



**TECHNISCHE
UNIVERSITÄT
DRESDEN**

Fakultät Elektrotechnik und Informationstechnik, Institut für Grundlagen der Elektrotechnik und Elektronik
Stiftungslehrstuhl Hochparallele VLSI-Systeme und Neuromikroelektronik

High Level Synthese von Datenpfaden mit optimierten Rekonfigurationskosten

Markus Rullmann, Renate Merker

München, 28. Mai 2008



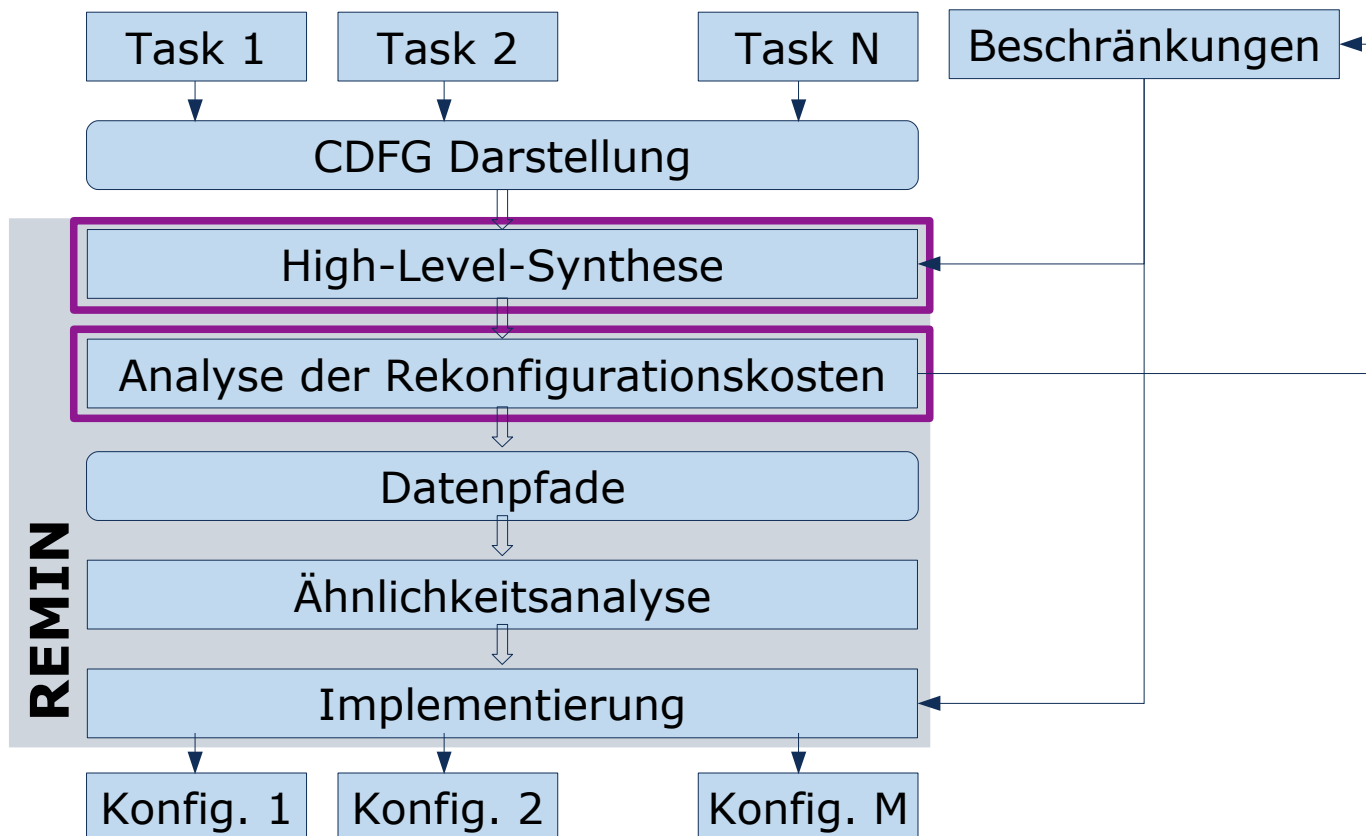
Gliederung

1. Einführung
2. Konfigurations-Übergangs-Modell
 - Bitebene
 - Strukturebene
3. High-Level-Synthese
 - Realisierungsmöglichkeiten der Rekonfiguration
 - Abbildungsvarianten bei der Bindung an Ressourcen
 - Diskussion am Beispiel
4. Zusammenfassung und Ausblick

Motivation

- Analyse und Implementierung effizient rekonfigurierbarer Schaltungen
 - Partitionierung in statische und dynamische Teile
 - Optimierung der Schaltung für Rekonfiguration
 - Kosten für Rekonfiguration, Ressourcen, Zeitverhalten
- Syntheseverfahren für rekonfigurierbare Schaltungen
- Rekonfigurationskosten auf allen Entwurfsebenen
 - Modelle für Konfigurations**daten** und Rekonfigurations**zeit**
 - Binäre Konfigurationsdaten
 - Statische Teile der Konfiguration
 - Strukturbasierte Beschreibungen
 - Netzlisten: wiederverwendete Logic und Verbindungen
 - CDFG: wiederverwendete Macros und Verbindungen

Einordnung in den Entwurfsfluß



Konfigurations-Übergangs-Modell

1. Einführung

2. Konfigurations-Übergangs-Modell

- **Bitebene**
- **Strukturebene**

3. High-Level-Synthese

- Realisierungsmöglichkeiten der Rekonfiguration
- Abbildungsvarianten bei der Bindung an Ressourcen
- Diskussion am Beispiel

