



A. Koch

Eingebettete Prozessorarchitekturen

4. Compilierung und Beispiele für VLIW-Prozessoren

Andreas Koch

FG Eingebettete Systeme und ihre Anwendungen
Informatik, TU Darmstadt

Wintersemester 2007/2008

Quellenangabe



A. Koch

Alle Zeichnungen in diesem Abschnitt aus Kapitel 4

[Bulldog: A Compiler for VLIW Architectures](#)

von

[John Ellis](#)

Dort auch weiterführende Erklärungen!

Vorgehensweise



A. Koch

- CFG erstellen
 - Hier: Jeder Basisblock enthält nur eine Operation
- Profiling
- Spuren im CFG bilden
- Jede Spur einzeln bearbeiten
- Ablaufplan erstellen
- Spur in CFG durch ihren Ablaufplan ersetzen
- Kompensationscode in CFG einfügen

Kompensationscode

Definitionen



Split

Bedingter Sprung aus der Spur **heraus**

Join

Sprung von aussen in die Spur **hinein**

On-Trace-Edge

Auf der Spur weiterverlaufender Ast eines Splits

Off-Trace-Edge

Von der Spur **abweichender** Ast eines Splits

A. Koch

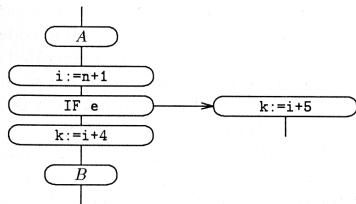
Splits

Ausgangssituation - Eine Operation verschoben



A. Koch

Eingabe-Spur



Ablaufplan

1	IF e1
2	i:=n+1
3	k:=i+4

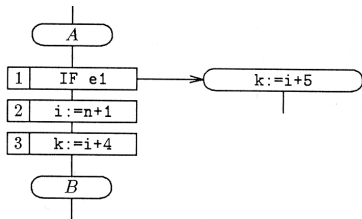
Annahme: Alle Operationen brauchen einen Takt

Splits

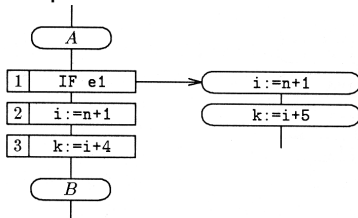
Eine verschobene Operation kompensieren



Einbauen des Ablaufplans in
CFG



Kompensationscode



Eingefügt an Anfang von
Off-Trace-Edge.

Fehler: $i := n+1$ kommt nicht
bei k an

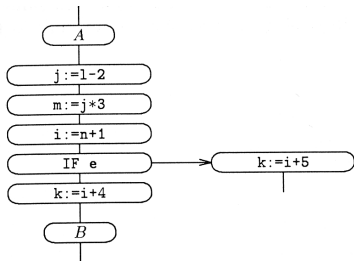
Splits

Ausgangssituation - Mehrere Operationen verschoben



A. Koch

Eingabe-Spur



Ablaufplan

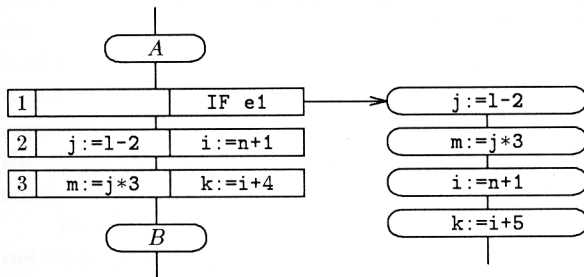
1		IF e1
2	j:=1-2	i:=n+1
3	m:=j*3	k:=i+4

Splits

Mehrere verschobene Operation kompensieren



A. Koch



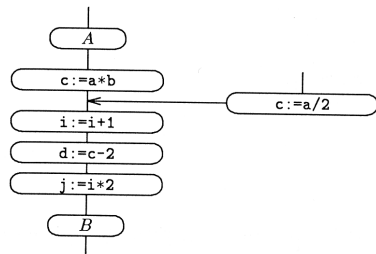
- Kopiere alle verschobenen Operationen in Off-Trace-Edge
- In der Auftretensreihenfolge aus Original-Spur

