technische universität
dortmund

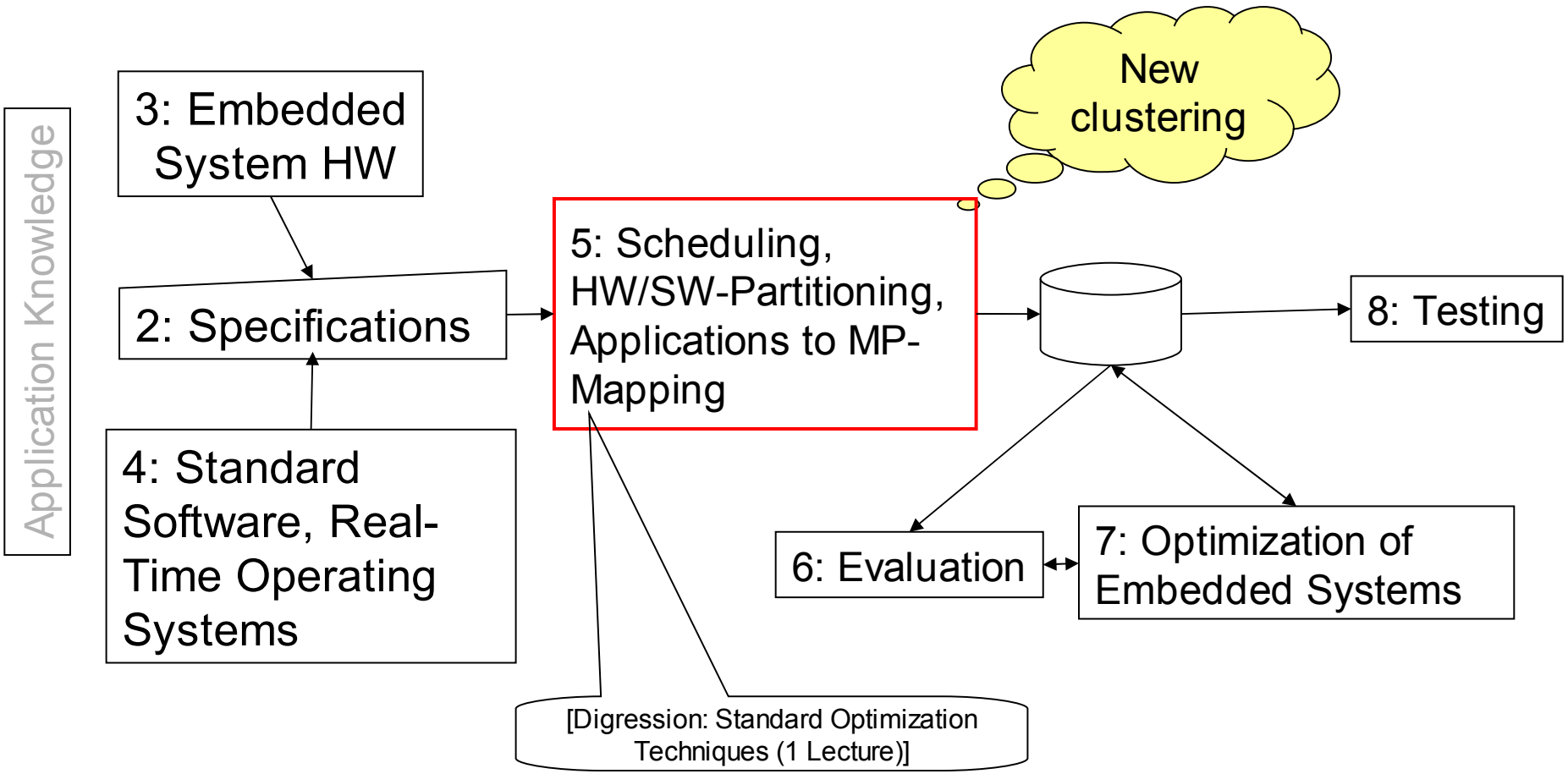
fakultät für informatik
informatik 12

Scheduling abhängiger Tasks
- Techniken aus Mikroarchitektursynthese -

Peter Marwedel
TU Dortmund
Informatik 12
Germany

2008/11/15

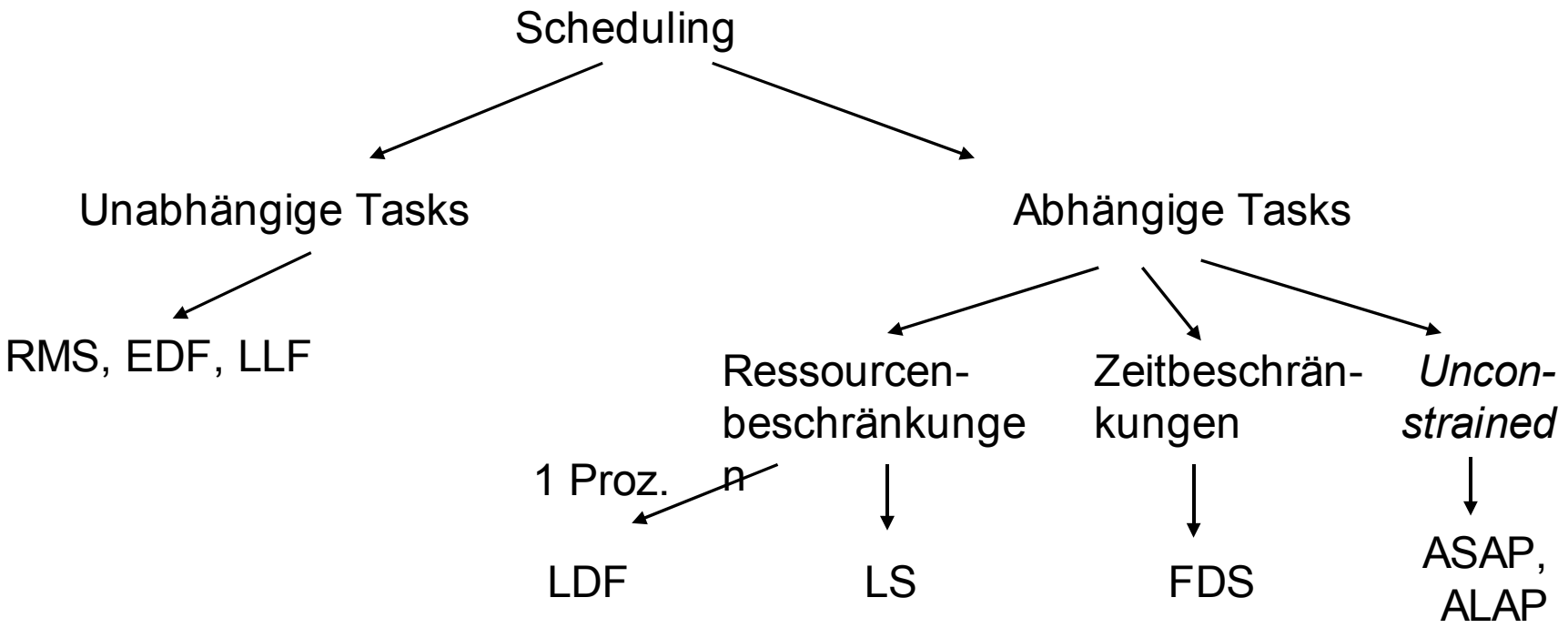
Structure of this course



Classes of mapping algorithms considered in this course

- **Classical scheduling algorithms**
  Mostly for independent tasks & ignoring communication, mostly for mono- and homogeneous multiprocessors
- **Dependent tasks as considered in architectural synthesis**
  Initially designed in different context, but applicable
- **Hardware/software partitioning**
  Dependent tasks, heterogeneous systems, focus on resource assignment
- **Design space exploration using genetic algorithms**
  Heterogeneous systems, incl. communication modeling

Klassen von Scheduling-Problemen



Nutzung von Scheduling-Algorithmen der Mikroarchitektursynthese

Synthese der RT-Struktur aus Algorithmen

Synonyme:

- *architectural synthesis*
- *behavioral synthesis,*
- *high-level synthesis*;

Übersetzung

- Operationen ↔ Tasks
- Ressourcen ↔ Prozessoren
- Datenflussgraphen (vielfach Bäume) ↔ Taskgraphen
- Kontrollschritte (vielfach Intervalle der Länge 1) ↔ Zeitintervalle

Datenflussgraphen

Berechnungen ohne Sprünge (d.h. innerhalb eines Basisblocks) können mit Datenfluss-Graphen dargestellt werden.