4. Zusammenfassung und Ausblick

Konfigurations-Übergangs-Modell

Graphen-basiertes Modell zur Rekonfiguration von Hardware

- Knoten: Konfigurations-Zustand
 - Attribut K : Konfiguration
- Kante: Konfigurations-Übergang
 - Attribut ΔK : Unterschied in der Konfiguration zwischen Ausgangs- und Zielknoten



Analyse:

- Rekonfigurations**zeit**: Kostenfunktion von ΔK
- Konfigurations**daten**: Kostenfunktion von $\bigcup_{\text{Ank. Kanten}} \Delta K$

Konfigurations-Übergangs-Modell für Konfigurationsdaten

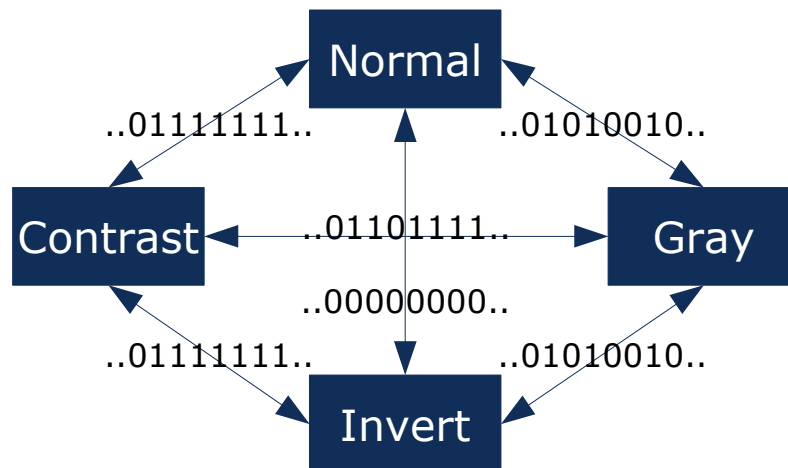


- Knoten: Konfigurations-Zustand
 - Attribut K : binäre Konfiguration des Moduls, gruppiert zu Frames
- Kanten: Konfigurations-Übergang
 - Attribut ΔK : markiert unterschiedliche Konfigurations-Frames
- Bitstreams realisieren Konfigurations-Übergänge
 - Mehrere Kanten können durch den gleichen Bitstream realisiert werden:
 - Bitstream aus $U \Delta K$ für mehrere Kanten
- Wie kann eine optimale Menge von Bitstreams ausgewählt werden?
 - Rekonfigurationszeit
 - Konfigurationsdaten
 - Lösung durch Ganzzahlige Lineare Optimierung

Beispiel: ESM Video Filter Tutorial

- Optimierungstool

- Rekonfigurationszeit, Konfigurationsdaten
- Direkter Übergang oder Abbildung auf Übergangssequenz



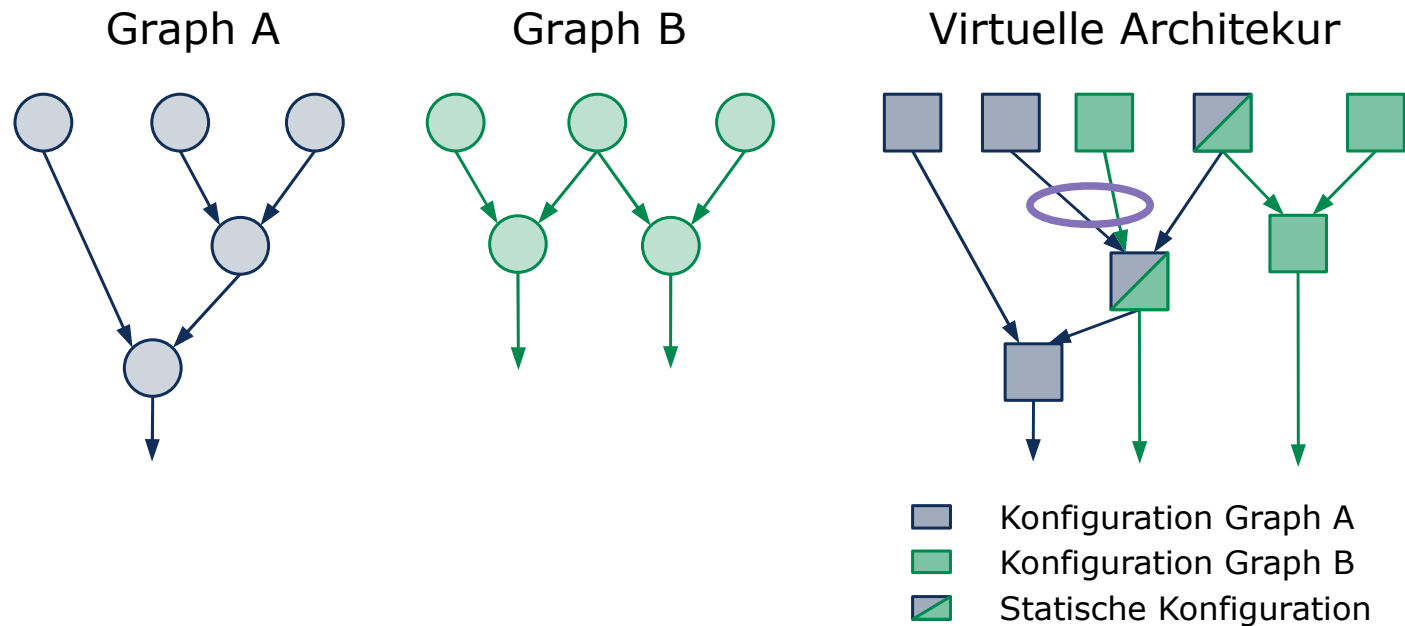
Bsp.: Zielkonfiguration *Contrast*

- Bitstream 1: „..01111111..“
Ausgangskonf.: *Normal, Invert*
- Bitstream 2: „..01101111..“
Ausgangskonf.: *Gray*

