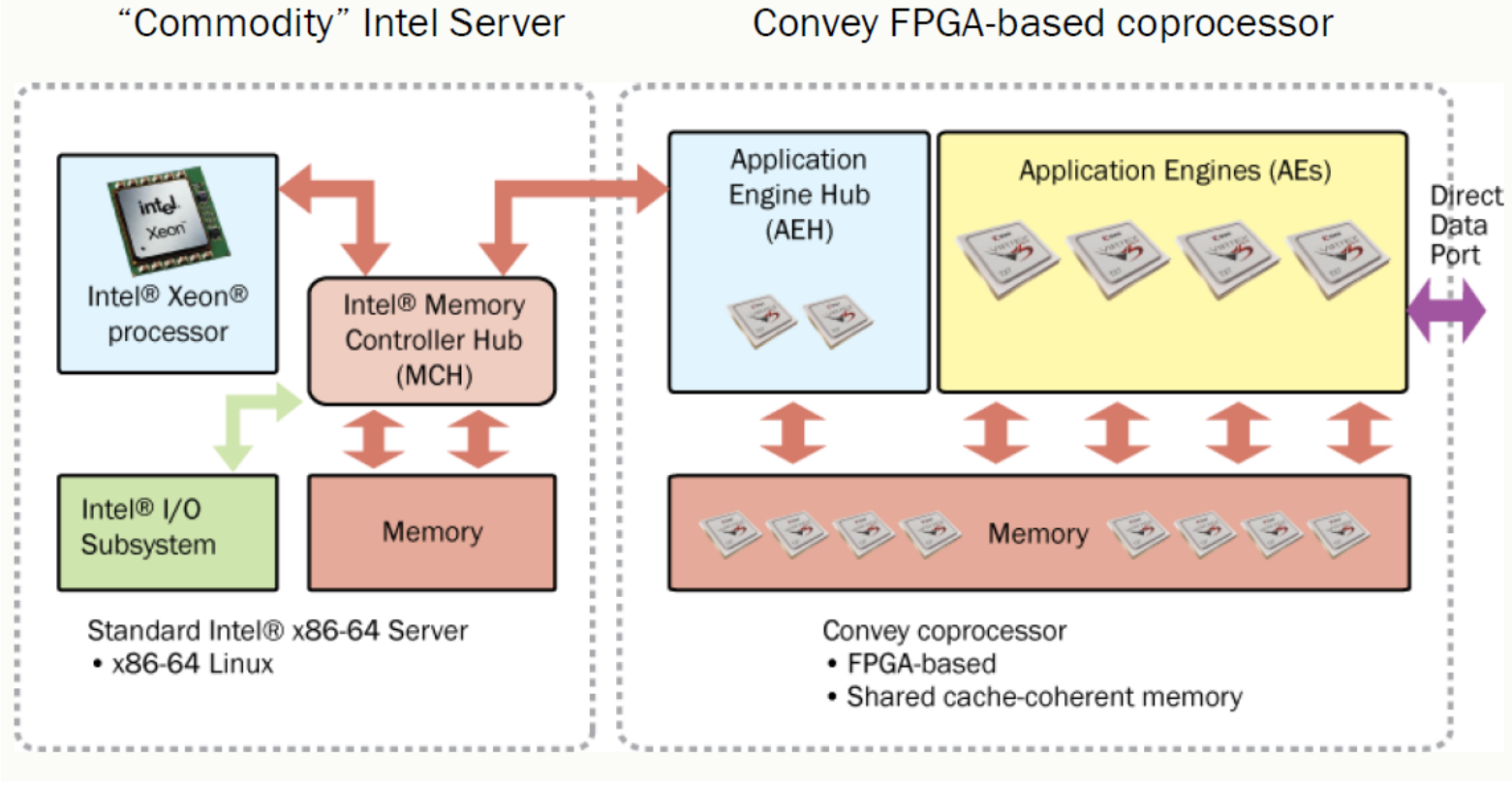
A. Koch



Beispiel für heterogenen Multiprozessor: Convey HC-1ex

Detailarchitektur: Speicher

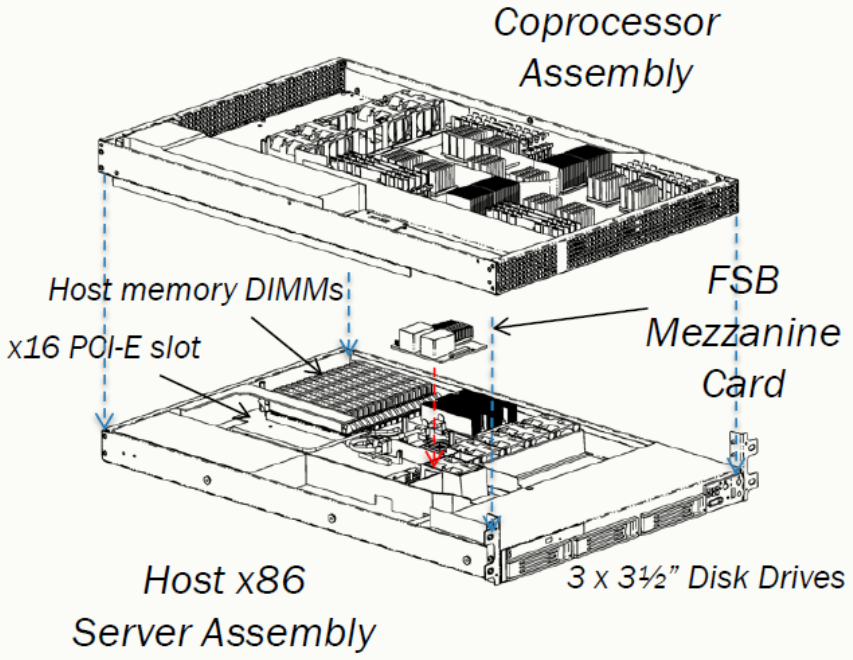
A. Koch



Beispiel für heterogenen Multiprozessor: Convey HC-1ex

Detailarchitektur: Mechanischer Aufbau

A. Koch



- **2U enclosure:**
  - Top half of 2U platform contains the coprocessor
  - Bottom half contains Intel motherboard