Joins

Ausgangssituation



Eingabe-Spur



Ablaufplan

1	$c := a * b$	$i := i + 1$
2	$d := c - 2$	$j := i * 2$

A. Koch

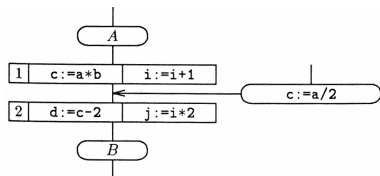
- Wie Ablaufplan in CFG einbauen?
- Join vor Takt 1?
 - $c := a * b$ überschreibt $c := a / 2$
- Join vor Takt 2?
 - Rechnet nicht mehr $i := i + 1$
- Nach Takt 2?
 - Berechnungen für i, d, j fehlen

Joins

Kompensationscode

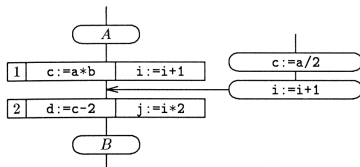


Wahl des Einsprungpunktes



- So früh wie möglich, aber:
- Keine Operation, die vorher **vor** Join lag
- ... darf nun dahinter liegen

Mit Kompensationscode



- Einfügen **in** Join-Kante
- Auch mit mehreren Instruktionen machbar
- Alle in Originalreihenfolge einkopieren

A. Koch

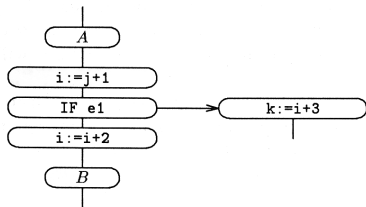
Verschieben vor Splits?

Ausgangssituation



A. Koch

Eingabe-Spur



Ablaufplan

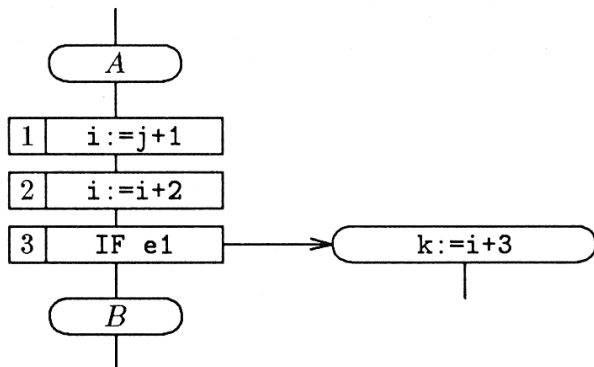
1	$i := j + 1$
2	$i := i + 2$
3	IF e1

Verschieben vor Splits?

Problem



A. Koch



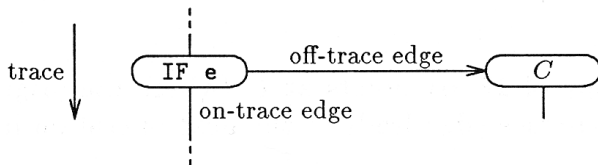
- $k:=i+3$ bekommt nun falschen Wert von i
- Was tun? Daten sind kaputt (i überschrieben)!
- Kann so **nicht** bei Ablaufplanung auftreten
 - i wird auf Off-Trace-Edge gelesen, i ist dort **live**
 - Kontrollkante **IF** $e1 \rightarrow i:=i+2$ einziehen
 - Anweisung $i:=i+2$ wird **nicht** vor **IF** $e1$ bewegt

Verschieben von Sprüngen



Wesentliche Unterscheidung zwischen
On-Trace und **Off-Trace**

A. Koch



- Sprungrichtung (wahr/falsch) häufig irrelevant
- Aber Ziel von Trace-Scheduling:
 - Vermeide **Nehmen** von Sprüngen innerhalb Spur
 - Warum?
 - Erinnerung: Zeitverschwendung bei Pipelining