Methode	\bar{t} [Fr]	\bar{s} [Fr]
Direct Transition, Min. Size	61.00	61.00
Direct Transition, Min. Time	44.83	108.50
Transition Sequence, Min. Size	63.66	41.25
CombitGen	61.00	61.00
Xilinx	88.00	88.00

Virtuelle Architektur

- Knoten: verfügbare oder benötigte Ressourcen aller Eingabe-Graphen
- Kanten: Resultat der Abbildung aller Graphen



Konfigurations-Übergangs-Modell für Netzlisten-basierte Entwurfsebenen

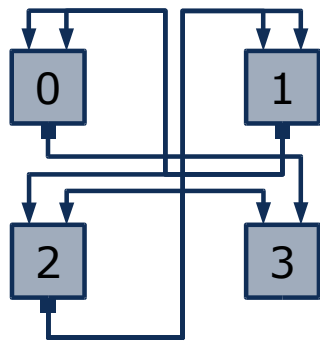
- Beschreibung der Module als Netzlisten
- Abbildung der Module auf eine virtuelle Architektur (VA)
- Knoten: Konfigurations-Zustand
 - Attribut K : Belegung und Konfiguration von Elementen der VA
- Kanten: Konfigurations-Übergang
 - Attribut ΔK : Änderung in Belegung und Konfiguration von Elementen der VA

Verschiedene Abbildungen auf die virtuelle Architektur ergeben andere Rekonfigurationskosten

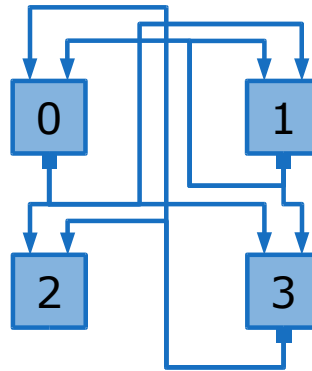
- Bewertung von Rekonfigurationszeit und Rekonfigurationsdaten anhand der Belegung der VA durch die Module
- Ziel: Identifikation von optimalen Abbildungen

Virtuelle Architektur zur Bewertung von Rekonfigurationskosten

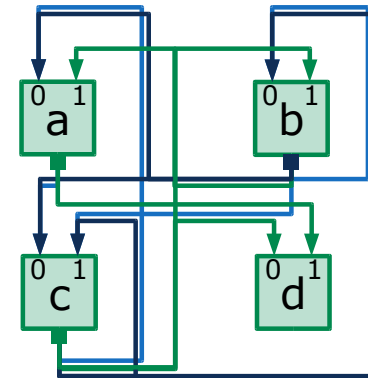
Graph A



Graph B



Virtuelle Architektur



Graph	Logik	Verbindungen											
VA	a b c d	b-a0	b-a1	b-c0	a-d1	c-b0	c-b1	c-c1	c-d0	a-c0	c-a0	b-c1	b-b0
A	0 1 2 3	1	1	1	1	1	1	1	1	0	0	0	0
B	3 1 0 2	0	1	0	1	0	1	0	1	1	1	1	1

Kostenfunktionen für eine virtuelle Architektur

Rekonfigurationsdaten

- Anzahl verschieden konfigurierter Elemente der Architektur
 - Bsp.: VA gesamt: 12 Verbindungen:
 - nur Graph A(4) + nur Graph B(4) + Graph A,B (4)
 - Dynamisch: 8 Verb., Statisch 4 Verb.

Rekonfigurationszeit

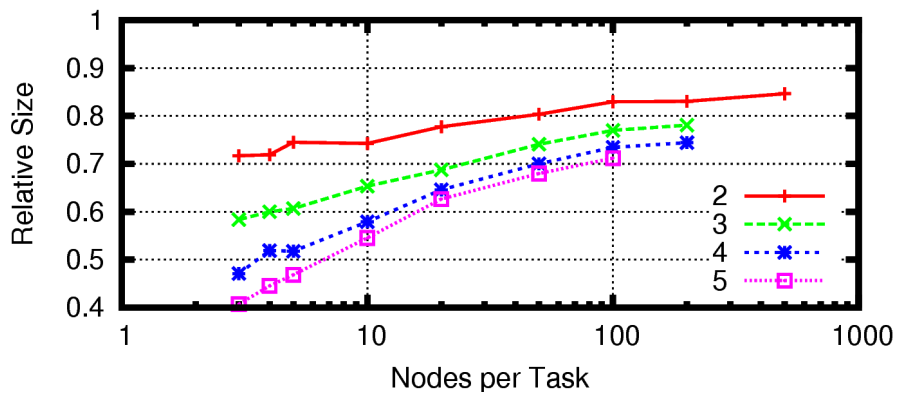
- Äquivalent zur Anzahl rekonfigurierter Elemente
 - Bsp.: Rekonfiguration der Verbindungen beim Übergang von Graph A zu B

	Ohne De-Konfiguration	Mit De-Konfiguration
Statisch	4	4
Rekonfiguration	4	8
Ohne VA	8	16