Anwendungseignung der Architekturen

Grobe “Pi-mal-Daumen” Richtlinien

Anbindung	Minimale effektive Berechnungsdauer	I/O Durchsatz
Alleinstehend	Sehr lang (10s)	Sehr niedrig
Angeschlossen	Lang (≈10ms)	Mittel
Multiprozessor	Mittel (≈100us)	Hoch
Koprozessor	Kurz (≈1us)	Hoch
Funktionseinheit	Sehr kurz (≈10ns)	Niedrig

Terminologie

Terminologie 1

- Konfigurierbarkeit
  - Struktur einer Recheneinheit anpassbar an Problem
  - Erlaubt räumliche Verteilung der Berechnung
    - Hardware-Beschleuniger für Software-Operationen
    - Beispiele: Tensilica Xtensa, ARC ARCtangent
- Rekonfigurierbarkeit
  - Konfigurierbarkeit nach Fertigung der Hardware
- Dynamische Rekonfigurierbarkeit
  - Rekonfiguration während des Betriebs
  - Auch *run-time reconfiguration* (RTR) genannt
- Partielle Rekonfiguration
  - RA kann auch teilweise rekonfiguriert werden
  - I.d.R. schneller als vollständige Rekonfiguration
- Programmierung
  - Änderung des Verhaltens bei gleicher Struktur

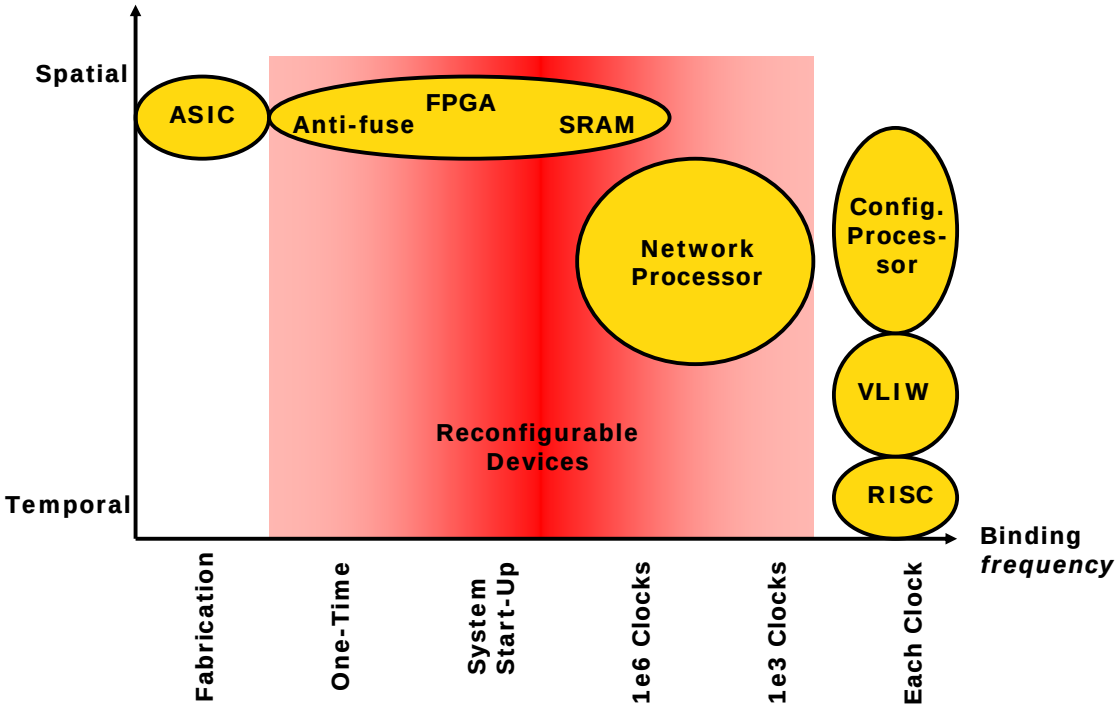
Diskussion

- FPGA häufig rekonfigurierbar betrieben
- Dynamische Rekonfiguration noch selten
  - Langsam (einige ms pro Rekonfiguration)
    - Xilinx Virtex 5: ca. 5ms/30K LUTs
  - Nur wenig Werkzeugunterstützung (wird aber besser . . .)
- Aber kombinierte Ansätze möglich
  - Konfigurierbare CPUs mit RCU
    - Stretch S5000
    - ST Chip mit Xtensa und FLEXEOS FPGA
  - Einige ASICs können teilweise rekonfiguriert werden
    - eASIC hat rekonfigurierbare Logik (kein Routing) im eASICore
  - Programmiere neue Werte in RCU Register
    - Viel schneller als Rekonfiguration

Terminologie 2

- Granularität
  - Funktionalität der einzelnen Recheneinheiten
  - Verbreitet: Wertetabellen (LUTs), ALUs, komplette Prozessoren
- Bindungsintervall
  - Kürzestes Zeitintervall zwischen Funktionsänderungen
  - Kann theoretisch sein (beim ASIC: ∞)
  - Auch verwendet: Bindungsfrequenz
    - Kehrwert von Intervall

Granularität und Bindungsintervall



A. Koch

- Gröbere Granularität $\rightarrow$ kürzere Bindungsintervalle
- Kürzere Bindungsintervalle $\rightarrow$ bessere Wiederverwendung von Ressourcen
- Wähle passende Technologie für Anwendung

