



Mikroarchitektursynthese

- Scheduling -

Peter Marwedel, Robert Pyka
Informatik 12

Übersicht

- Einführung
- SystemC
 - Vorlesungen und Programmierung
- FPGAs
 - Vorlesungen
 - VHDL-basierte Konfiguration von FPGAs
- Algorithmen
 - ➔ - Mikroarchitektur-Synthese
 - Automaten-synthese
 - Logiksynthese
 - Layoutsynthese

Mikroarchitektursynthese

Synonyme:

- *behavioral synthesis*,
- *high-level synthesis*;
- Synthese der RT-Struktur (Implementierung mit Addierern, Registern, Multiplexern, ...)
- Synthese von Rechenwerk (*data path synthesis*) und Steuerwerk (*controller synthesis*)

Startebene: Verhalten in Form von Algorithmen, z.B. in imperativer Sprache

Zielebene: i.d.R. RTL (nur selten auch Geometrie betrachtet)

Literatur

- Marwedel: Synthese und Simulation von VLSI-Systemen, Hanser, 1993
- J. Teich, Digitale Hardware/Software-Systeme, Springer, 1997
- D. Gajski, N. Dutt, A. Wu, S. Lin: High-level synthesis, introduction to chip and system design, Kluwer, 1992
- R. Camposano, W. Wolf: High-Level VLSI Synthesis, Kluwer, 1991
- P. Michel, U. Lauther, P. Duzy: The Synthesis Approach to digital system design, Kluwer, 1992
- P. Eles, K. Kuchcinski, Z. Peng: System Synthesis with VHDL, Kluwer, 1998

Data Path Synthesis

Im Folgenden: Beschränkung auf das Rechenwerk
(Steuerwerke später): *data path synthesis*.

Ursprünge:

- Universität Kiel (Zimmermann, Marwedel et al.),
- Carnegie Mellon Universität Pittsburgh (Barbacci, Thomas, Parker).

Eine gründliche Analyse der Anforderungen ist v.a. bei berechnungsintensiven Spezifikationen erforderlich.

Nachfolgend betrachten wir i.d.R. *untimed behavior*, d.h. wir nehmen an, dass im Allg. das Scheduling noch nicht stattgefunden hat.

Daten- und Kontrollflussgraphen (1)

Berechnungen (ohne Sprünge) können mit Datenfluss-Graphen dargestellt werden.

Beispiel: Berechnung von Determinanten:

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}$$

als Formel:

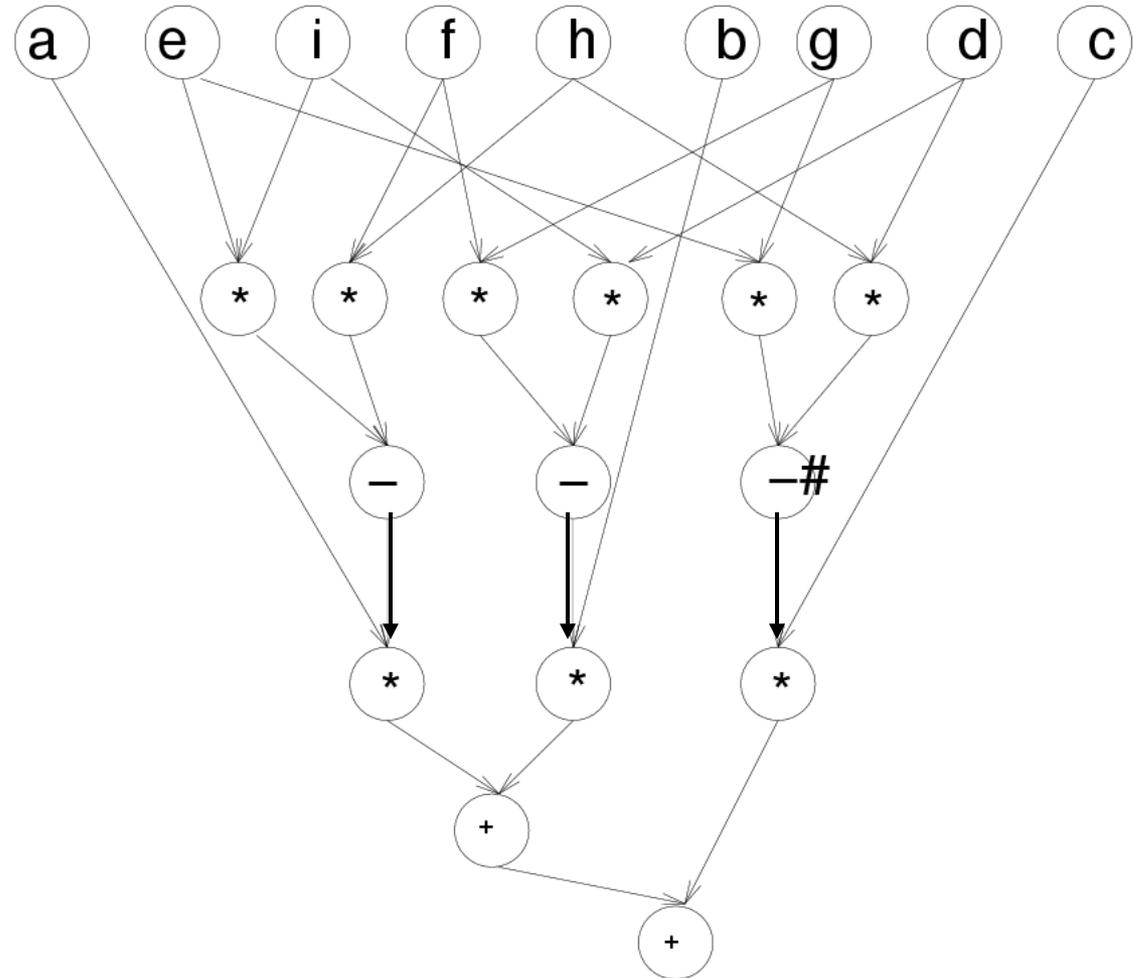
$$\det = a * (e * i - f * h) + b * (f * g - d * i) + c * (d * h - e * g)$$

Daten- und Kontrollflussgraphen (2)

Def.: Ein **Basisblock** ist eine (maximale) Codesequenz, die eine Verzweigung höchstens an ihrem Ende und mehrere Vorgänger (im Sinne eines Verschmelzens von Kontrollflüssen) höchstens an ihrem Anfang besitzt.

Bei Beschränkung auf Basisblöcke können Anweisungen zu Datenfluss-Graphen (DFGs) zusammengefasst werden

Datenflussgraph für das Determinantenbeispiel



$$\begin{aligned} \det &= a * (e * i - f * h) \\ &+ b * (f * g - d * i) \\ &+ c * (d * h - e * g) \end{aligned}$$

#: $b - a$

Kontroll/Datenflussgraphen (CDFGs)

In der Spezifikation auch Verzweigungen:

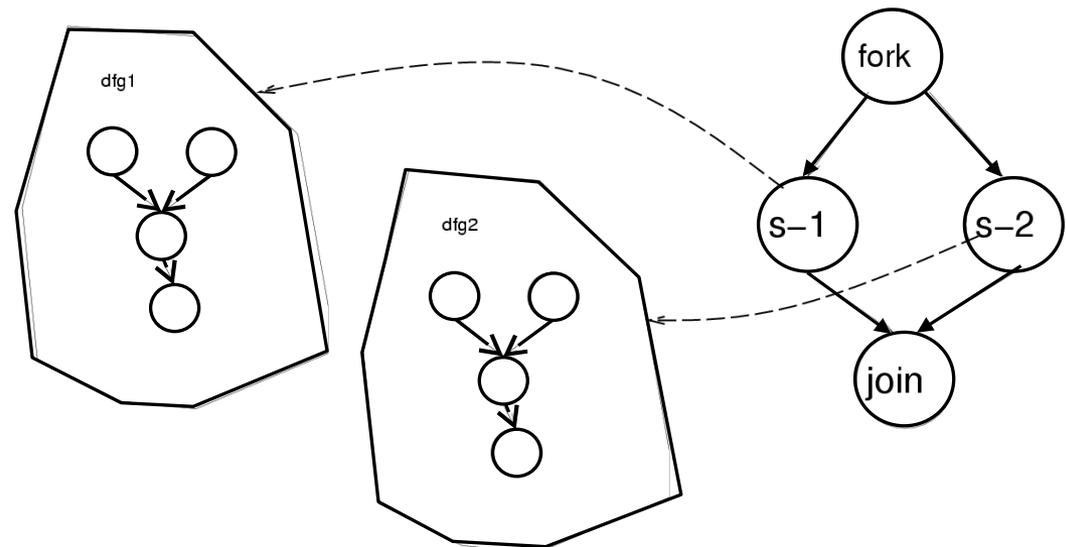
→ separate Kontrollfluss-Graphen zur Darstellung der Programm-Kontrolle.