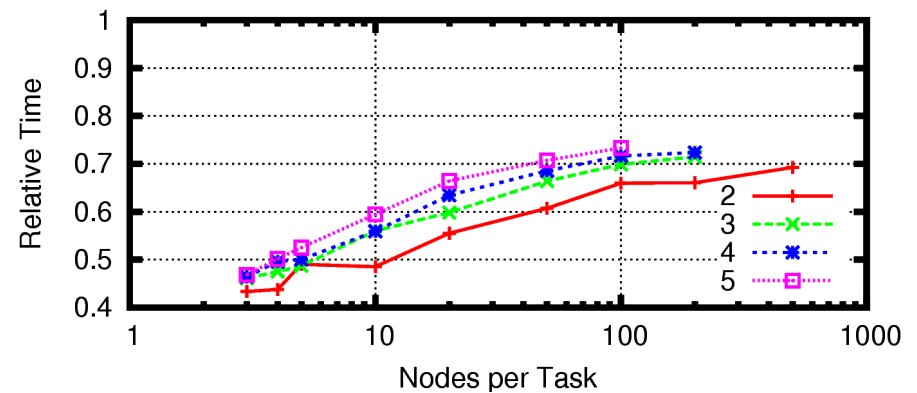
Abbildung mit reduzierten Rekonfigurationskosten

- Analyse der Rekonfigurationskosten für zufällige Graphen mit 2 Eingängen und 1 Ausgang pro Knoten
- Abbildung auf die virtuelle Architektur optimiert mit sim. Annealing

Relative Bitstream Size for Connectivity



Relative Reconfiguration Time for Connectivity



Vergleich des Modells

Binärdaten

- Konkrete Daten für
 - Konfigurationsdaten
 - Erw. Rekonfigurationszeit
- Optimierungspotential
 - Bitstream-Auswahl
 - Bitstream-Kompression

Strukturdaten

- Modell für
 - Konfigurationsdaten
 - Erw. Rekonfigurationszeit
- Optimierungspotential
 - Abbildungsvarianten bei:
 - **Synthese**
 - Mapping
 - Platzierung
 - Routing

High-Level-Synthese

1. Einführung
2. Konfigurations-Übergangs-Modell
 - Bitebene
 - Strukturebene
- 3. High-Level-Synthese**
 - **Realisierungsmöglichkeiten der Rekonfiguration**
 - Abbildungsvarianten bei der Bindung an Ressourcen
 - Diskussion am Beispiel
4. Zusammenfassung und Ausblick

High-Level-Synthese

Rekonfigurationkosten-Optimierte Implementierung von Tasks

- Freiheitsgrade bei Bindung/Scheduling
- Zuordnung von Tasks zu Konfigurationen
 - Control-basierte Rekonfiguration und HW Rekonfiguration

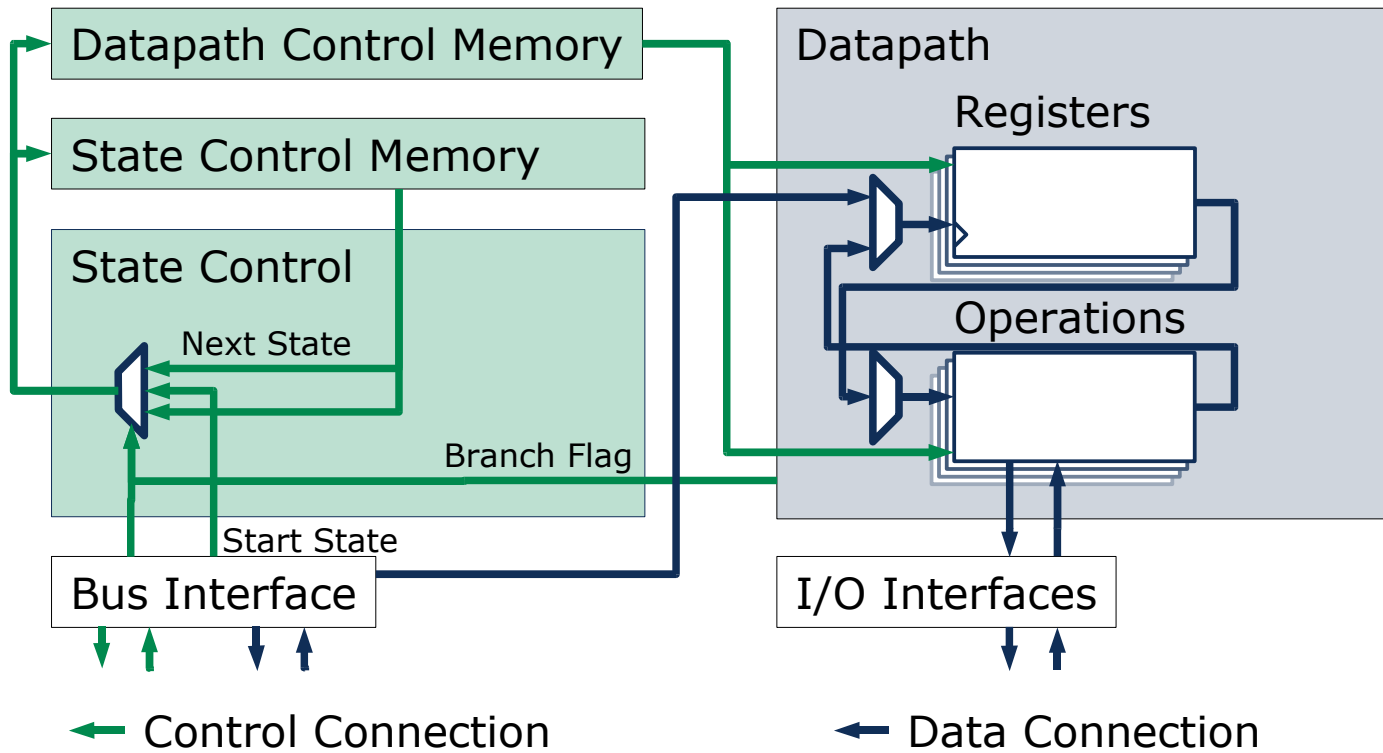
Vorteile

- Höheres Abstraktionsniveau
- Effektivere Verarbeitung auf höherer Ebene:
 - Virtuelle Architektur: Operationen (Macros), Verbindungen (Busse)
- Partitionierung der Implementierung
 - Trennung in Controller und Datenpfad