ACS Programmierung

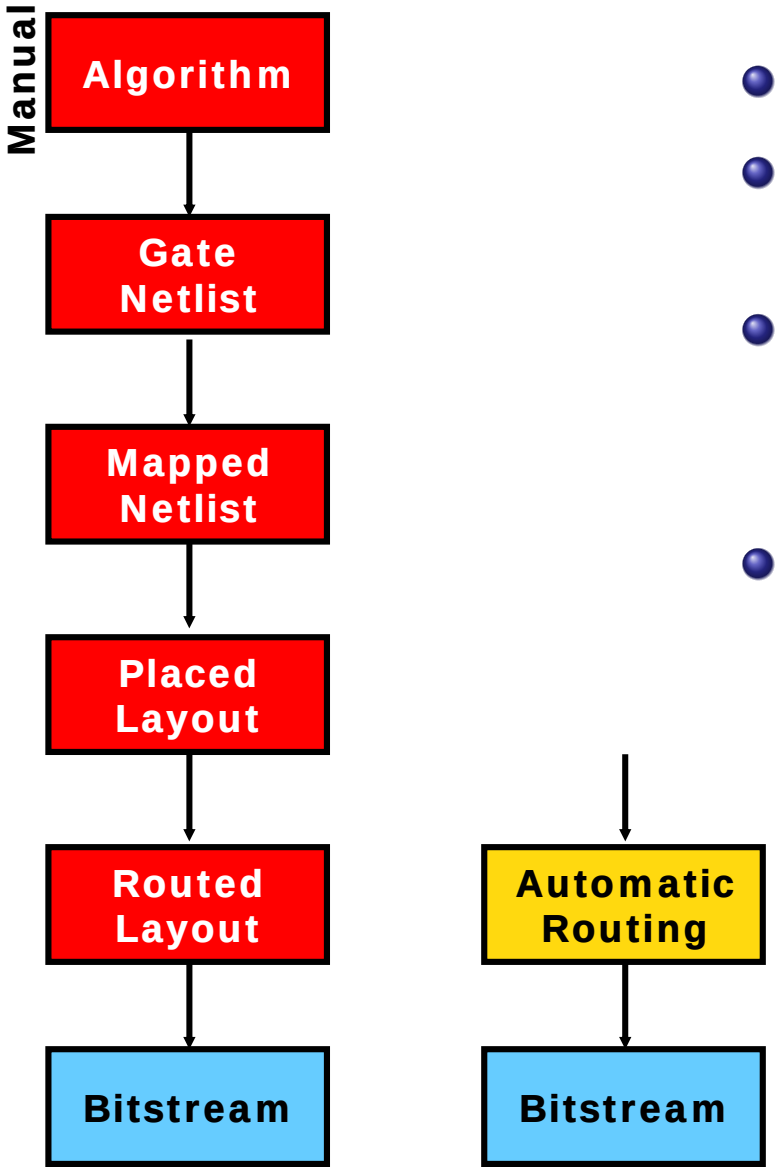
Freiheitsgrade

- Nur Hardware oder Hardware+Software-Teile beschreiben?
- Werkzeugunterstützung: Manuell . . . Vollautomatisch
- Art der Beschreibung angepasst an Berechnungsmodell
  - Datenflußdiagramme (Signalverarbeitung)
  - Zustandsautomaten (z.B. Harel-Diagramme)
  - Imperative Programmiersprachen
  - Strukturelle Beschreibung (z.B. Schaltpläne)

Hardware und/oder Software

- Hängt von Anwendung ab
- Szenarien für reine Hardware-Beschreibung
  - Hochgeschwindigkeitsschnittstellen (I/O)
    - Ereignisdetektion im Teilchenbeschleuniger
  - Glue-Logic zum Verknüpfen anderer Chips
  - Einfache Zustandsautomaten
    - Ampelsteuerung, Toaster, Kaffeeautomaten
- Szenarien für kombinierte Hardware/Software-Beschreibung
  - Rechenintensive Algorithmen (*kernels*) auf RCU
    - Vergleichsweise kleine Teile (verschachtelte Schleifen)
    - Verarbeiten oft Datenströme (reguläre Zugriffsmuster)
  - Komplexer Kontrollfluß in Software verbleibt auf CPU
    - Bis hin zur Betriebssystemebene

Manueller Entwurfsfluß

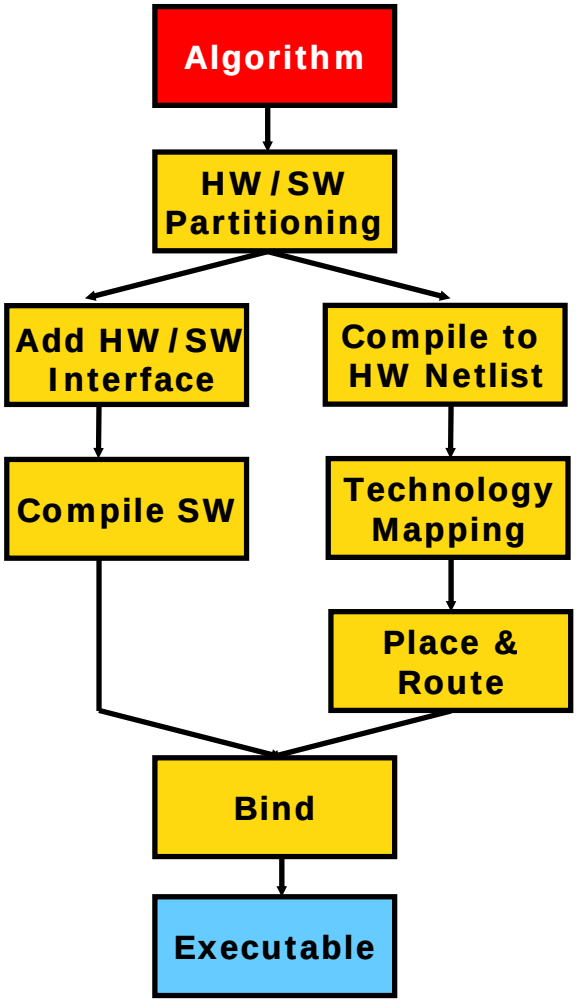


- Nur noch selten so verwendet
- Gelegentlich noch für Hochgeschwindigkeitsanwendungen
- Fein abgestimmte Hardware-Blöcke
  - Wiederverwendet
- I.d.R. mit automatischem Routing