In Kombination mit Datenfluss-Graphen und Verweisen zwischen diesen kommt man zu Kontroll/Datenfluss-Graphen (CDFGs).

Beispiel:

```

if <bedingung> then
  <statements-1>
else
  <statements-2>
  
```



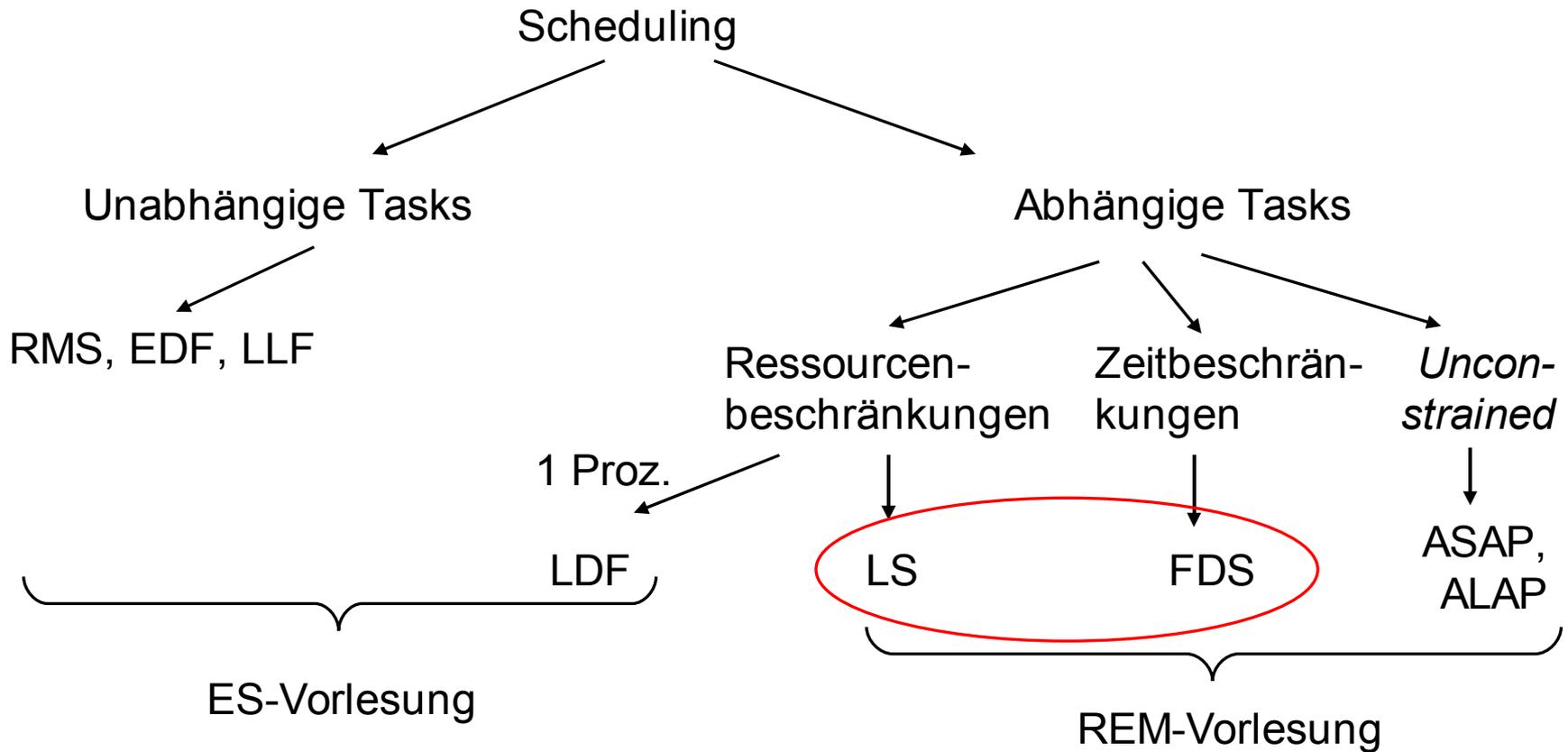
Aufgabe der Mikroarchitektursynthese

Aufgabe: zu einem vorgegebenen CFG eine HW-Implementierung erzeugen, die vorgegebene *constraints* einhält. Erwartung, dass drei Aufgaben gelöst werden:

- Das **Scheduling** (deutsch: „Ablaufplanung“). Es wird für jede Operation festgelegt, **wann** sie ausgeführt wird.
- Die **Bereitsstellung** (engl.: *allocation*) von Ressourcen. (Addierern, usw.).
- Die **Zuordnung** (engl.: *(resource) binding*). Es wird für jede Operation festgelegt, **welche** Hardware-Ressource die Ausführung übernimmt.

Suche nach *optimalen* Architekturen erfordert simultane Lösung der drei Aufgaben. Scheduling ist NP-hart.
→ vielfach Zerlegung in einzelne Phasen.

Klassen von Scheduling-Problemen



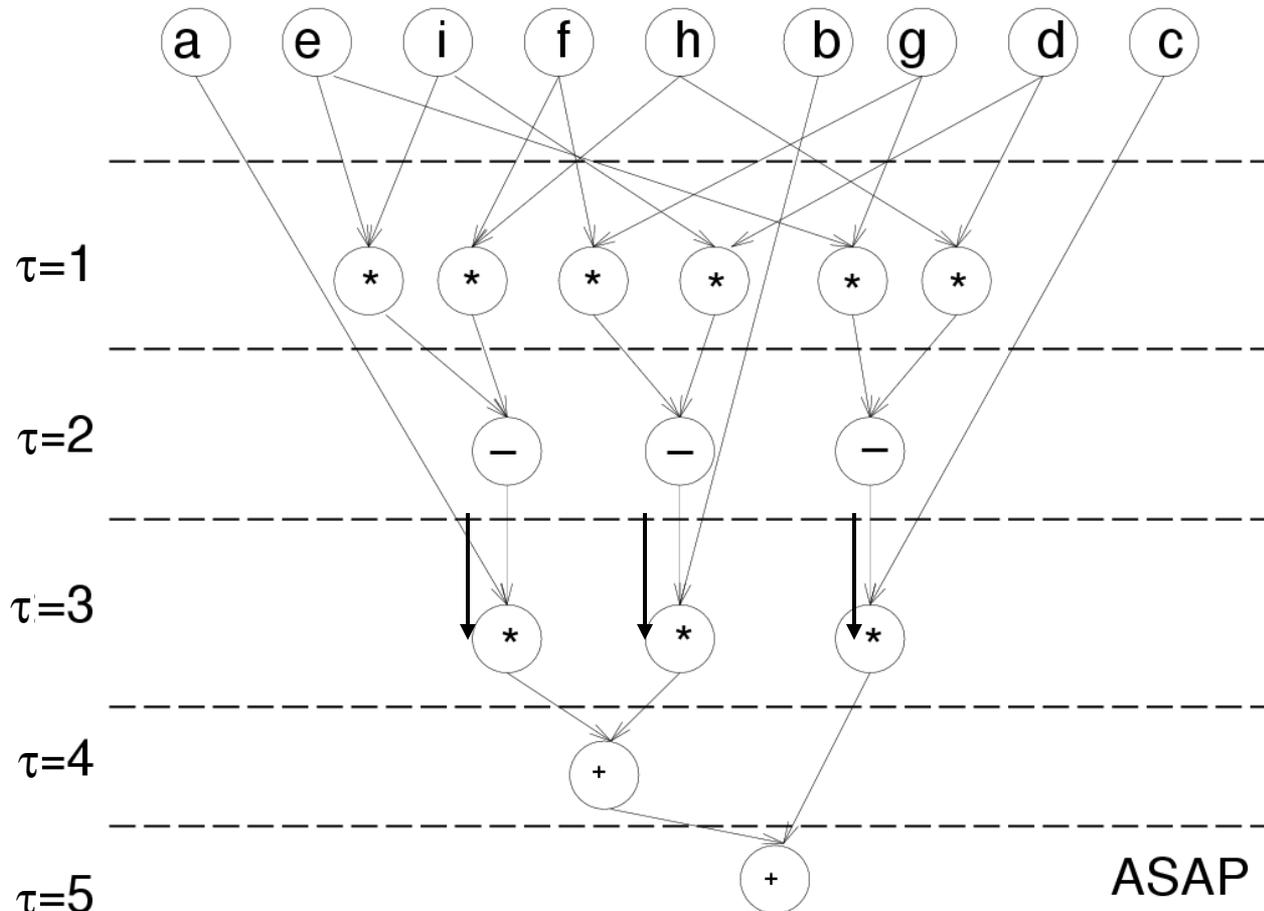
Scheduling für DFGs (1)

Einfachste
Scheduling-
Verfahren:

1. as soon as possible (ASAP) scheduling:

Operationen so früh wie möglich ausführen.