Rekonfiguration auf zwei Ebenen

- 1. Ebene (RC1) Konfiguration für die Architektur
 - Laden über Konfigurationsschnittstelle
 - Rekonfiguration in 10-100-... Taktzyklen
- 2. Ebene (RC2) Steuerung der Verarbeitung
 - Taktbasierte Steuerung von Verbindungen und Funktion der Funktionseinheiten
- Mehr Flexibilität im Datenpfad -> Höherer Ressourcenbedarf für RC2 Steuerung
- Zugeschnittener Datenpfad -> Häufige Rekonfiguration in RC1

Architekturtemplate, Partitionierung



Architekturtemplate, Rekonfigurations-Möglichkeiten

Hybride Partielle Rekonfiguration von **Teilen** der Module

State Control Memories:

- RC1: Rekonfiguration von Speicher-Layout und -Inhalt
- RC2: Adaption des Speicherinhaltes
 - Caching von Steuerbefehlen- und Sequenzen

Datenpfad:

- RC1: Rekonfiguration des Datenpfades
 - Wiederverwendung von Operationen und Verbindungen
- RC2: Steuerung des Datenpfades
 - während der Task-Verarbeitung
 - bei Task-Wechsel

High-Level-Synthese

1. Einführung
2. Konfigurations-Übergangs-Modell
 - Bitebene
 - Strukturebene
- 3. High-Level-Synthese**
 - Realisierungsmöglichkeiten der Rekonfiguration
 - **Abbildungsvarianten bei der Bindung an Ressourcen**
 - Diskussion am Beispiel
4. Zusammenfassung und Ausblick

Bindung an Ressourcen

Szenarios

Verschiedene Strategien für die Ressourcenbindung, u.a.:

- Synthese von mehreren Tasks in eine Konfiguration
- Synthese von Tasks in separate Konfigurationen

Bewertung der Lösungen nach Kosten

- Ressourcenbedarf des Datenpfades
- Wiederverwendung von Ressourcen
- Ressourcenbedarf für die Steuerinformationen
- Zeitverhalten

Abbildungseffekte bei der Bindung an Ressourcentypen

Argument/Port Mapping



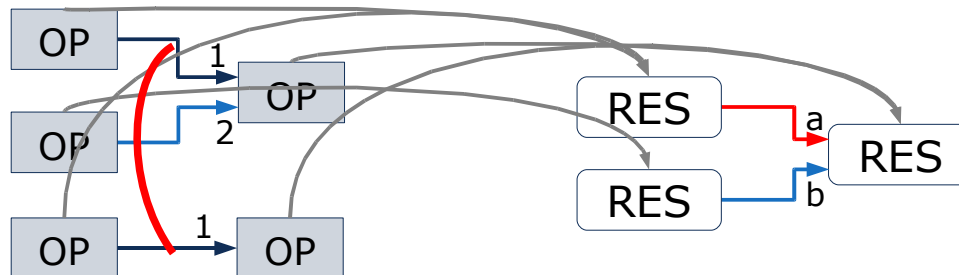
Kommutative Bindung



Bindungsvielfalt



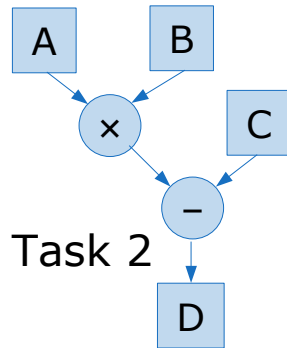
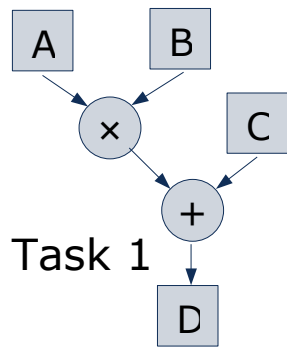
Potentiell wiederverwendbare Verbindungen



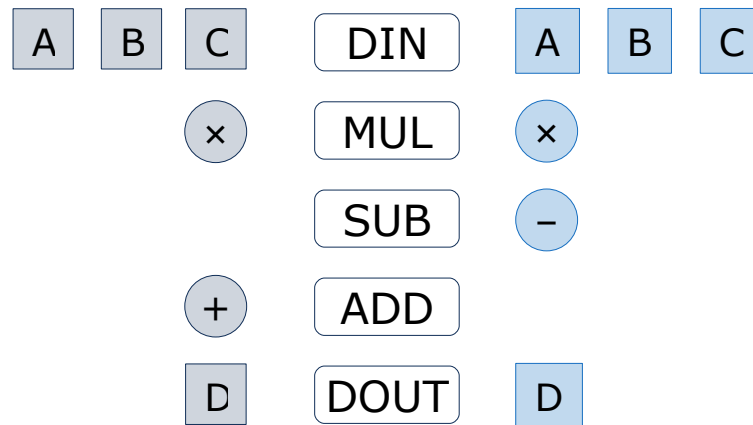
Realisierte wiederverwendete Verbindungen ergeben sich erst aus der Bindung an Ressourceninstanzen!