A. Koch

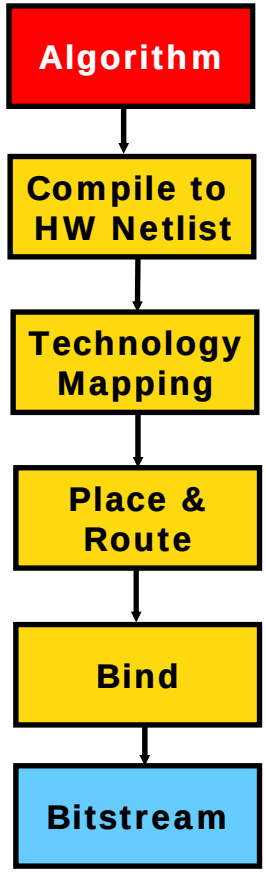
Vollautomatische Entwurfsflüsse

Übersetzung von Algorithmen in . . .



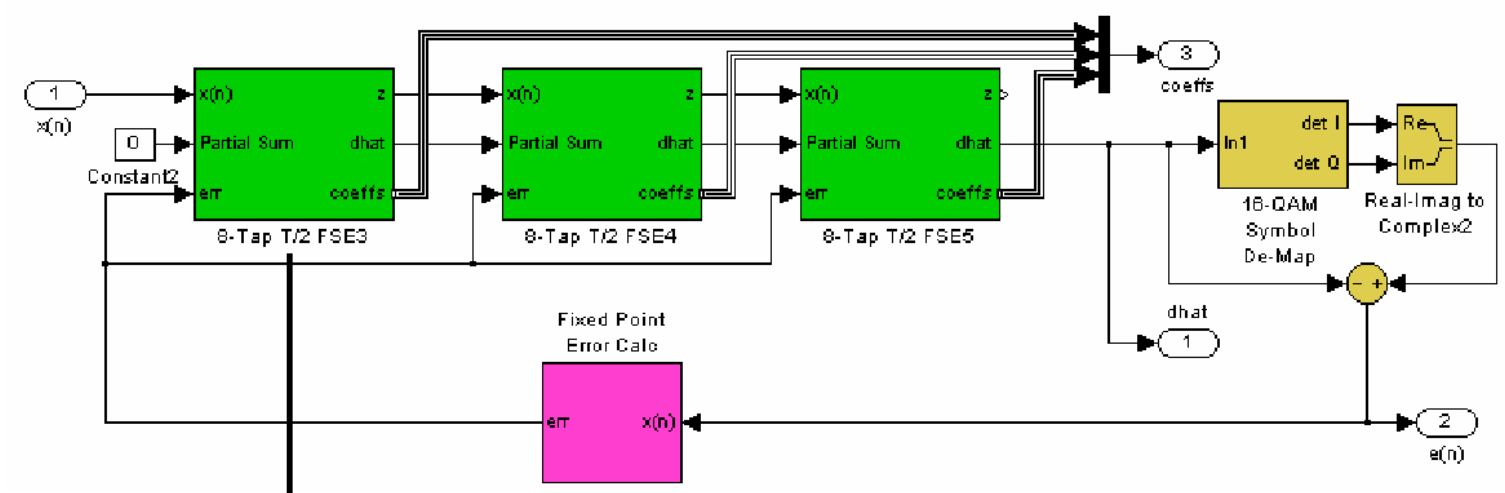
- HW+SW
  - GarpCC
  - NimbleC
  - COMRADE
  - Nymble
- Reine HW
  - HDL Synthese
  - Synopsys SynphonyCC
  - Xilinx AutoPilot
  - UCR ROCCC
- Oft
  - Manuelle Partitionierung

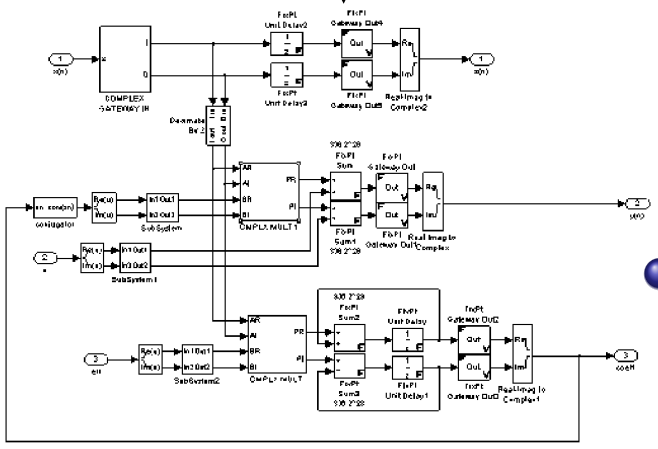
A. Koch



Graphische Beschreibung

A. Koch





- Verbinde Funktionsblöcke
  - Hierarchisch und parametrisiert
  - Ptolemy, SPW, Simulink, Schaltplan
- Beispiel: Fractionally Spaced Equalizer
  - In Simulink
- Oft in DSP + Telekommunikation

Quelle: C. Dick, H.M. Pedersen

Textuelle Beschreibung

Embedded Systems & Applications

FIR in ANSI C

```
fir(  int input[],
      int coef[], int nCoef,
      int output[], int nOut )
   {
     int i, j;
     int  sum;

     for (j =  0; j < nOut; j++) {
       sum =  0;
       for (i =  0; i < nCoef; i++){
         sum +=  input[j+i] * coef[i];
       }
       output[j] =  sum >> 15;
     }
   }
```

FIR in SilverC

```
void run (void)
{
  fract16 sum;
  loop (int l=0; l<nOut; l++) dataflow {
    sample = input.read();
    sum = 0.0;
    unroll (int i=0; i<nCoef; i++) {
      sum = sum + coefReg[i] * sample[nCoef-i];
    }
    output.write(sum);
  }
}
```