Beispiel
Determinantenberechnung:

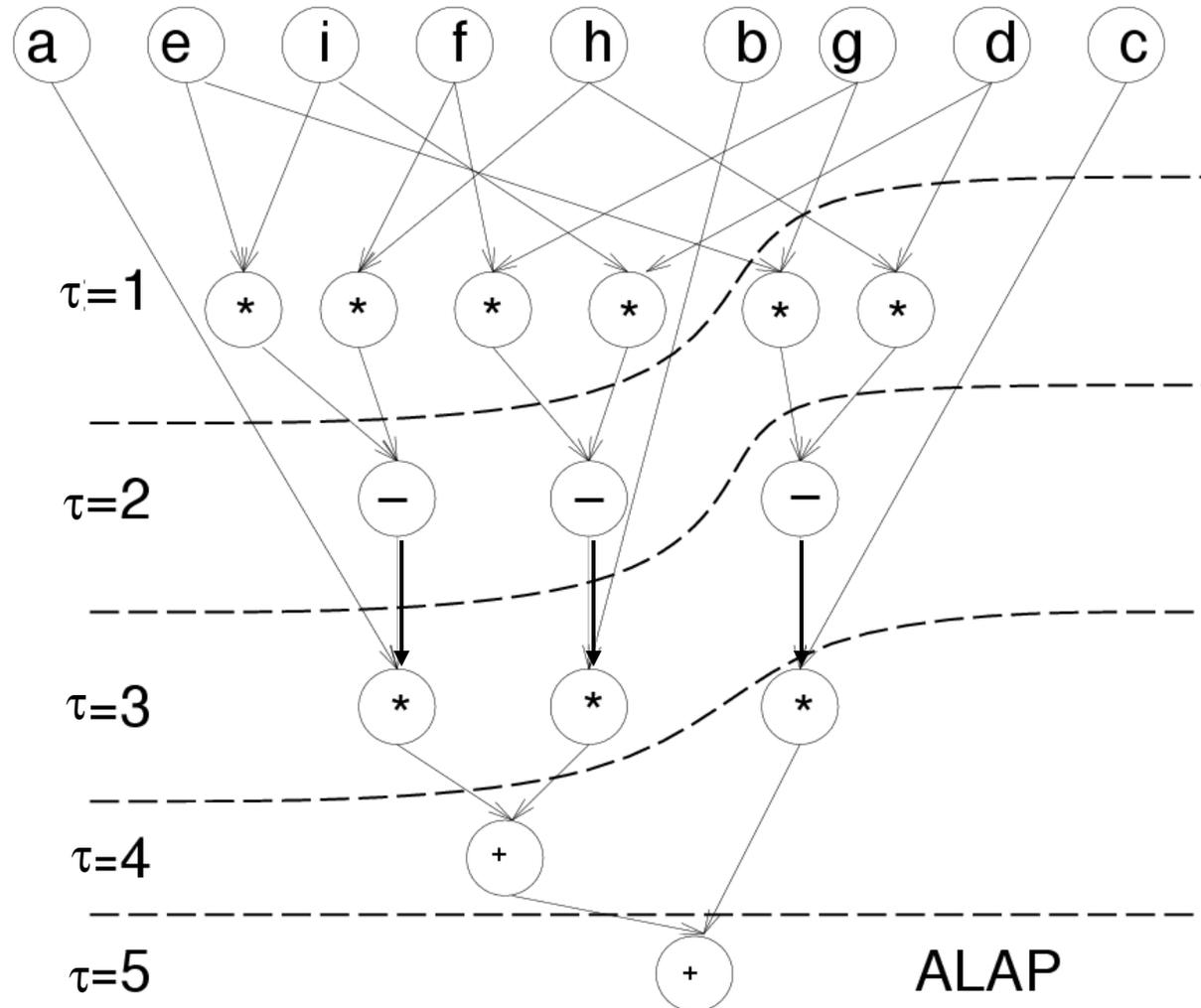


Scheduling für DFGs (2)

1. *As-late-as-possible (ALAP) scheduling:*

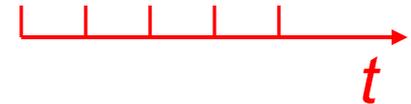
Operationen
so spät wie
möglich
ausführen.

Beispiel
Determinanten-
tenberechnung:

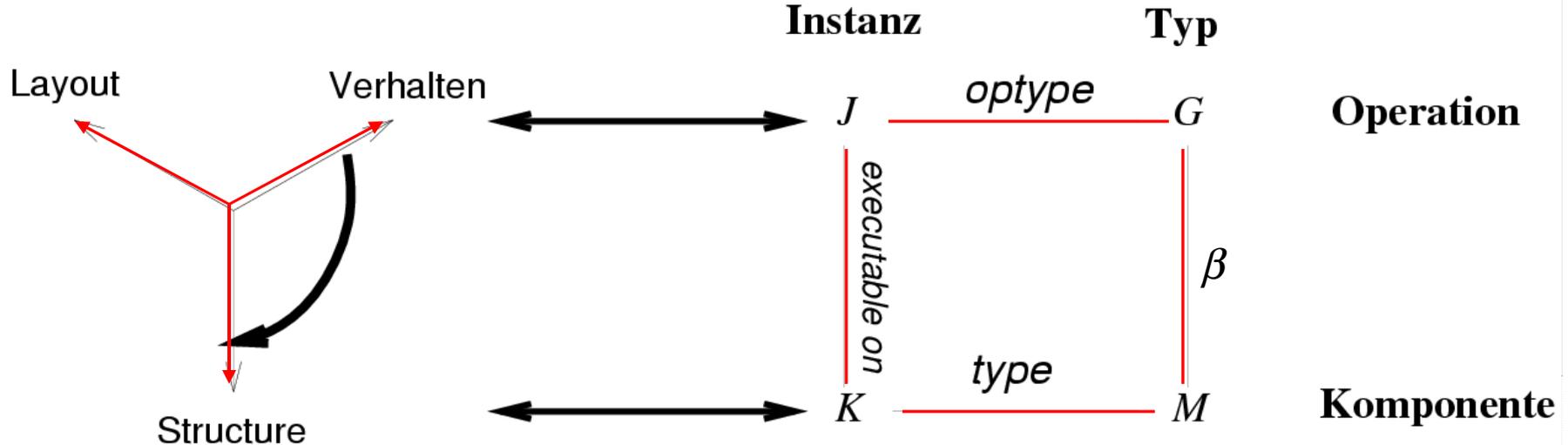


Scheduling für DFGs (3)

- Bislang: Annahme gleicher Ausführungsgeschwindigkeiten aller Operationen.
- Übliche Annahme: synchrone Hardware. Einzelne „Zeiten“: Intervalle zwischen zwei Taktimpulsen (**Kontrollschritten** \triangleq Abschnitten zwischen zwei **wait-Anweisungen beim *timed behavior***).
- Komplexere Scheduling-Verfahren: unterschiedliche Ausführungsgeschwindigkeiten, möglichst günstige Allokationen, u.a.m.