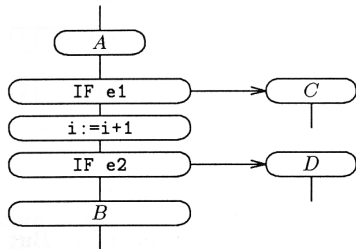
Verschieben von Sprüngen über Splits

Ausgangssituation



A. Koch

Eingabe-Spur



Ablaufplan

1	IF e2
2	IF e1
3	i:=i+1

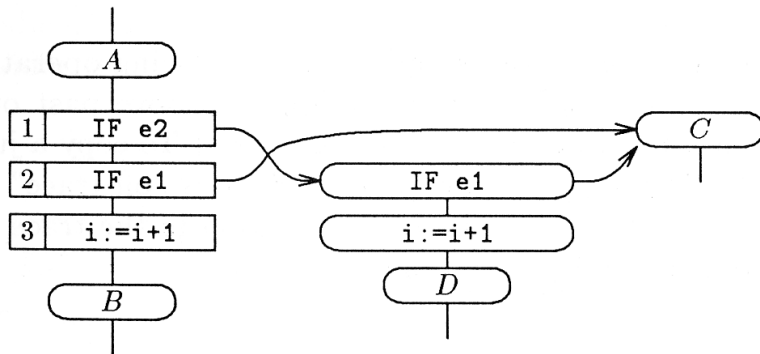
IF e2 jetzt am Anfang

Verschieben von Sprüngen über Splits

Kompensationscode



A. Koch



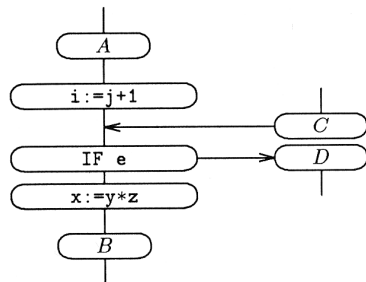
- Gleiches Verhalten
- Sequenzen wie **A**, **IF e2**, **IF e1**, **C** stören nicht
- Programmverhalten bleibt gleich

Verschieben von Sprüngen über Joins

Ausgangssituation



Eingabe-Spur



Ablaufplan

1	IF e
2	x:=y*z
3	i:=j+1

A. Koch

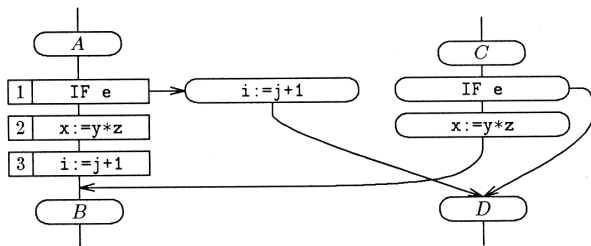
- **IF e** jetzt am Anfang, über Join hinweg bewegt
- Frühester Join **nach** Takt 3
- Sonst würde nach C fehlerhafterweise $i:=j+1$ gerechnet

Verschieben von Sprüngen über Joins

Kompensationscode



A. Koch



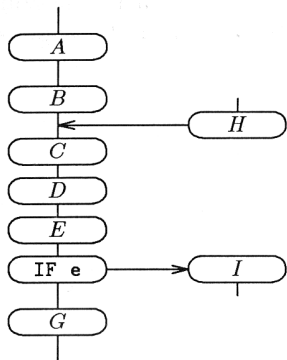
- $i:=j+1$ unter $\text{IF } e$ bewegt, muß aber **vor** D ausgeführt werden
 - Also in Off-Trace-Edge einkopieren
- $\text{IF } e$ und $x:=y*z$ nun über Join
 - Also in Join-Kante einkopieren

Kompliziertere Split/Join-Szenarios

Ausgangssituation



Eingabe-Spur



Ablaufplan

1	<i>D</i>	IF e
2	<i>B</i>	
3	<i>C</i>	<i>E</i>

A. Koch

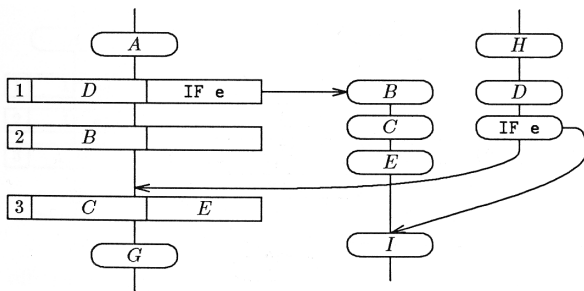
- Join möglich zwischen Takt 2 und 3
- Vorher würde fehlerhafterweise **B** ausgeführt