A. Koch

- Sehr hohe Programmiersprachen: MATLAB
- Konventionelle Programmiersprachen: C, Java
- Spezialisierte ACS-Sprachen:
  - Lava, Pebble, Handel-C, SilverC
- Hardware-Beschreibungssprachen:
  - Verilog, VHDL, System-C

Quelle: Quicksilver Tech

HDL-basierte Programmierung

- Am weitesten verbreitete Programmiermethode
- Beschreibe HW-Teile auf RCU in HDL
  - Üblicherweise auf Register-Transfer-Ebene (RTL)
  - Einige strukturelle Konstrukte für Spezialblöcke
    - Multiplizierer, On-Chip-Speicher, DLLs/PLLs
- Beschreibe SW-Teile auf CPU in Hochsprache (HLL)
  - C, C++, selten Java
- Leidlich robuster Werkzeugfluß
  - HLL Kompilierung
  - HDL Synthese
  - Technologieabbildung, Platzierung, Verdrahtung,
  - Simulation

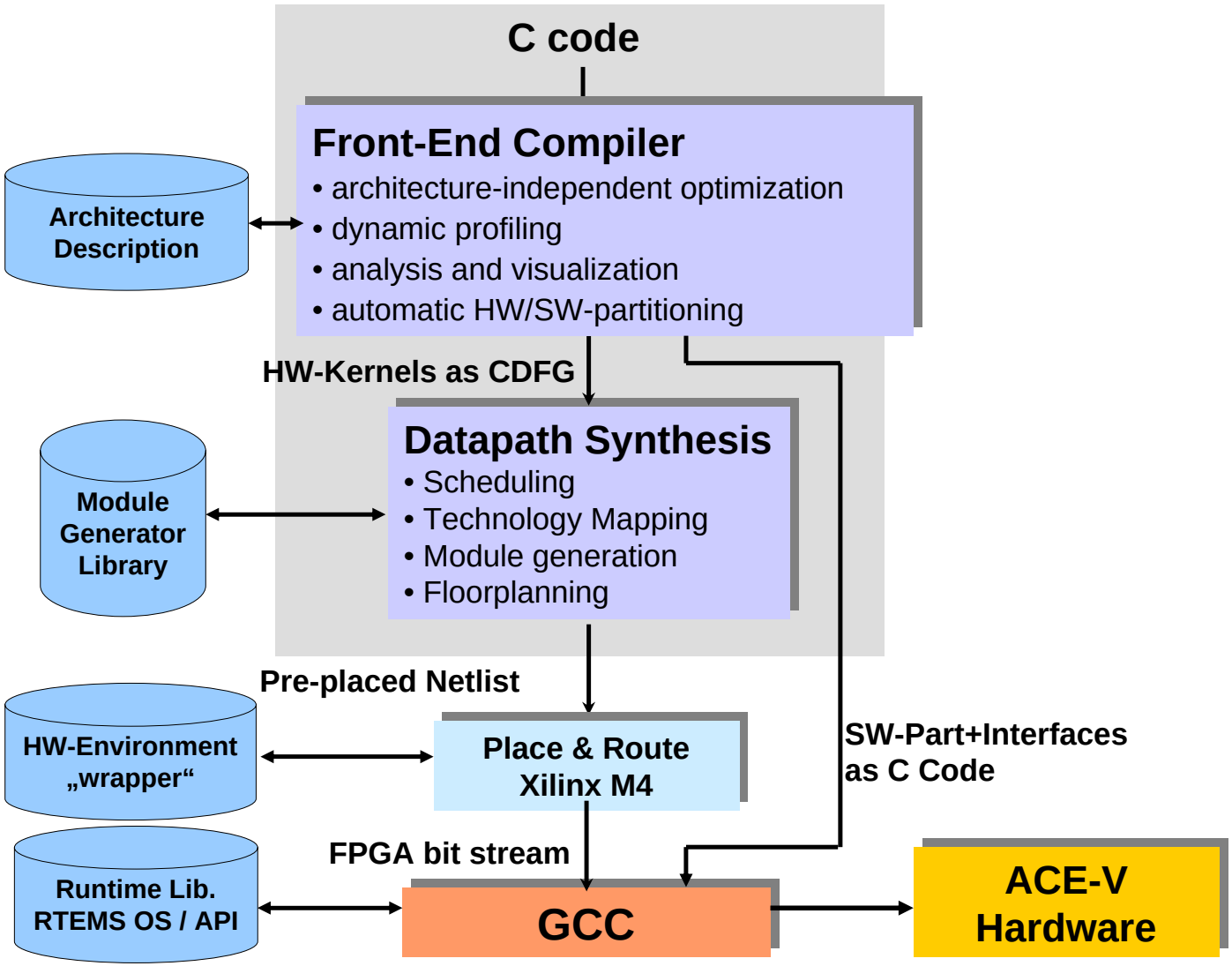
➥Nächste Vorlesung im Detail

Ggf. vorher noch Verilog wiederholen (TGDI, CMS)!

Automatische HLL Übersetzung

Experimenteller Compiler-Fluß (aktuelles Forschungsthema!)