Darstellung von Ressourcen

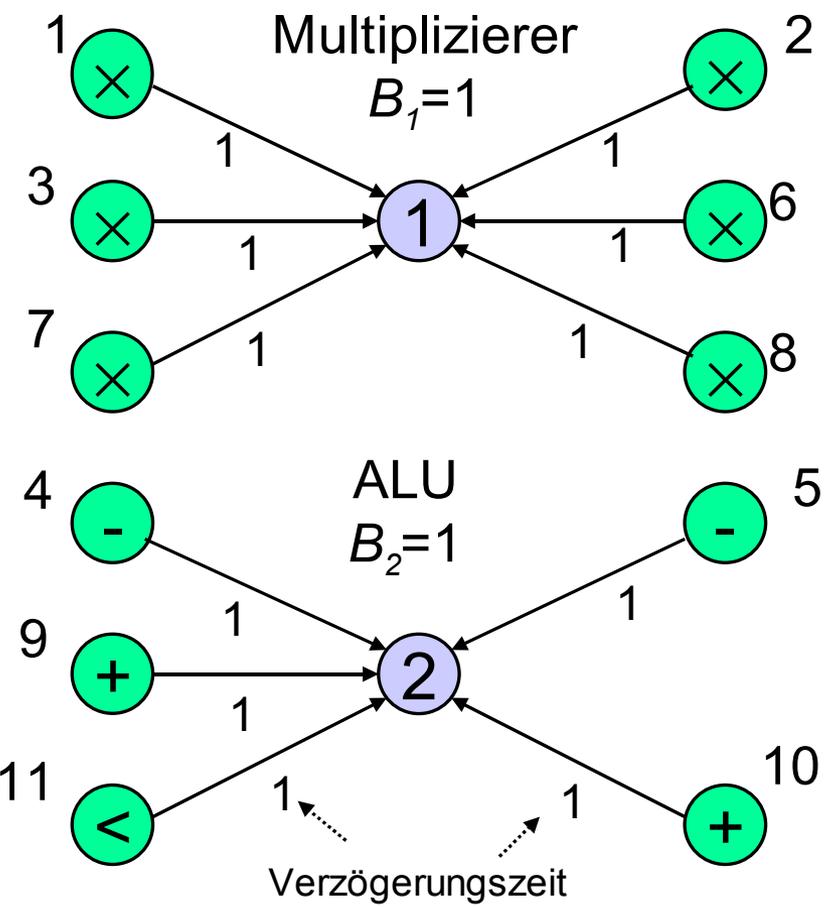
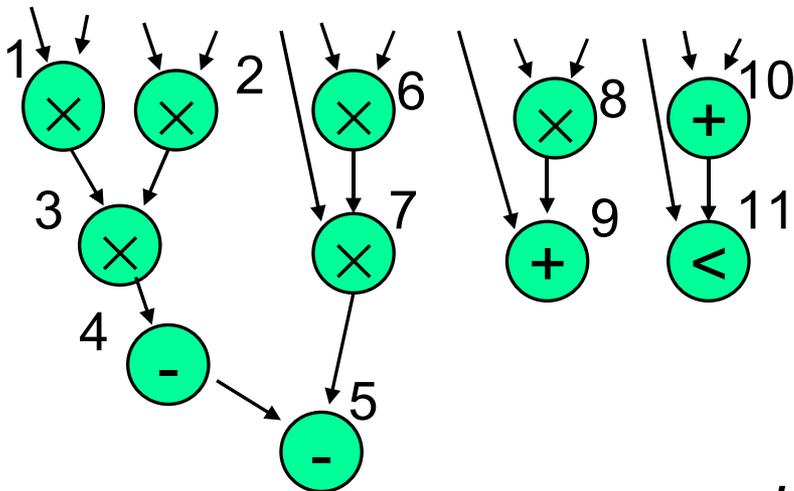


- $optype$: Typ einer Operation im DFG.
- $type: K \rightarrow M$: Typ eines Bausteinexemplars.
- $\beta(m): M \rightarrow \wp(G)$: Funktionalität eines Bausteins:
 $\forall m \in M, g \in G: g \in \beta(m) \Leftrightarrow m$ kann Operation g ausführen.
- $j \in J \text{ executable_on } k \in K \Leftrightarrow optype(j) \in \beta(type(m))$.
- $i \in I$: Indexmenge von Kontrollschritten.
- $\ell(j, m)$: Anzahl Kontrollschritte für das Ausführen von j auf m .

Ressourcengraph $G_R=(V_R, E_R)$

Darstellung von

- $\text{type}(j) \in \beta(m)$
- und $\ell = \ell(j, m)$
- und B_m : Anzahl der Bausteine vom Typ m



Nach J. Teich.

List-Scheduling

List scheduling ist eine Weiterentwicklung der ASAP/ALAP-Verfahren.

Vorbereitung:

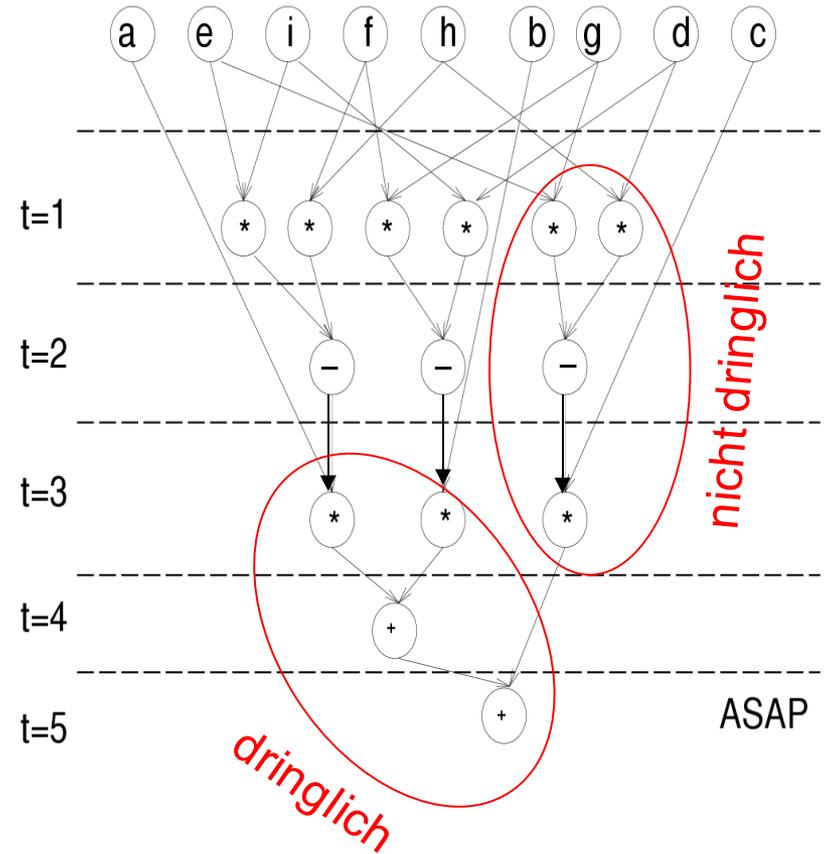
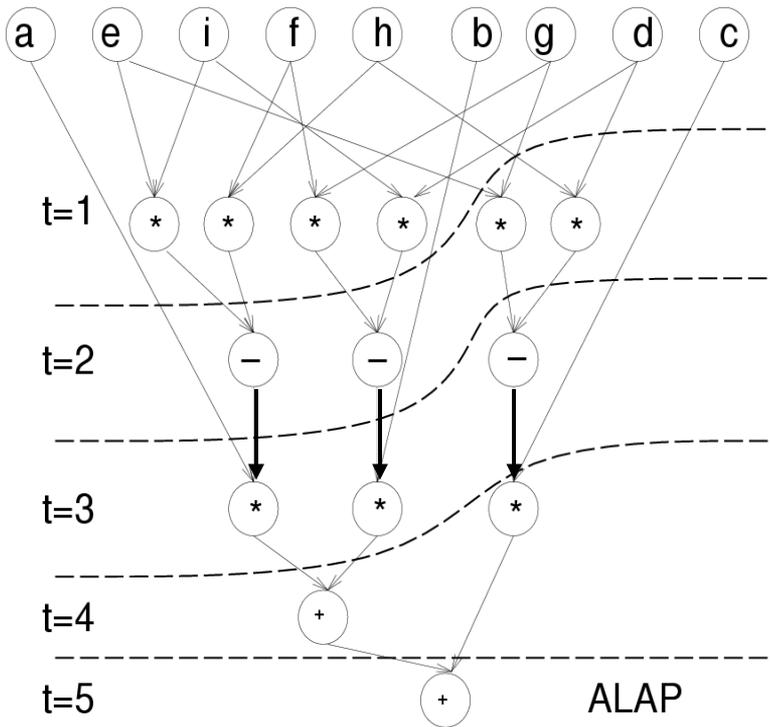
- Topologisches Sortieren des DFG $G=(V,E)$
- Ermittlung einer **Dringlichkeitsfunktion** (Priorität) für jeden Knoten:

Mögliche Dringlichkeitsfunktionen p :

- Anzahl der Nachfolgerknoten
- Gewicht des längsten Pfades
- **Mobilität**=Differenz zwischen ALAP und ASAP Schedule

Quelle: Teich: Dig. HW/SW Systeme

Mobility als Dringlichkeitsmaß



Mobility erlaubt keine präzise Bevorzugung bestimmter Knoten.