Strategien zur Bindung an Ressourcentypen

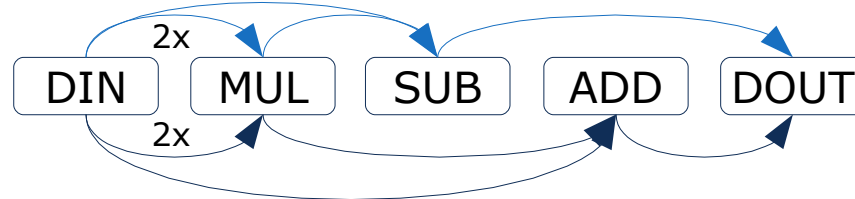
Unabhängige Bindung



Bindung

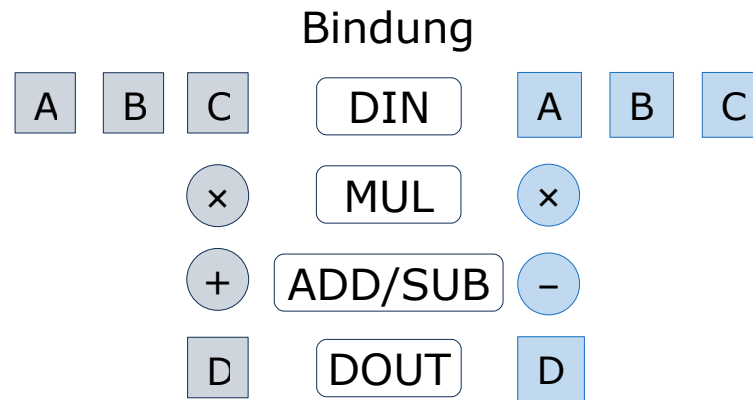
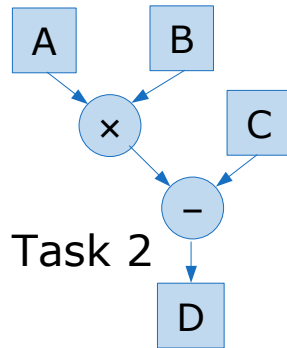
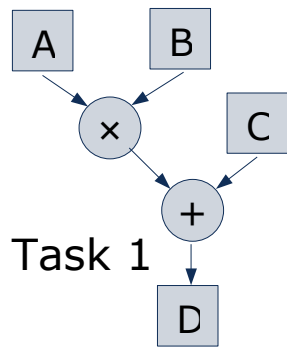


Virtuelle Architektur

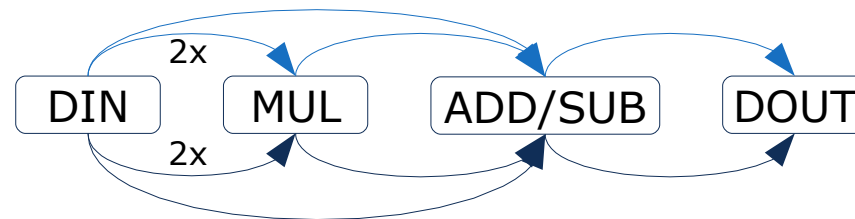


Strategien zur Bindung an Ressourcentypen

Gemeinsame Bindung



Virtuelle Architektur



Bindung wird bewertet nach potentiell wiederverwendeten Verbindungen

Gemeinsame Bindung an Ressourcentypen: Varianten

Bindung aller Tasks in einem Schritt

1. Minimale Vielfalt an Ressourcentypen

- Verwendung möglichst weniger verschiedener Ressourcentypen bei der Bindung
- Größerer Freiheitsgrad bei der Bindung an Ressourceninstanzen

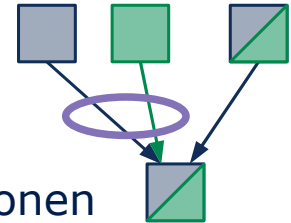
2. Minimale Anzahl verschiedenartiger Verbindungen

- Abbildung Datenabhängigkeiten auf wenige Arten von Verbindungen (Quell-Modul, Ausgabeport zu Ziel-Modul, Eingabeport)
- Ermöglicht Wiederverwendung gleicher Verbindungen bei der Bindung an Ressourceninstanzen

Effektive Bindung der Operationen innerhalb von Tasks und für die Tasks insgesamt

Bindung an Ressourceninstanzen

- Verfügbare Instanzen bilden die Knotenmenge der VA
- Bindung ist beschränkt durch Schedule
- Bindungsvarianten führen zu verschiedener Belegung der VA
 - wiederverwendeten Ressourcen
- Verbindungen in der VA ergeben sich aus der Bindung:
 - benötigten Steuer-Multiplexer in einer Konfiguration
 - wiederverwendete Verbindungen
- Bewertung versch. Zuordnungen von Tasks zu Konfigurationen



Anwendung des Rekonfigurations-Kostenmodells für Strukturdaten zur Optimierung der Instanzbindung

Zusammenfassung der Syntheseschritte

Bindung an Ressourcentypen

- Neuzuordnung von Ein- oder Ausgabeports
- Vertauschbare Ein- und Ausgabeports

Scheduling

- Einfügen von temporären Registern in Datentransfers
- Einschränkungen für die Bindung an Instanzen

Bindung an Ressourceninstanzen

- Abbildung von Datentransfers auf Datenverbindungen
- Mehrfachnutzung von Datenverbindungen durch verschiedene Datentransfers.
- Steuerung der Datentransfers durch Multiplexer

High-Level-Synthese

1. Einführung

2. Konfigurations-Übergangs-Modell

- Bitebene
- Strukturebene

3. High-Level-Synthese

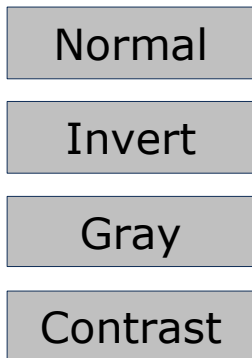
- Realisierungsmöglichkeiten der Rekonfiguration
- Abbildungsvarianten bei der Bindung an Ressourcen
- **Diskussion am Beispiel**