Kompliziertere Split/Join-Szenarios

Kompensationscode: 1. Versuch



A. Koch



- **B**, **C** und **E** nun unter **IF e**
 - Also in Off-Trace-Edge einfügen
- **D** und **IF e** nun über Join bewegt
 - Also in Join-Kante einkopieren

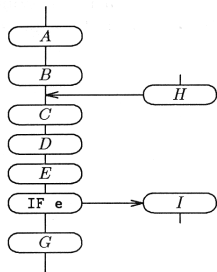
Kompliziertere Split/Join-Szenarios

Annahme: Eintritt bei **H** und **IF e** wahr

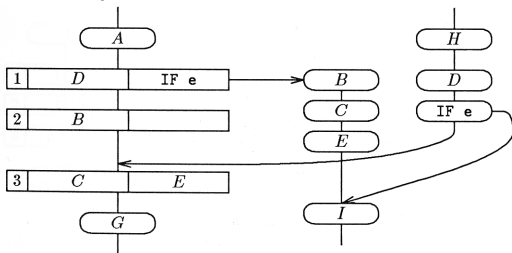


A. Koch

Original



Mit Kompensationscode



Fehler: Anderer Ablauf!

Statt H,C,D,E, **IF e**, I **nun fehlerhafterweise** H,D, **IF e**, I

Kompliziertere Split/Join-Szenarios

Regel für das Kopieren von Sprüngen in Join-Kanten



A. Koch

Alle Operationen der Spur, die zwischen dem ursprünglichen Join und dem bewegten Sprung (\mathbf{IF}) lagen, die selbst **noch nicht** in die Join-Kante einkopiert wurden, **müssen** in den Off-Trace-Edge des in die Join-Kante kopierten Sprunges eingefügt werden.

Kompliziertere Split/Join-Szenarios

Diskussion



A. Koch

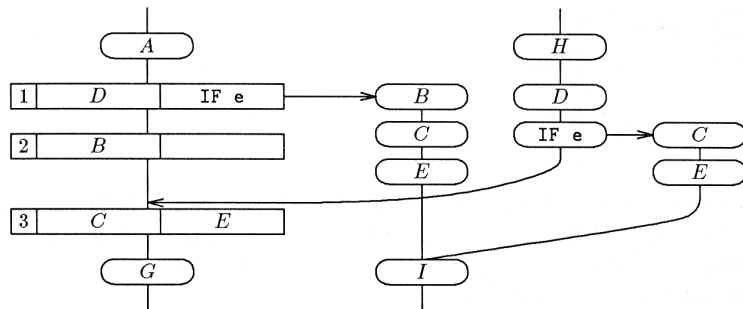
- Was für Anweisungen sind das?
- Solche, die im Ablaufplan **nach** dem Join liegen
- Und damit nicht selbst in die Join-Kante einkopiert wurden (“... die werden ja 'eh noch ausgeführt”)
- Diese werden bei Kopieren eines Sprungs in die Join-Kante
- ... auf dessen Off-Trace-Kante dann **nicht** mehr ausgeführt

Kompliziertere Split/Join-Szenarios

Korrektter Kompensationscode



A. Koch



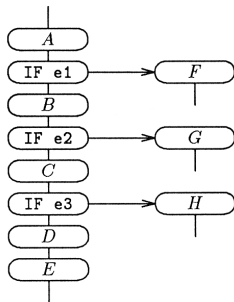
- c und E wurden in 1. Versuch nicht kopiert
- Join-Punkt liegt ja **davor**
- Kopie wäre also unnötig ...
- Nun wird aber **IF e** in Join-Kante kopiert
 - Bei **e=true**: Keine Rückkehr mehr in Spur, gleich Sprung nach **I**
- Nun fehlen c und E vor **I** → **Einkopieren!**