Wiederholt durchzuführende Berechnungen

- Wiederholte Berechnung der Menge der auf einem Bausteintyp m ausführungsbereiten Operationen, deren Vorgänger alle bekannt sind:

$$A_{i,m} = \{v_j \in V: \text{type}(v_j) \in \beta(m) \wedge \forall j': (v_{j'}, v_j) \in E: i > \tau(v_{j'}) + \ell(j) - 1\}$$
- Menge der Operationen des Typs m , die im Kontrollschritt i noch ausgeführt werden:

$$G_{i,m} = \{v_j \in V: \text{type}(v_j) \in \beta(m) \wedge \tau(v_j) + \ell(j) - 1 \geq i\}$$
- Bestimmung einer Menge S_i von zu startenden Operationen mit

$$|S_i| + |G_{i,m}| \leq B_m$$

Algorithmus

List($G(V,E)$, β , B , ρ , ℓ) {

$i:=0$;

repeat {

for ($m=1$) **to** Anzahl Modultypen {

Bestimme Kandidatenmenge $A_{i,m}$;

Bestimme Menge nicht beendeter Operationen $G_{i,m}$;

Wähle eine Menge maximaler Priorität S_i mit

$$|S_i| + |G_{i,m}| \leq B_m$$

foreach ($v_j \in S_i$): $\tau(v_j) := i$; (*setze schedule fest*)

} $i:=i+1$;

}

until (alle Knoten von V geplant);

return (τ);

Auch ohne Ressourcen-
beschränkung
anwendbar

Komplexität: $O(|V|)$



Beispiel

Schedule für
Ressourcengraph wie vorhin
mit der Länge des Pfades als
Dringlichkeitsmaß p .

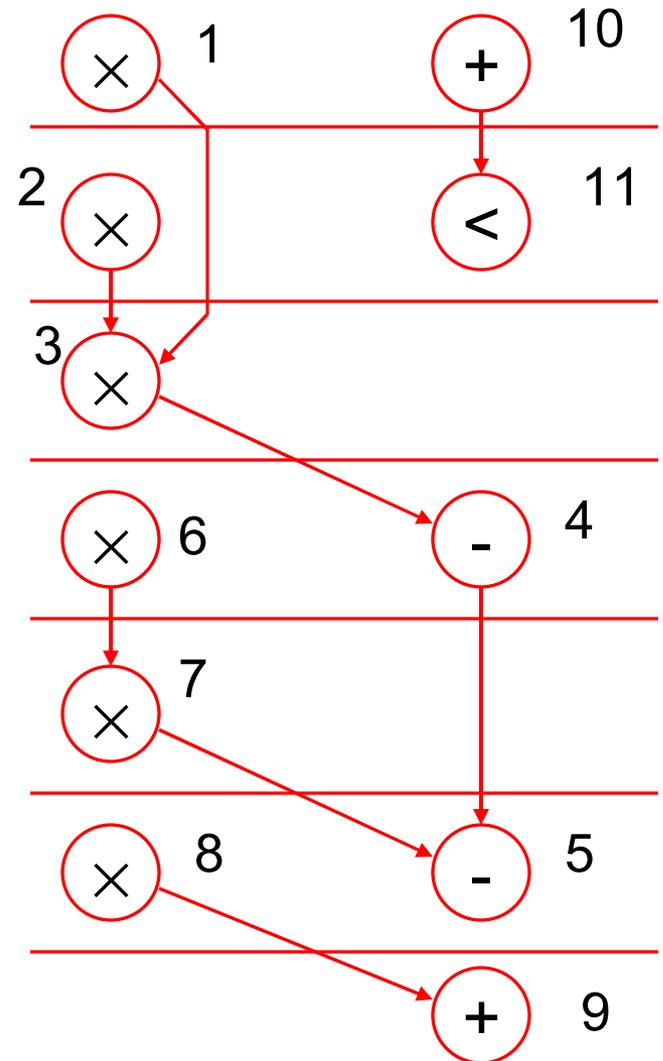
$$p(v_1)=p(v_2)=4$$

$$p(v_3)=p(v_6)=3$$

$$p(v_4)=p(v_7)=p(v_8)=p(v_{10})=2$$

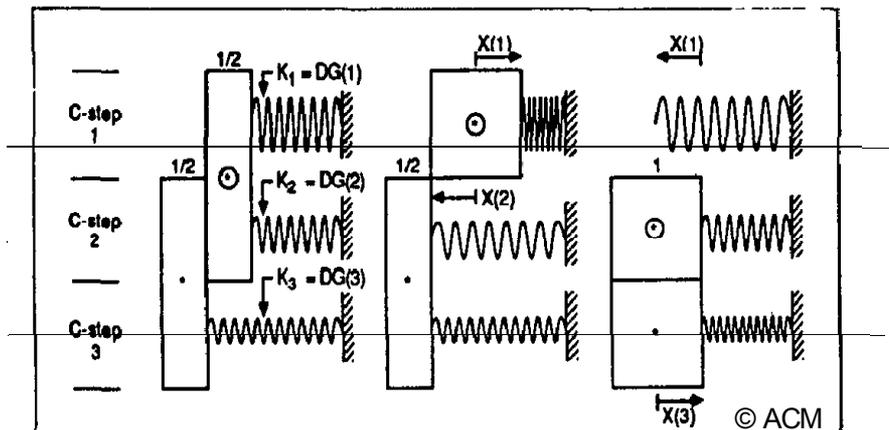
$$p(v_5)=p(v_9)=p(v_{11})=1$$

$$G_{i,m}=0, \text{ weil } \ell=1 \quad \forall i,j,m.$$



Scheduling mit Zeitbeschränkungen

*Force-directed scheduling** zielt auf eine gleichmäßige Ressourcenauslastung bei vorgegebener Ausführungszeit. Basiert auf Federmodell und Hooke'schem Gesetz:



* [Pierre G. Paulin, J.P. Knight, Force-directed scheduling in automatic data path synthesis, *Design Automation Conference* (DAC), 1987, S. 195-202]



Das *diffeq*-Beispiel

Die Differentialgleichung

$$y'' + 3zy' + 3y = 0$$

kann durch den folgenden Algorithmus gelöst werden:

while ($z < a$) **do**

begin

$z_l := z + dz;$

$u_l := u - (3 \cdot z \cdot u \cdot dz) - (3 \cdot y \cdot dz);$

$y_l := y + (u \cdot dz);$

$z := z_l;$

$u := u_l;$

$y := y_l;$

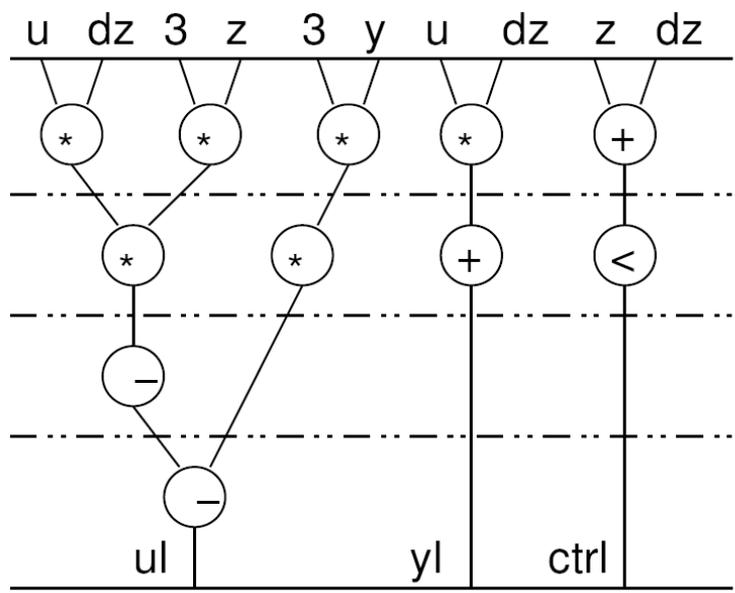
end;