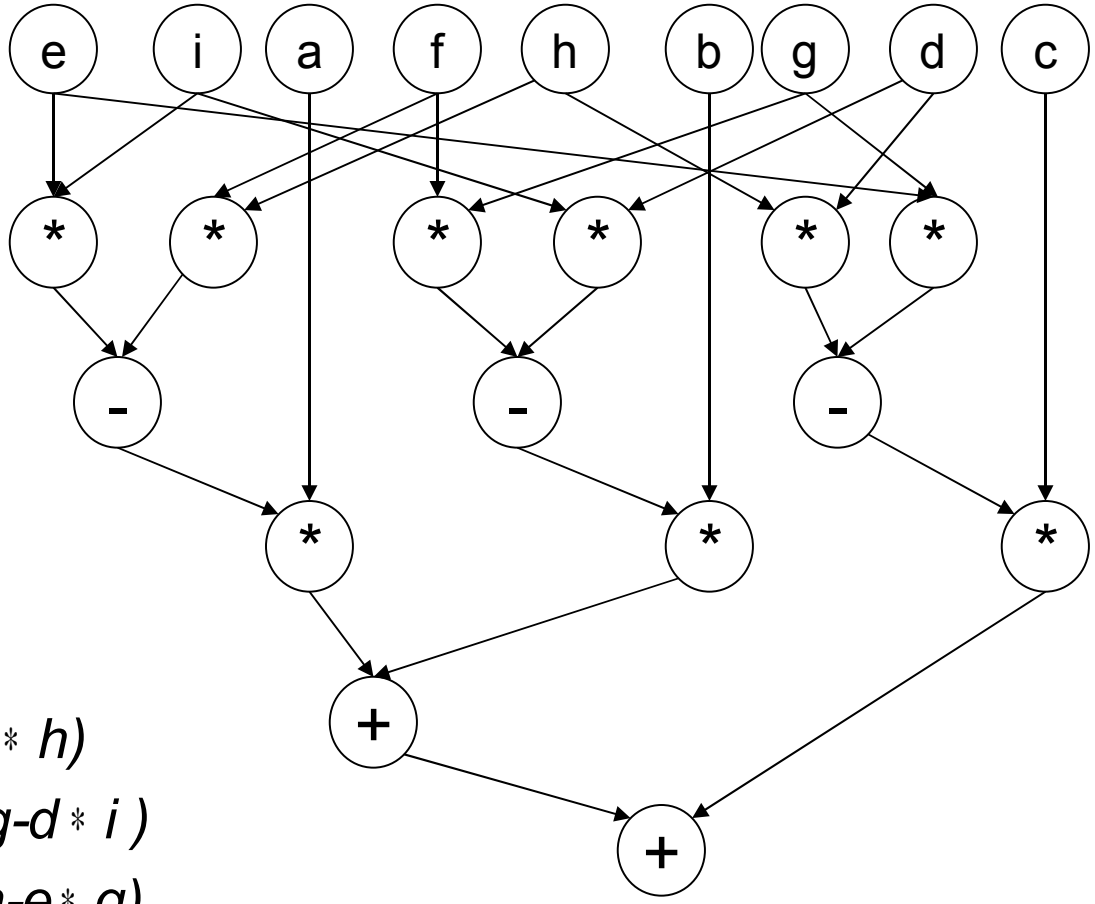
Beispiel: Berechnung von Determinanten:

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

als Formel:

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

Datenflussgraph für das Determinantenbeispiel



det = $a * (e * i - f * h)$

$+ b * (f * g - d * i)$

$+ c * (d * h - e * g)$

As soon as possible (ASAP) scheduling: Prinzip

ASAP: Alle Operationen werden so früh wie möglich eingeplant

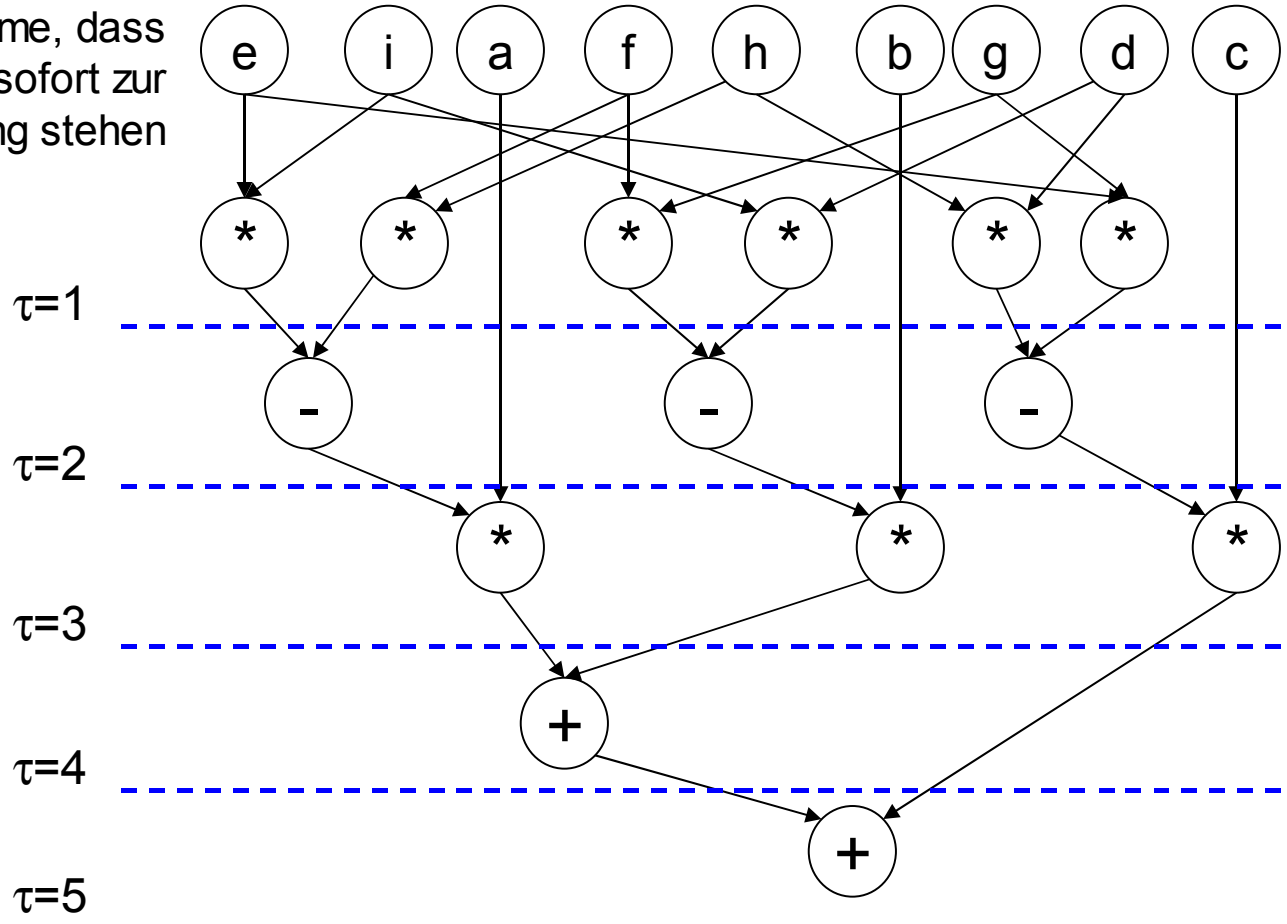
Schleife über alle Zeitschritte (aufsteigend):