A. Koch



Beispielprogramm: $j \cdot 13^k$

Verkompliziert durch Prüfung auf Wertüberschreitung

Embedded Systems & Applications

A. Koch

```
void
main(int argc, char *argv[])
{
  int i, j, k;

  // Integer value of the first command line parameter
  j = atoi(argv[1]);
  // Integer value of the second command line parameter
  k = atoi(argv[2]);

  for (i = 0; i < k; i++)
    {
      j = j * 13;
      if (j > 1000000)
        printf("j=%d too large in loop i=%d\n", j, i);
    }

  printf("result: j = %d\n", j);
}
```

Sample execution

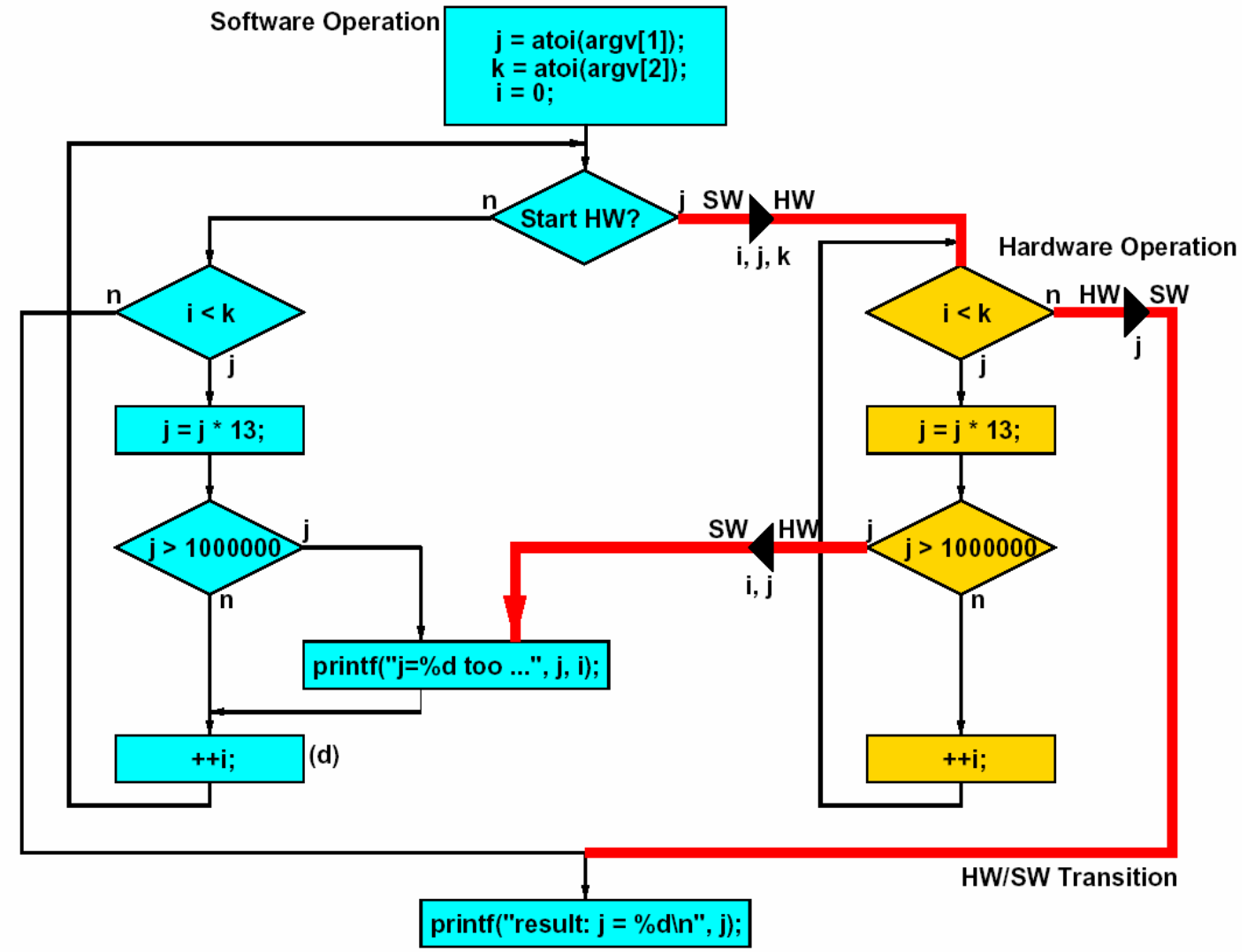
```
$ ./a.out 10 5
j=3712930 too large in loop i=4
result: j = 3712930
```

Hardware/Software-Partitionierung
Aufteilen des Programmes auf CPU und RCU

- Dynamisches Profiling erkennt Kernel
  - Problem: `printf()` nicht effizient auf RCU realisierbar
- Die meisten Werkzeuge geben hier auf!
  - Bestenfalls wird eine aussagekräftige Meldung angezeigt
- Anderer Ansatz
  - Bestimme, wie oft nicht-RCU-Anweisung tatsächlich ausgeführt wird
    - Datenabhängig!
  - Falls sie nur selten auftritt $\rightarrow$ RCU doch noch verwenden
  - Ausnahmefall muß aber korrekt bearbeitet werden