ASAP- und ALAP-schedules von *diffeq*

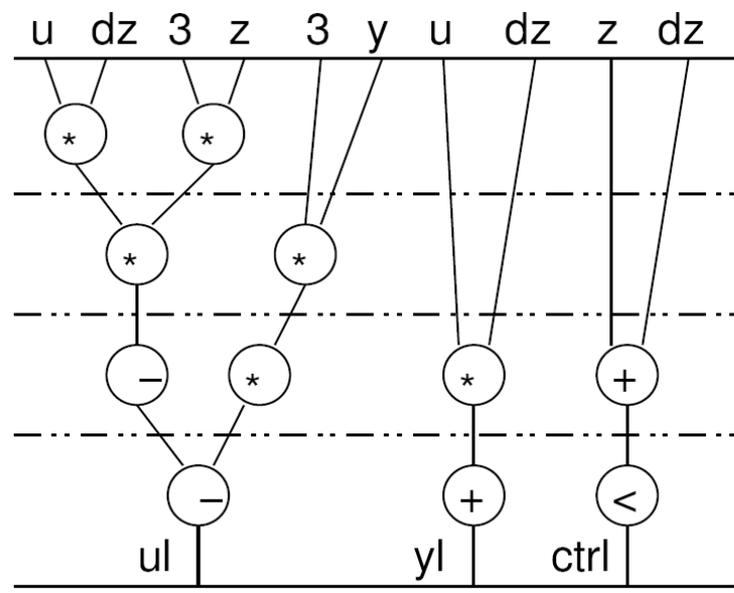
```

while (z < a) do
begin
  zl := z + dz;    ul := u - (3 · z · u · dz) - (3 · y · dz);
  yl := y + (u · dz); z := zl;
  u := ul;        y := yl;
end;
    
```

ASAP



ALAP



Jeweils Betrachtung von Klassen von Operationen

Partitionierung der (im DFG benutzten) Operationstypen G in Klassen $\{H_p \in \wp(G)\}$ so, dass für die Operationen in den verschiedenen H_p jeweils disjunkte Bausteintypen in Frage kommen:

$$\forall g \in H_p \quad \forall g' \in H_{p'} : \{m \mid g \in \beta(m)\} \cap \{m' \mid g' \in \beta(m')\} = \emptyset$$

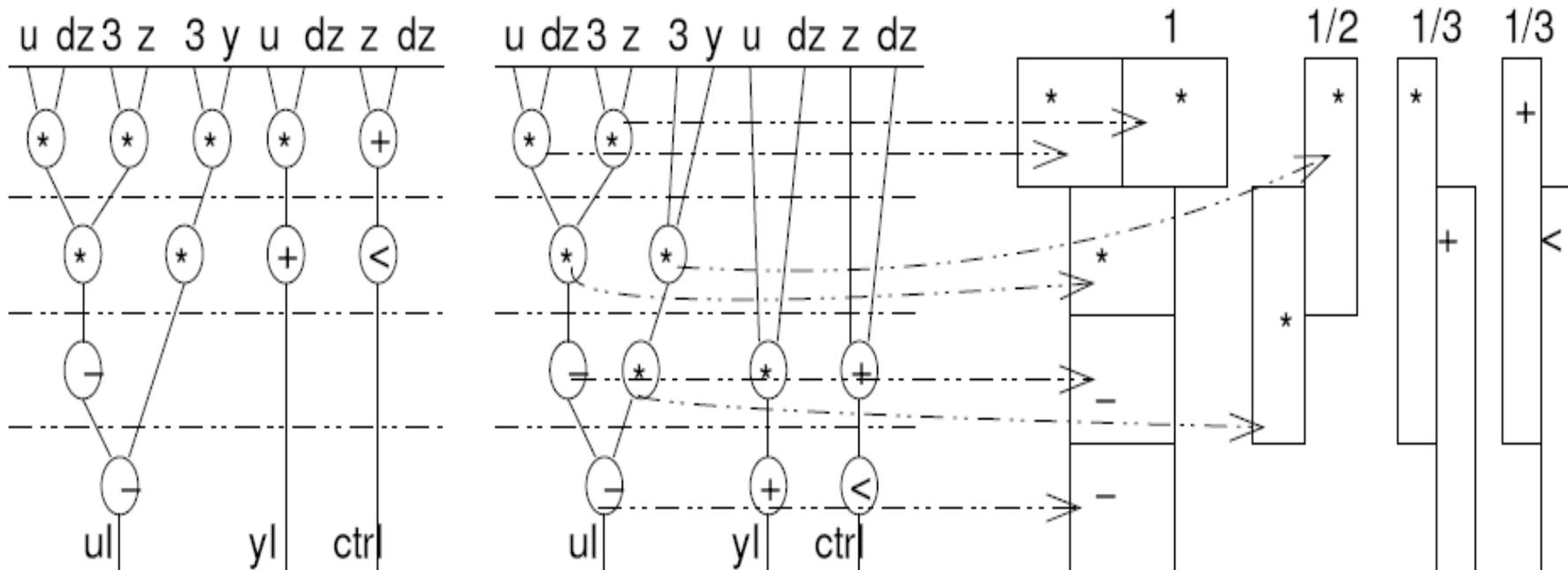
Beispiel:

$$\beta(1) = \{+, -\}, \quad \beta(2) = \{*\} \rightarrow H_1 = \{+, -\}, \quad H_2 = \{*\}$$

Verfahren muss für alle H_p durchgeführt werden.

Im Folgenden: Betrachtung einer repräsentativen Menge H .

1. Erzeugung eines Zeitrahmens $R(j)$ und 2. einer „Wahrscheinlichkeit“ $P(j,i)$ für Zuordnung $j \rightarrow i$



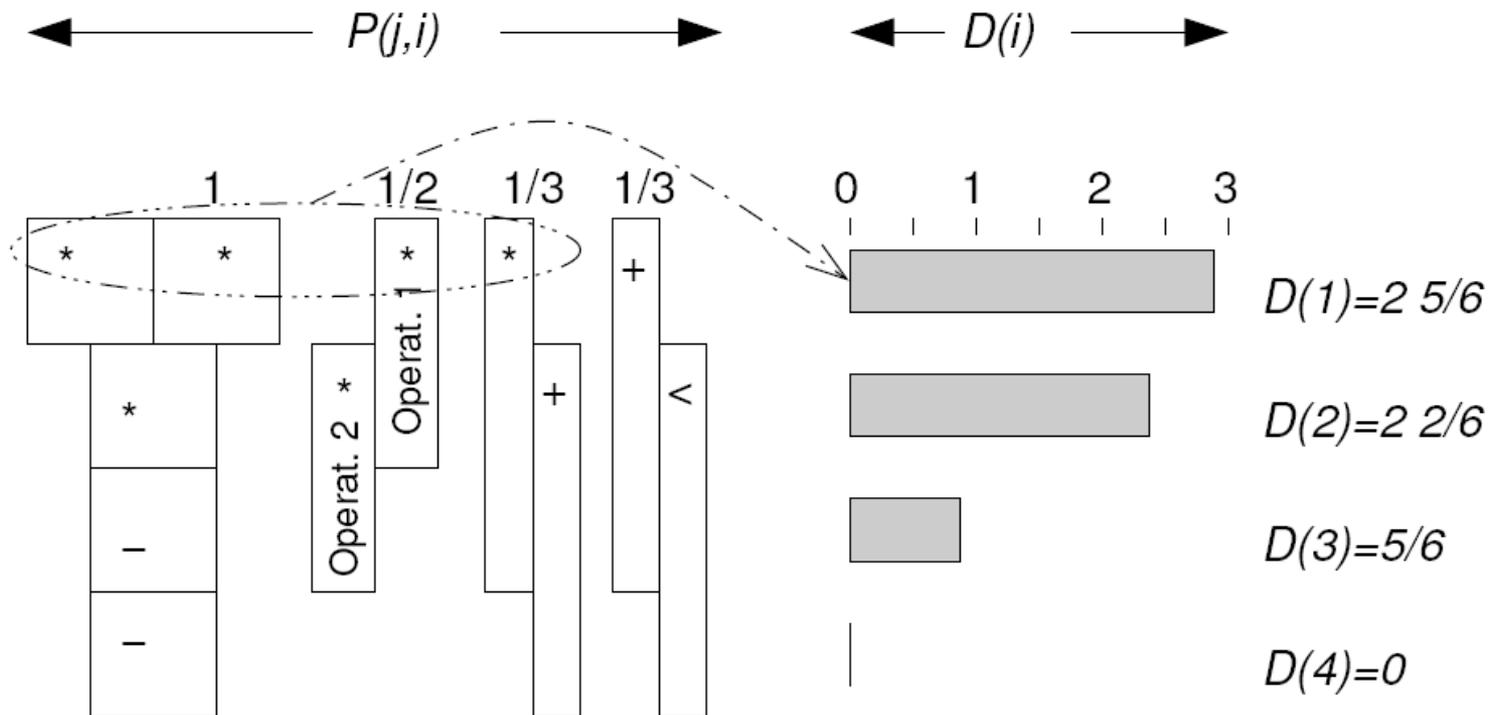
$R(j) = \{\text{ASAP-Kontrollschritt} \dots \text{ALAP-Kontrollschritt}\}$