- Bestimme alle noch nicht eingeplanten Operationen, für die alle Vorgängeroperationen bereits ein Ergebnis geliefert haben.
- Plane den Start dieser Operationen für den aktuellen Zeitschritt ein.

As soon as possible (ASAP) scheduling: Beispiel

Determinante

Annahme, dass Werte sofort zur Verfügung stehen

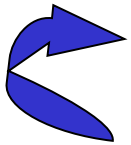


Ende der Bearbeitung der Operation bzw. Nummer des Kontroll-schritts.

As-late-as-possible (ALAP) scheduling: Prinzip

ALAP: Alle Operationen werden so spät wie möglich eingeplant

Starte beim letzten Kontrollschritt bzw. bei maximaler Zeit*:



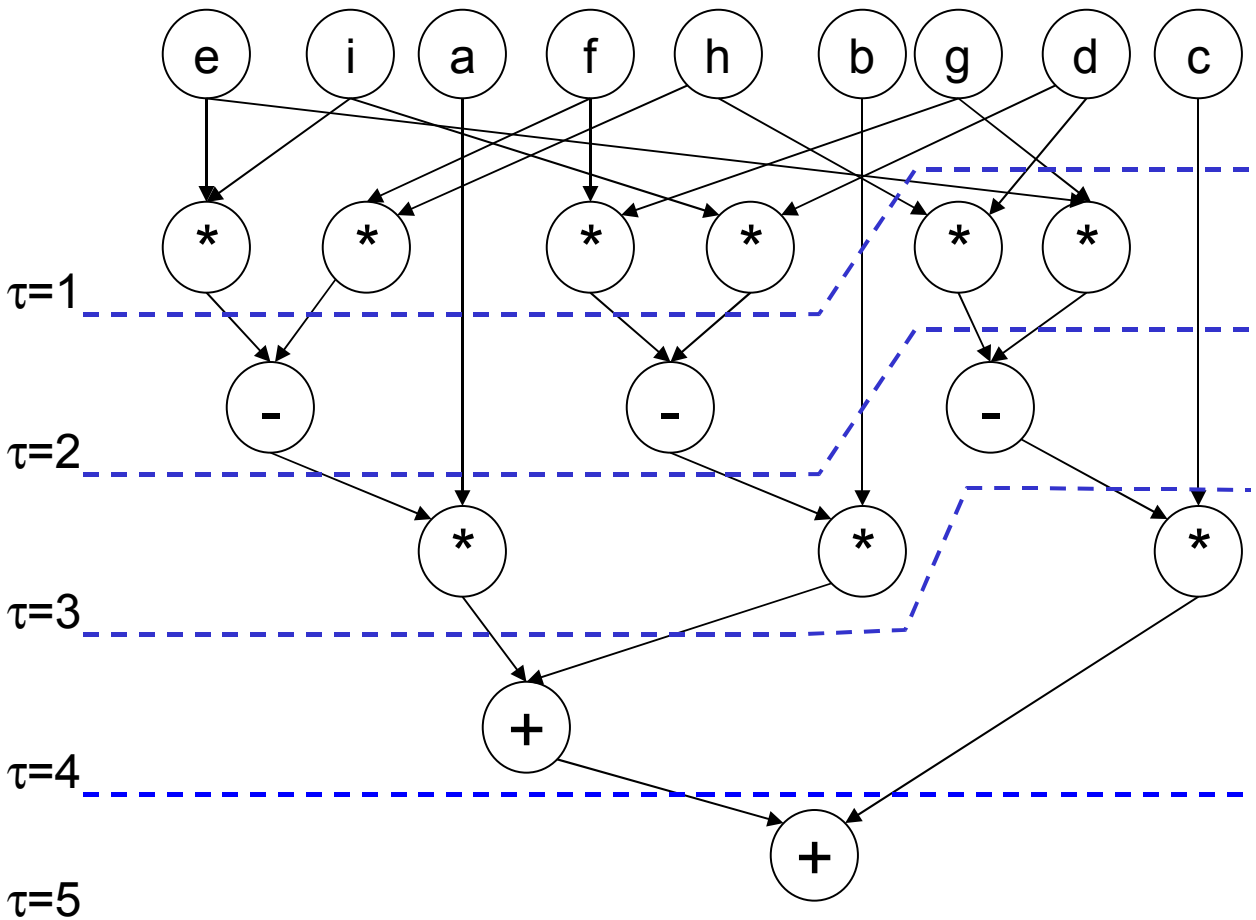
Plane Operationen ohne Nachfolger bzw. mit eingeplanten Nachfolgern zeitlich ein.