4. Zusammenfassung und Ausblick

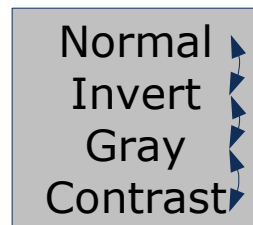
Beispiel: Rekonfigurierbarer Videofilter

- Tasks: Normal, Invert, Gray, Contrast
- HW Konfigurationen:

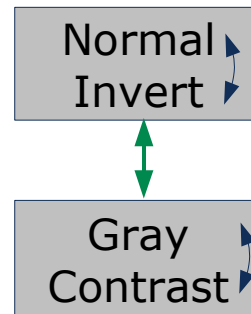
1 Konfiguration
(no reuse)



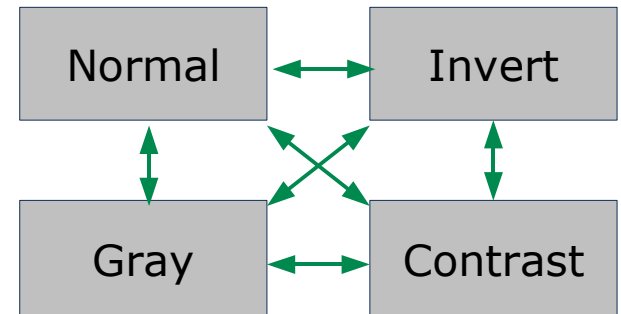
1 Konfiguration



2 Konfigurationen



4 Konfigurationen

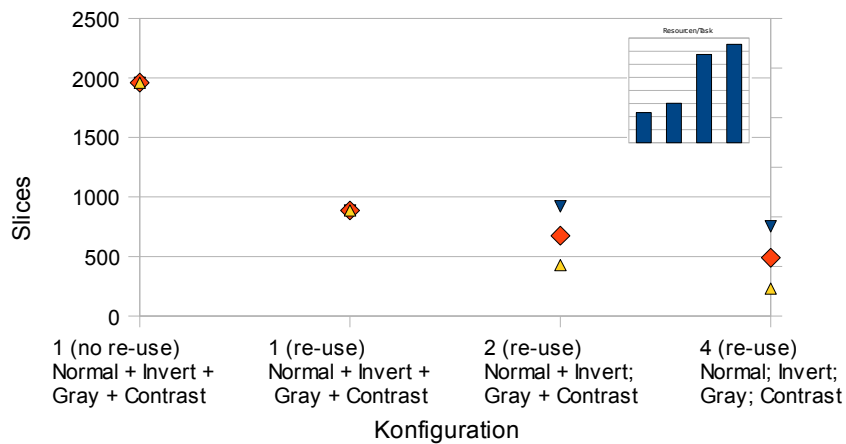


↕ RC1 Rekonfiguraton ↪ RC2 Rekonfiguraton

Methoden im Vergleich: Ressourcenbelegung

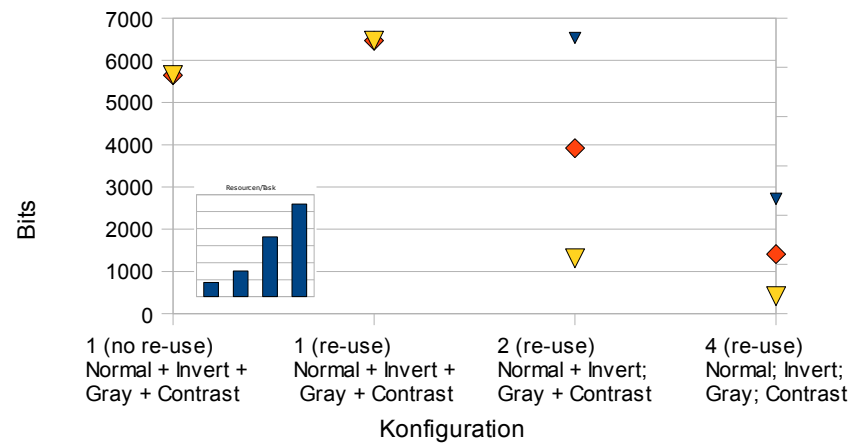
Ressourcenbelegung für die Konfigurationen