➥HW und SW-Versionen des Kernels bereithalten

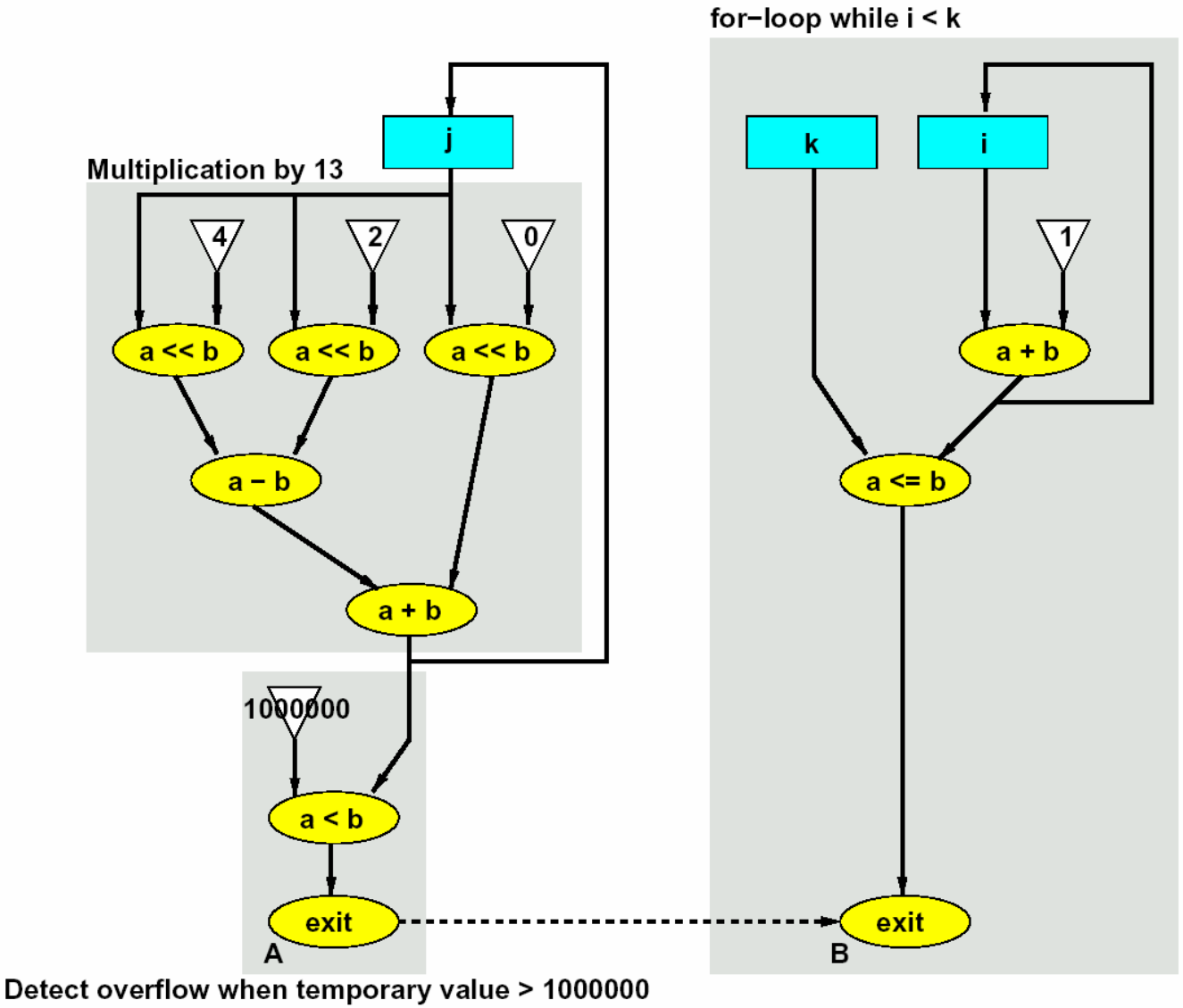
Hardware/Software-Ablauf



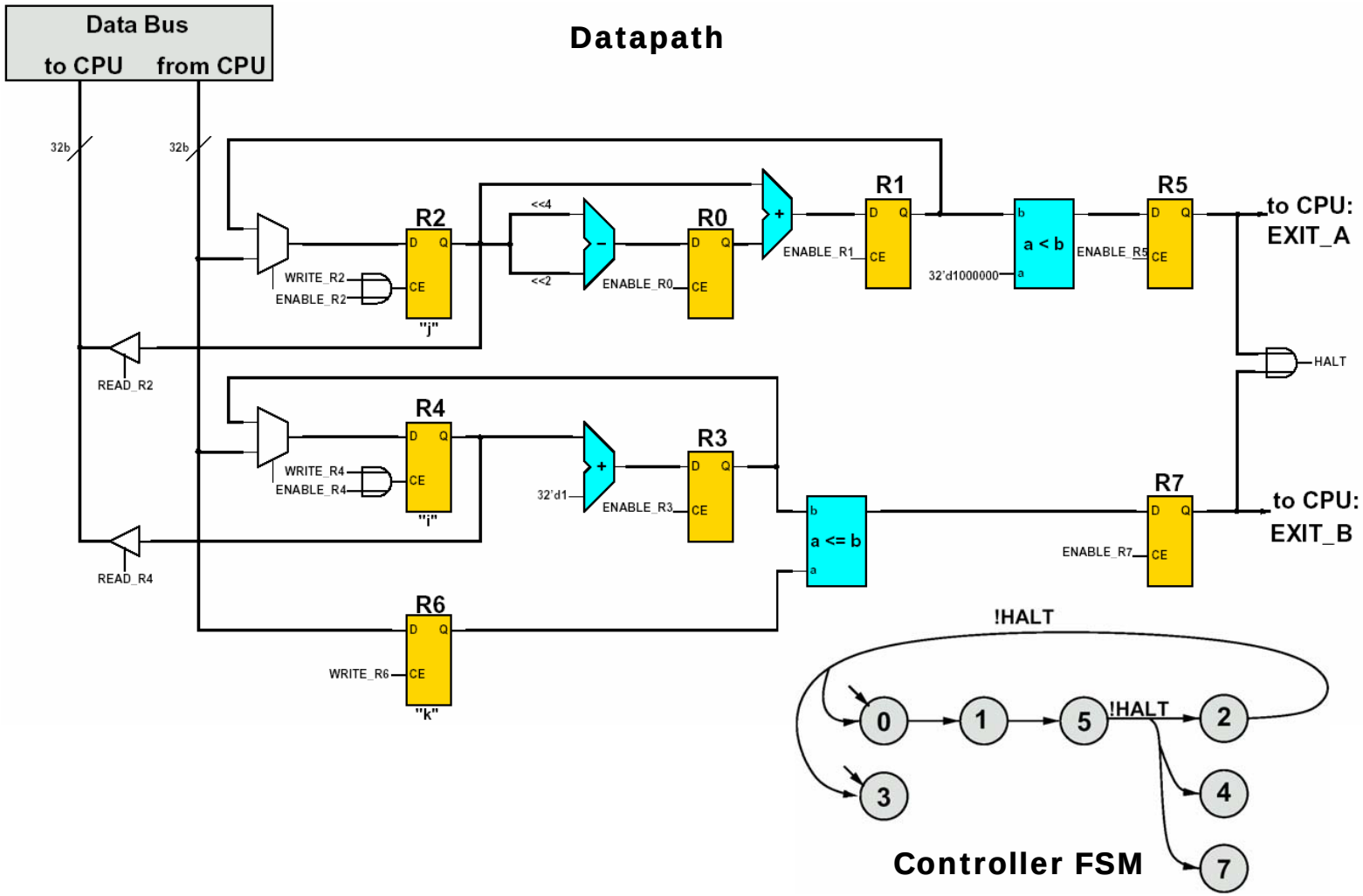
A. Koch

Kontroll-Datenflußgraph
CDFG

A. Koch



Erzeugte Hardware auf RCU



A. Koch

Hardware/Software-Schnittstelle

Embedded Systems & Applications

A. Koch

```
// Transfer software variables into RCU register
  rc[2] = j;
  rc[6] = k;
Loophead: // Destination jump label for restarting RCU after exception processing
  rc[4] = i;

  // Start RCU execution and wait for completion indicator (interrupt)
  rc[HW_START_REG] = 1;
  acev_wait();

  // OK, RCU execution stopped. Find out why ...
  if (rc[HW_EXIT_REG] == HW_EXIT_A) { // RCU indicated overflow of temporary value.

    // Fetch current values from RCU registers into software variables
    j = rc[2];
    i = rc[4];

    // Execute rest of this iteration in software
    printf("j=%d too large in loop i=%d\n", j, i);
    i = i + 1;

    // Now execute next iteration
    goto Loophead;
  } else /* HW_EXIT_B: RCU indicated normal exit */ {
    // Fetch final result from RCU register into corresponding variable
    j = rc[2];

    // Finish by executing remaining non-kernel instructions in software
    printf("result: j = %d\n", j);
  }
```

ACS-Programmiertricks

Grundsätzliche Tipps

Für hohe Rechenleistung

- Nicht einfach Software-Programm übersetzen
- An echte Hardware denken
  - Digitale Signalverarbeitung seit den 1950er Jahren
  - Ohne software-programmierbare Prozessoren
  - Alles in Spezial-Hardware realisiert
  - Viele der damaligen Algorithmen fast in Vergessenheit geraten

Beispiele

- Coordinate Rotation Digital Computer (CORDIC)
  - Berechne trigonometrische und transzendente Funktionen nur durch Shift/Add