* Man baut eine Liste von „Kontrollschritten" bzw. von Operationen, die in einem Zeitintervall stattfinden sollen, von hinten nach vorn auf.

As-late-as-possible (ALAP) scheduling: Beispiel

Determinante

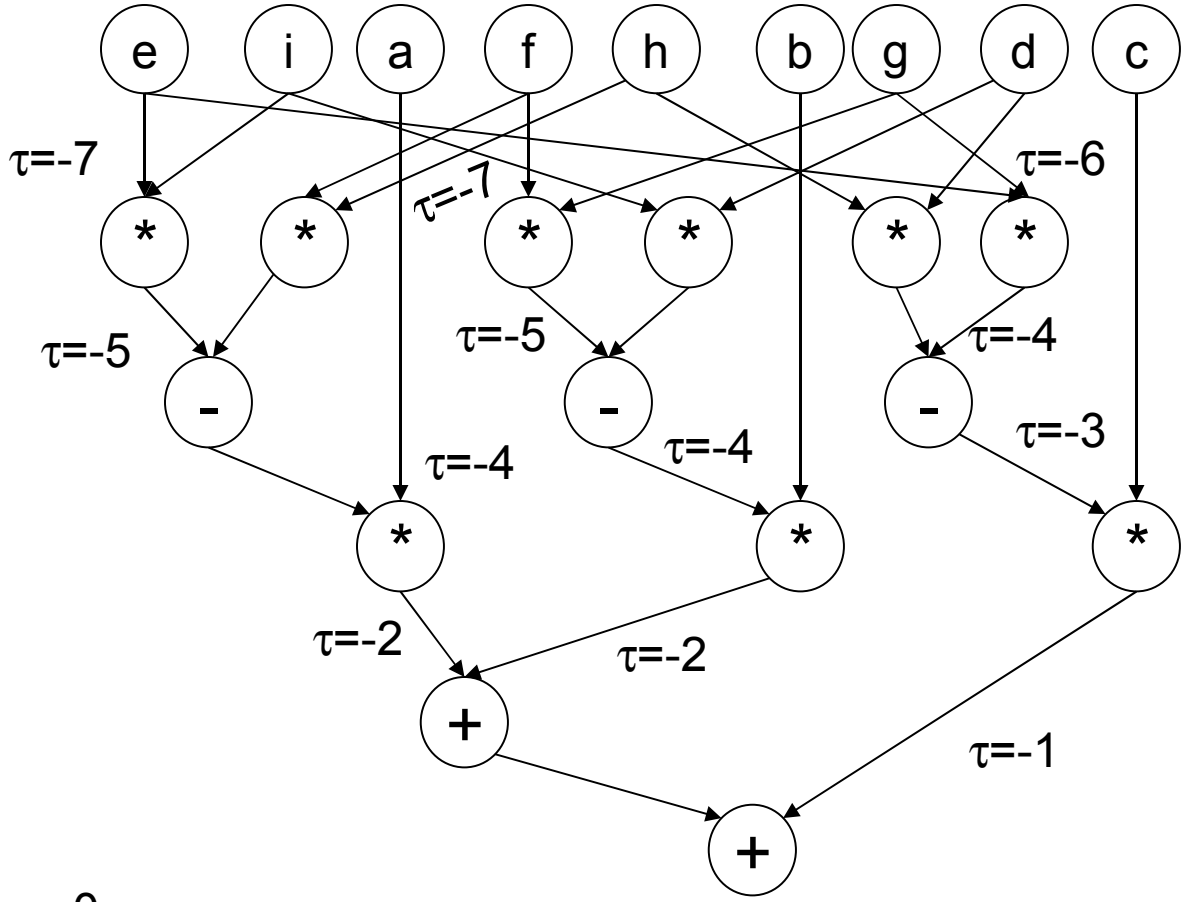
Schedule besitzt Einfluss auf die Anzahl benötigter Ressourcen



Starte hier:

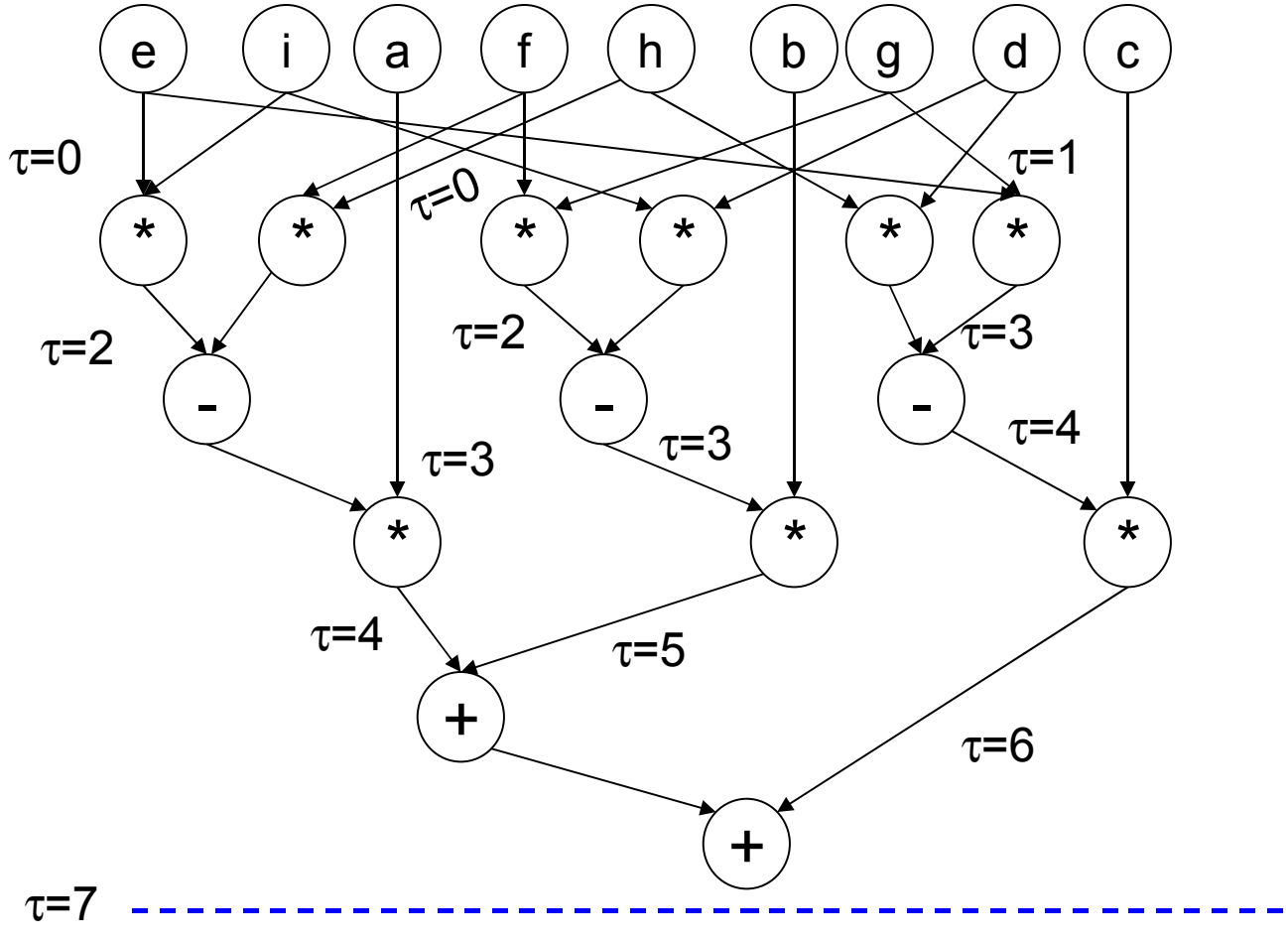
Unterschiedliche Ausführungsgeschwindigkeiten