Parallele bedingte Sprünge

Ausgangssituation



A. Koch



1	A			
2	E	IF e1	IF e2	IF e3
3	B	C	D	

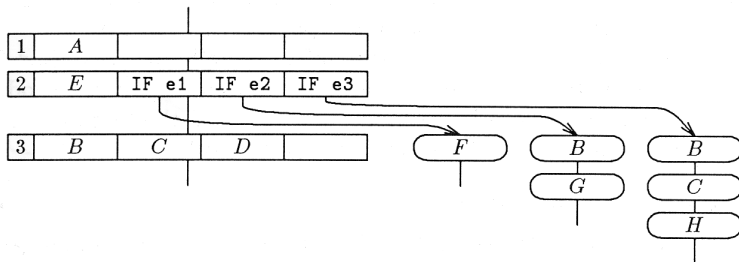
- Annahme: Bedingungen werden vorher berechnet

Parallele bedingte Sprünge

Kompensationscode



A. Koch



- **B** in Ablaufplan **unter IF e2** bewegt
 - Also in Off-Trace-Edge einkopieren
- Analog **B** und **c** für **IF e3**

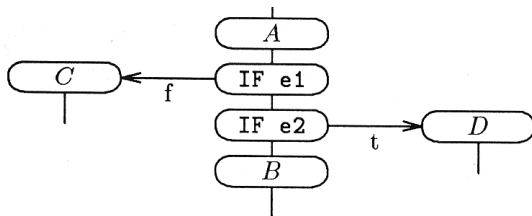
Sonderfall: Genommener Sprung auf Spur

Ausgangssituation



- Bisher Spur verläuft durch nichtgenommene Sprünge
 - **f**=fallthrough, **t**=taken (**nicht** false/true!)
- Nun Betrachtung des anderen Falls

A. Koch



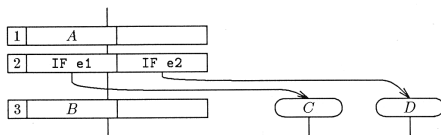
1	A	
2	IF e1	IF e2
3	B	

Sonderfall: Genommener Sprung auf Spur

Auswirkung und Lösung



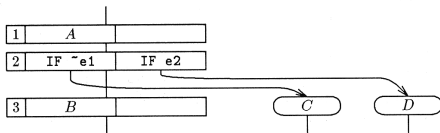
In CFG eingebauter Ablaufplan



A. Koch

Fehler: Nun ist **IF e2** Fallthrough von **IF e1**

Korrektur



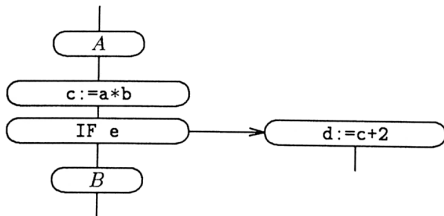
Bedingung **e1** invertieren, jetzt ist **IF e2** wieder Fallthrough

Handhabung von Mehrtakt-Operationen

Annahme hier: Multiplikation braucht 3 Takte



Spur



A. Koch

Ablaufplan

1	$c := a * b$	IF e
2	*2	
3	*3	

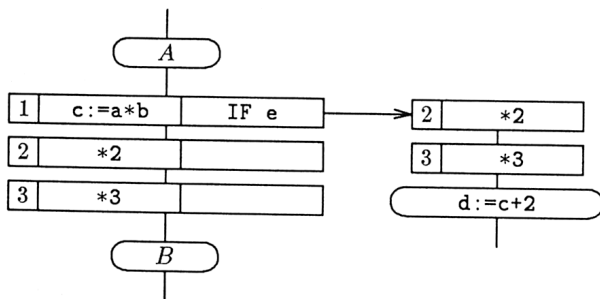
- *2 und *3 sind zusätzliche Takte für Multiplikation
- Problem: Erste Off-Trace-Operation braucht Multiplikationsergebnis

Handhabung von Mehrtakt-Operationen

Kompensationscode



A. Koch



- **Partiellen Ablaufplan** in Off-Trace-Edge einfügen
 - Bisher nur Operationen eingefügt, nun Ablaufplan!
- Verzögert Zugriff auf Multiplikationsergebnis



- Einbau von Ablaufplänen in CFG
- Kompensationscode
- Splits
- Joins
- Kombinierte Szenarios
- Bedingte Sprünge
- Mehrtaktige Operationen