$$P(j, i) = \begin{cases} \frac{1}{|R(j)|} & \text{falls } i \in R(j) \\ 0 & \text{sonst} \end{cases}$$

3. Bestimmung einer „Verteilung“ $D(i)$

(Anzahl von Operationen aus H im Kontrollschritt i)

$$D(i) = \sum_{j, \text{type}(j) \in H} P(j, i)$$

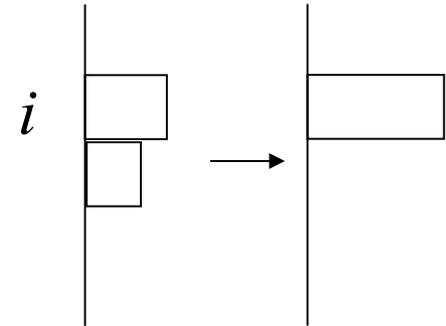


4. Berechnung von direkten Kräften (1)

- $\Delta P_i(j, i')$ ist die Änderung der Kraft auf j im Kontrollschritt i' , wenn j auf i abgebildet wird.

Die neue Wahrscheinlichkeit j in i auszuführen, ist 1; die alte war $P(j, i)$.

Die neue Wahrscheinlichkeit j in $i' \neq i$ auszuführen, ist 0; die alte war $P(j, i)$.



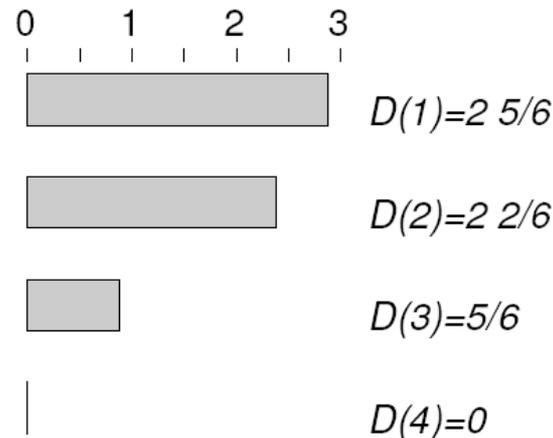
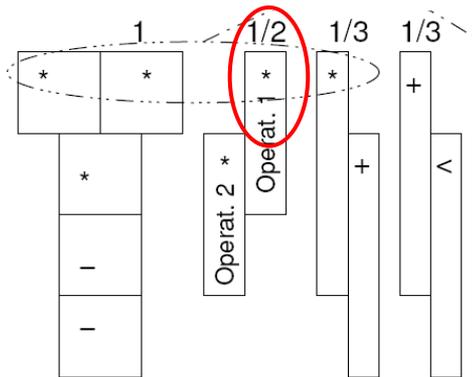
$$\text{☞ } \Delta P_i(j, i') = \begin{cases} 1 - P(j, i) & \text{falls } i = i' \\ -P(j, i) & \text{sonst} \end{cases}$$

4. Berechnung von direkten Kräften (2)

- $SF(j, i)$ ist die gesamte Änderung der (direkten) Kräfte aufgrund der Zuordnung von j zu i .

$$SF(j, i) = \sum_{i' \in R(j)} D(i') \Delta P_i(j, i') \quad \Delta P_i(j, i') = \begin{cases} 1 - P(j, i) & \text{falls } i = i' \\ -P(j, i') & \text{sonst} \end{cases}$$

Beispiel

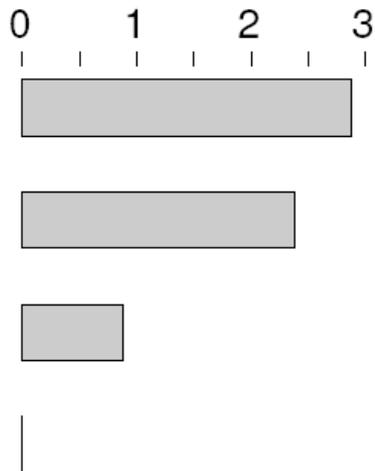
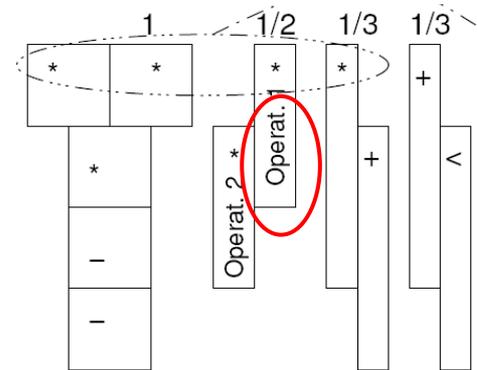


$$SF(1, 1) = 2 \frac{5}{6} (1 - \frac{1}{2}) - 2 \frac{2}{6} (\frac{1}{2}) =$$

$$\frac{1}{2} (\frac{17}{6} - \frac{14}{6}) = \frac{1}{2} (\frac{3}{6}) = \frac{1}{4}$$

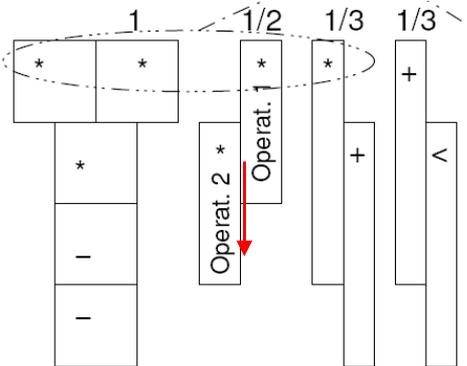
4. Berechnung von direkten Kräften (3)

Berechnung der direkten Kraft für die Zuordnung von Operation 1 in Kontrollschritt 2.



$$\begin{aligned}
 D(1) &= 2 \frac{5}{6} & SF(1, 2) &= D(1) * \Delta P_2(1, 1) + D(2) * \Delta P_2(1, 2) \\
 D(2) &= 2 \frac{2}{6} & &= 2 \frac{5}{6} * (-0,5) + 2 \frac{2}{6} * 0.5 \\
 D(3) &= 5/6 & &= -\frac{17}{12} + \frac{14}{12} \\
 D(4) &= 0 & &= -\frac{3}{12} = -\frac{1}{4}
 \end{aligned}$$

5. Berechnung von indirekten Kräften (1)



Die Zuordnung von Operation 1 zu CS 2 impliziert die Zuordnung von Operation 2 zu CS 3

☞ Betrachtung von Vorgänger- und Nachfolgerkräften

$$VF(j, i) = \sum_{j' \in \text{Vorgänger von } j} \sum_{i' \in I} D(i') \Delta P_{j,i}(j', i')$$

$$NF(j, i) = \sum_{j' \in \text{Nachfolger von } j} \sum_{i' \in I} D(i') \Delta P_{j,i}(j', i')$$