Determinante

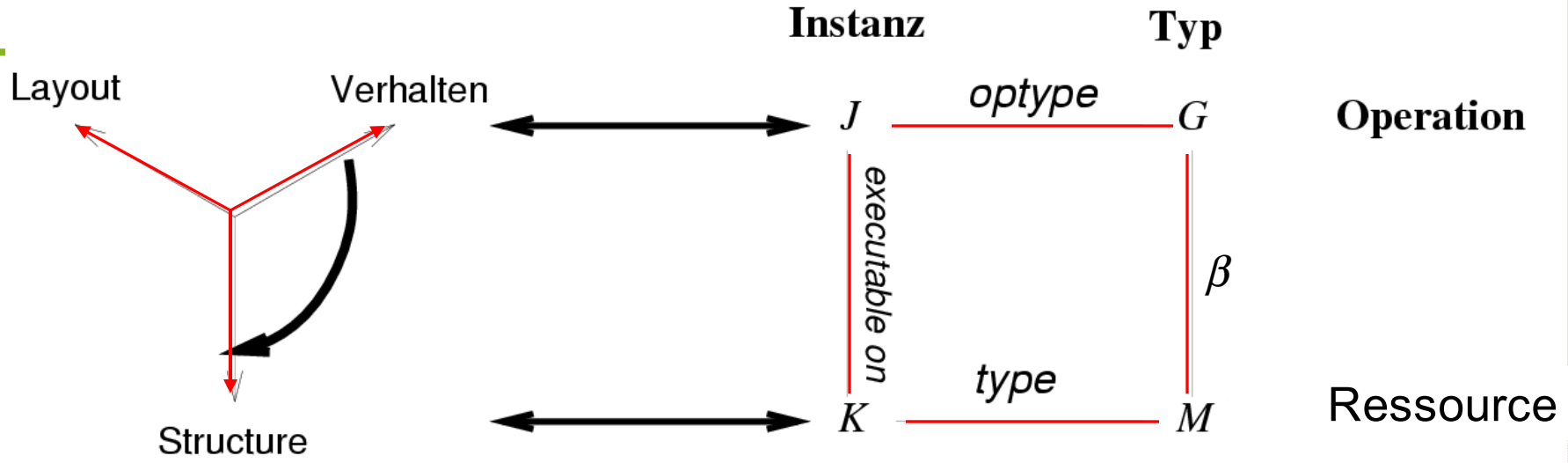


Unterschiedliche Ausführungsgeschwindigkeiten

Determinante



Darstellung von Ressourcen

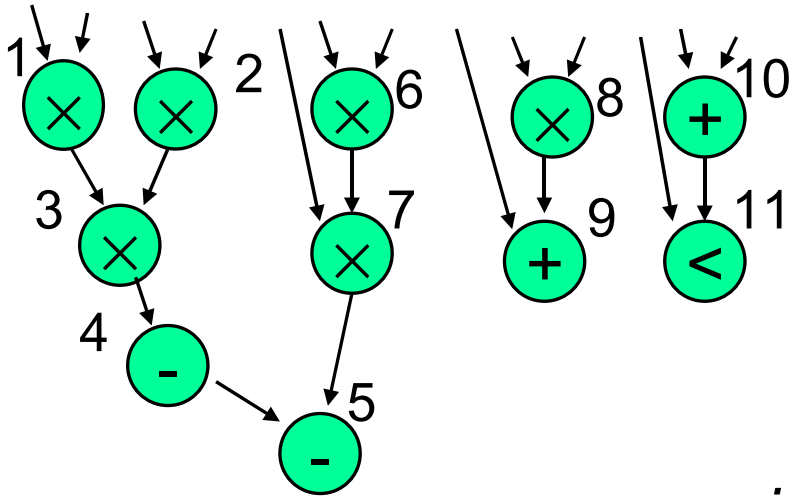


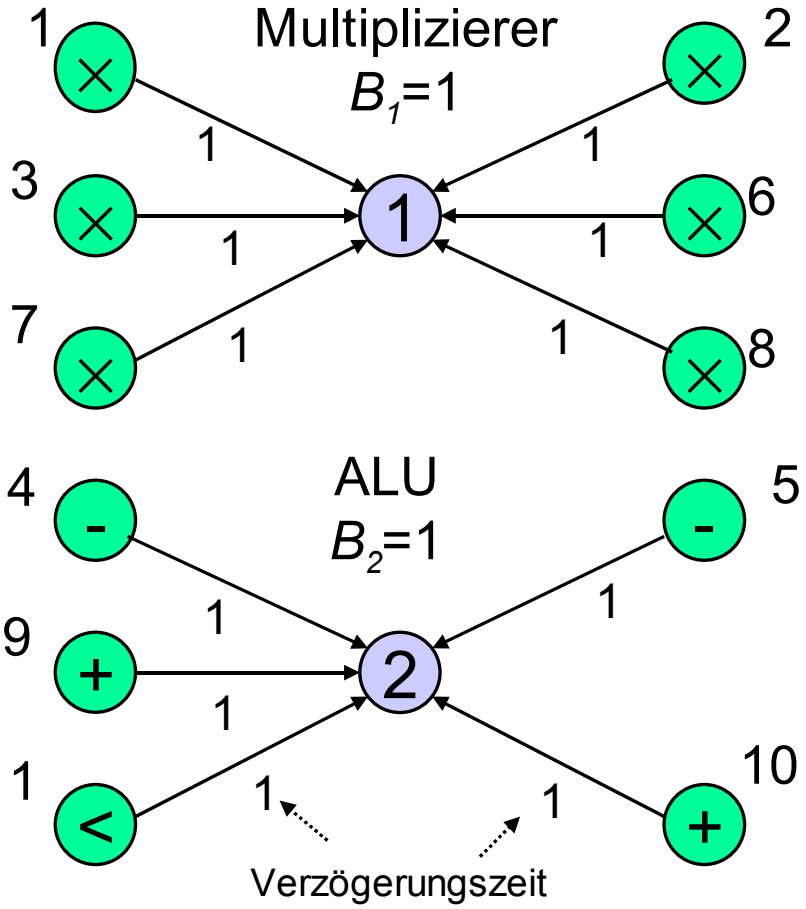
- *optype*: Typ einer Operation im DFG.
- *type*: $K \rightarrow M$: Typ eines Ressourcenexemplars.
- $\beta(m)$: $M \rightarrow \wp(G)$: Funktionalität einer Ressource:
  $\forall m \in M, g \in G: g \in \beta(m) \Leftrightarrow m$ kann Operation g ausführen.
- $j \in J$ executable_on $k \in K \Leftrightarrow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

Ressourcengraph $G_R=(V_R,E_R)$

Darstellung von

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





Nach J. Teich.