Datenpfad

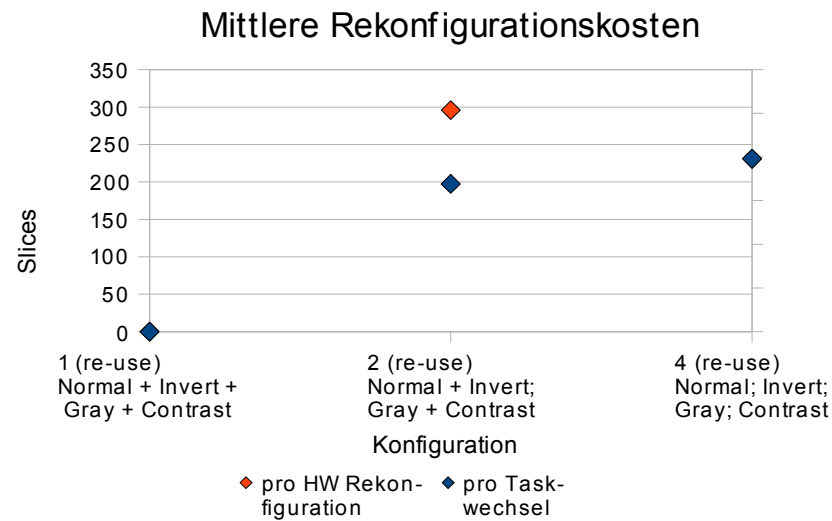


Ressourcenbelegung für die Konfigurationen

Control Memory



Methoden im Vergleich: Rekonfigurationskosten



Zusammenfassung und Ausblick

1. Einführung

2. Konfigurations-Übergangs-Modell

- Bitebene
- Strukturebene

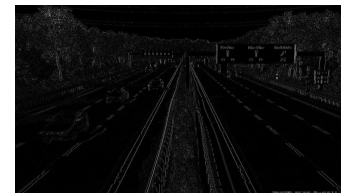
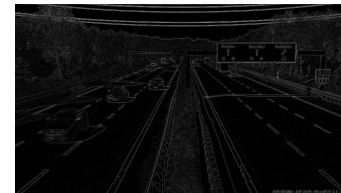
3. High-Level-Synthese

- Realisierungsmöglichkeiten der Rekonfiguration
- Abbildungsvarianten bei der Bindung an Ressourcen
- Diskussion am Beispiel

4. Zusammenfassung und Ausblick

ESM Demonstrator

- Anwendungs-Demonstration des High-Level-Synthese-Tools
 - Echtzeit-Videoverarbeitung:
VGA Auflösung (640x480, 25fps)
 - Eine Konfiguration, 3 Tasks:
 - nur Video Ausgabe
 - Sobel Filter 2D
 - Sobel Filter Horizontal
 - Tasks verarbeiten autonom:
 - FIFO Kontrolle
 - Synchronisation
 - Filter-Kerne
 - Partner: TUM, U. Erlangen



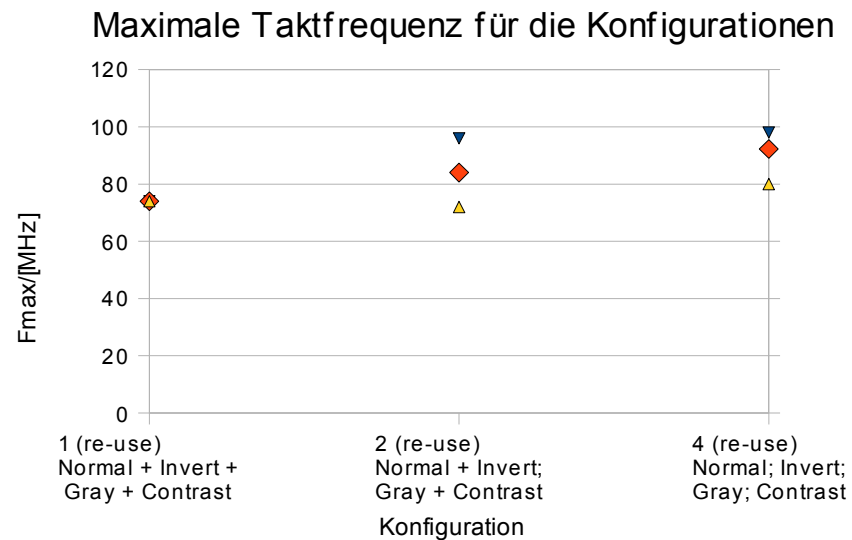
Kooperationen

- TU München, PD Dr. Walter Stechele
 - Engines als Beispiele für die High-Level-Synthese
- Universität Paderborn, Prof. Dr. Marco Platzner (ReConOS) und Universität Erlangen-Nürnberg, Prof. Dr. Jürgen Teich (ESM)
 - FPL Demonstrator: „Multithreaded Design of Reconfigurable Applications using High-Level Synthesis“

Zusammenfassung

- Konfigurations-Übergangsmodell zur Bewertung von Rekonfigurationskosten
- Auswirkungen der Bindung auf die Rekonfigurationskosten
- High-Level-Synthese-Tool zur Untersuchung von Abbildungs- und Implementierungsvarianten
- Optimierung der Wiederverwendung von Ressourcen und Verbindungen für die Rekonfiguration auf Steuerungs- und Architekturebene

Methoden im Vergleich: Taktfrequenz



High Level Synthese

Übersicht

- Spec: ANSI-C Subset oder CDFG + HW Library
- HLS mehrerer Tasks
- Zuordnung zu Modulen/Konfigurationen
- Output: vhdl, c-code interface
- **Problem: Transformation des CDFG zum Architektur-graphen, Ähnlichkeit beider**
- HW Library:
 - Macros für HW-Access (FIFO I/O etc)
 - erlaubt HW-spezifische Konstrukte im C-Code

Bindung an Ressourcen

Trade-off: Rekonfigurationskosten (RC1) und Ressourcenbedarf (RC2)

- Gemeinsam genutzte **Verbindungen**
 - weniger Steuer-Multiplexer
- Gemeinsam genutzte **Ressourcen**
 - mehr Steuer-Multiplexer, um alle Verbindungen zu realisieren
- Komplexere Ressourcentypen
 - mehr Logik
 - mehr Steuer-Information

Implementierungsvarianten

Bindung mit dem VA Modell erlaubt die Bewertung verschiedener Implementierungsvarianten

- Zuordnung von Tasks zu Konfigurationen
 - 1. Ebene (RC1) Taskwechsel durch HW-Rekonfiguration
 - Wiederverwendung von Ressourcen zwischen Konfigurationen
 - Rekonfigurationkosten in Slices/Verbindungen
 - 2. Ebene (RC2) Taskwechsel durch Steuerung der Verarbeitung
 - Wiederverwendung von Ressourcen in der Konfiguration
 - Kosten für zusätzliche Steuerlogik zur Rekonfiguration