- Länge eines Vektors (a, b)
  - Langsam: $d = \sqrt{a^2 + b^2}$
  - Falls 10% Abweichung akzeptabel ist: $d = \max\{a, b\} + 1/2 \cdot \min\{a, b\}$

Spezialisierte Zahlenformate

A. Koch

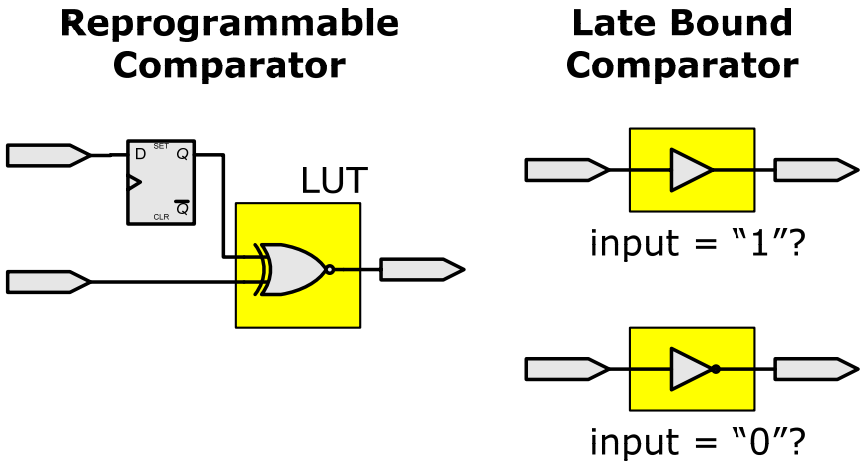
- Einfach: Passe Operatorbreiten an Datenbreiten an
  - Geht nur mit internen Daten
  - Schnittstellen haben i.d.R. feste Breite
  - Beispiel: 8b $*$ 12b = 20b, statt einem 32b Operator
- Mittel: Modifizierte Standardformate
  - Spezielle Fixkommaformate: 8b.4b
  - Spezielle Gleitkommaformate
    - Z.B. Höhere Genauigkeit, aber kleinerer Wertebereich
  - Passe an Anforderungen von unterschiedlichen Stellen im Algorithmus an
- Komplex: Nicht-Standarddarstellungen
  - Number Theoretic Transforms (NTT) schneller als FFT
  - Brauchen z.B. Einerkomplement (Mersenne NTT), Diminished-1 (Fermat NTT)

Partielle Evaluation

A. Koch

- Reduziere Hardware durch Propagieren von Konstanten
- Geschieht bei Erzeugung der Schaltung
  - HDL-Synthese
  - Parametrisierte Modulgeneratoren
- Sehr häufiger Einsatz: Multiplikation mit Konstanten
  - Siehe voriges Beispiel für HLL-Kompilierung
- Andere Anwendungen
  - Krypto-Schlüssel-spezifische RCUs

Late Binding
Eingeschränkte Form der partiellen Laufzeitrekonfiguration



A. Koch

- Verändere Funktion der Schaltung
- Aber erhalte ihre Struktur
- Platzierung und Verdrahtung bleiben gleich
- Nur der Inhalt der Logikblöcke wird geändert
- Kürzere Verzögerungen und flächeneffizienter als Programmierung
- Aber Änderungen sind langsamer (Rekonfiguration)

Zusammenfassung

ACS Zusammenfassung

- Systemarchitektur
- Terminologie
- Programmierverfahren
- Hochsprachenkompilierung
- Programmiertricks

➥Beim nächsten Mal

- Verilog
- Echtes Programmbeispiel
- Demo des Werkzeugflusses