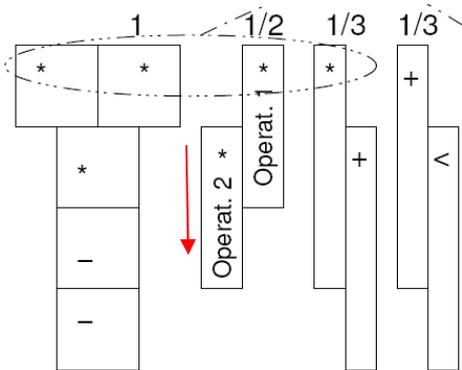
$\Delta P_{j,i}(j', i')$ ist die Änderung der Wahrscheinlichkeit der Zuordnung von j' zu i' aufgrund der Zuordnung von j zu i

5. Berechnung von indirekten Kräften (2)

$$VF(j, i) = \sum_{j' \in \text{Vorgänger von } j} \sum_{i' \in I} D(i') \Delta P_{j,i}(j', i')$$

$$NF(j, i) = \sum_{j' \in \text{Nachfolger von } j} \sum_{i' \in I} D(i') \Delta P_{j,i}(j', i')$$

Beispiel: Berechnung der Nachfolgerkraft für die Zuordnung von Operation 1 in Kontrollschritt 2



$$\begin{aligned}
 NF(1, 2) &= D(2) * \Delta P_{1,2}(2, 2) + D(3) * \Delta P_{1,2}(2, 3) \\
 &= 2\frac{2}{6} * (-0,5) + \frac{5}{6} * 0.5 \\
 &= -\frac{14}{12} + \frac{5}{12} \\
 &= -\frac{9}{12} = -\frac{3}{4}
 \end{aligned}$$

Gesamtkräfte

Die Gesamtkraft ergibt sich als Summe der direkten und der indirekten Kräfte:

$$F(j, i) = SF(j, i) + VF(j, i) + NF(j, i)$$

Im Beispiel:

$$F(1, 2) = SF(1, 2) + NF(1, 2) = -\frac{1}{4} + \left(-\frac{3}{4}\right) = -1$$

Der niedrige Wert lässt die Zuordnung von Operation 1 zu CS 2 sehr vorteilhaft erscheinen.

Gesamtablauf

procedure kräfteverfahren;

begin

ASAP-Scheduling;

ALAP-Scheduling;

while nicht alle Operationen eingeplant **do**

begin

wähle Operation mit niedrigster Gesamtkraft aus;

plane Operation in dem Kontrollschritt mit

niedrigster Kraft ein;

berechne Ausführungsintervalle neu;

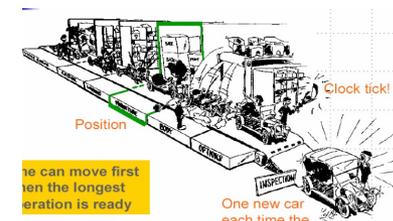
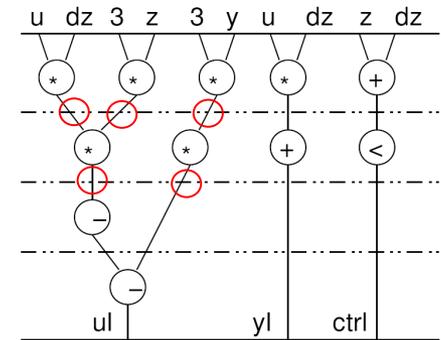
berechne $D(i)$ neu;

end;

end

Eigenschaften von *force-directed scheduling* (FDS)

- Kann auf die Behandlung von Pufferregistern und Verbindungen ausgedehnt werden, indem die entsprechenden Operationen und Ressourcen eingeführt werden.
- FDS kann mit LS kombiniert werden.
- Es gibt diverse Erweiterungen, die bestimmte Nachteile ausgleichen.
- FDS ist ein populäres Verfahren auch außerhalb des Mikroelektronikentwurfs.

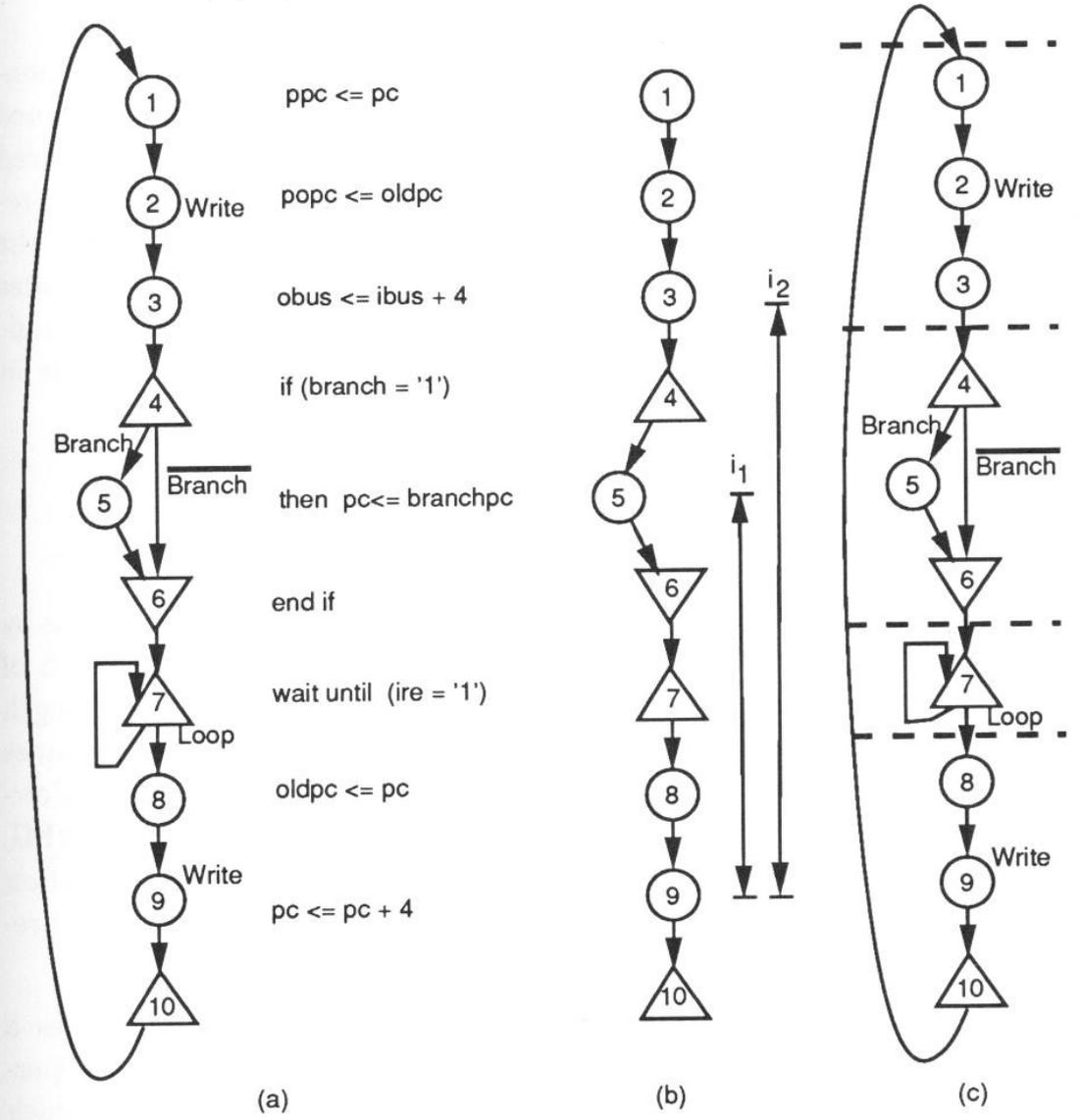


www.it.lth.se/courses/ds/material/Lectures/Lecture6.pdf

Path-based scheduling

Schwergewicht auf möglichst großer Ausführungsgeschwindigkeit der wichtigsten Pfade.
 Zunächst werden alle Pfade berechnet.
 Diese werden dann unabhängig voneinander „eingeplant“ und anschließend werden die Schedules verschmolzen.

Input Ports: ppc, ibus, branch, ire
Output Ports: popc, popc, obus
Variables: pc, oldpc



© Gajski et al.

Figure 7.14: Path-based scheduling: (a) an example CDFG, (b) a path in the CDFG with constraint intervals, (c) scheduled CDFG.

Zusammenfassung

- Begriff der Mikroarchitektur-Synthese
- Scheduling
 - ASAP
 - ALAP
 - *List scheduling* (LS)
 - *Force-directed scheduling* (FDS)