List-Scheduling

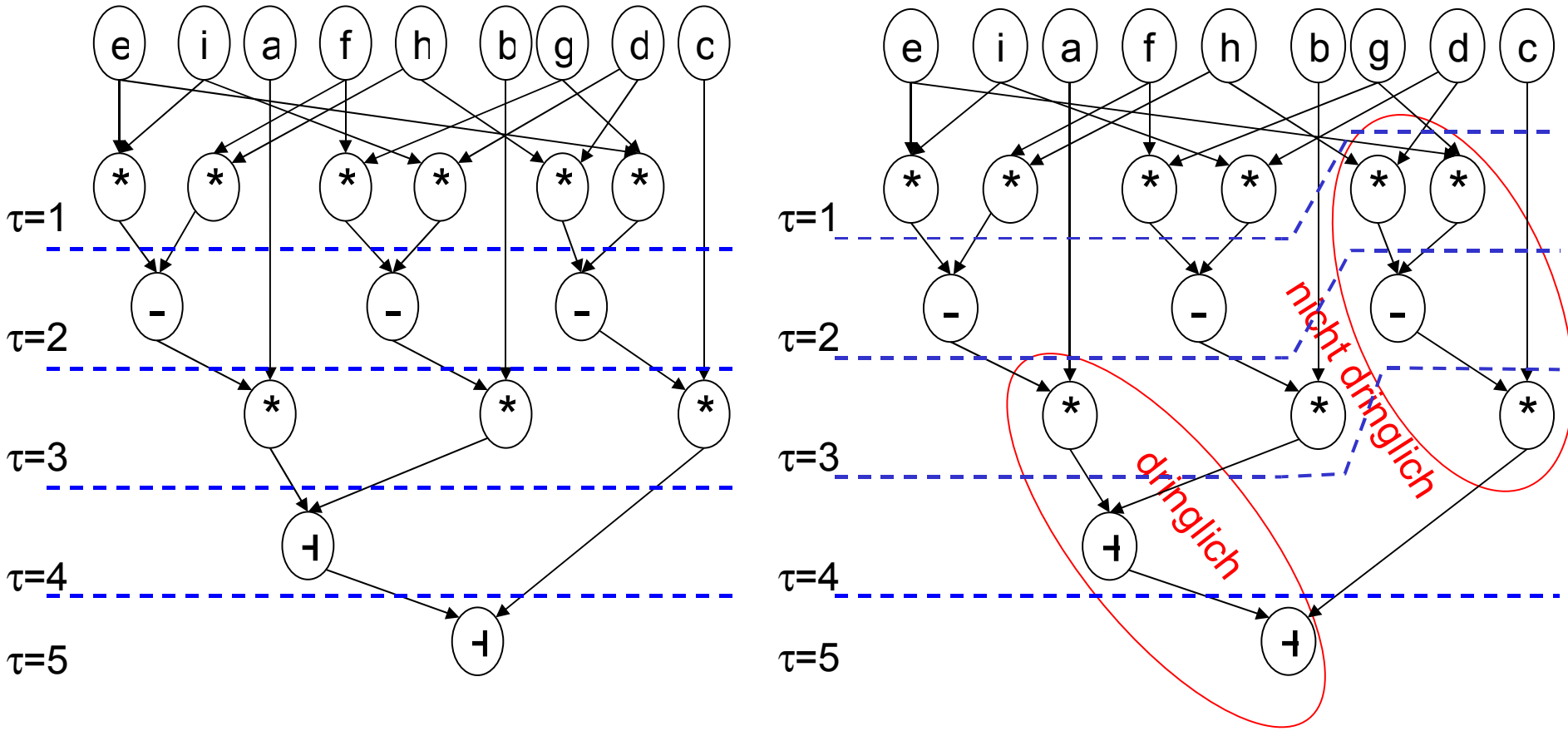
Listscheduling ist eine Weiterentwicklung der ASAP/ALAP-Verfahren.

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 Berechungen

- Wiederholte Berechnung der Menge der auf einem Bausteintyp *m* ausführungsbereiten Operationen, deren Vorgänger alle bekannt sind:
  $A_{i,m}=\{v_j \in V: type(v_j) \in \beta(m) \wedge \forall j': (v_{j'},v_j)\in E: i > \tau(v_{j'})+\ell(j')-1\}$
- Menge der Operationen des Typs *m*, die im Kontrollschritt *i* noch ausgeführt werden:
  $G_{i,m} = \{v_j \in V: type(v_j) \in \beta(m) \wedge \tau(v_j)+\ell(j)-1 \geq i \geq \tau(v_j)\}$
- 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, p, ℓ){
i:=0;
repeat {
for (m=1) **to** Anzahl Ressourcentypen {
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*.

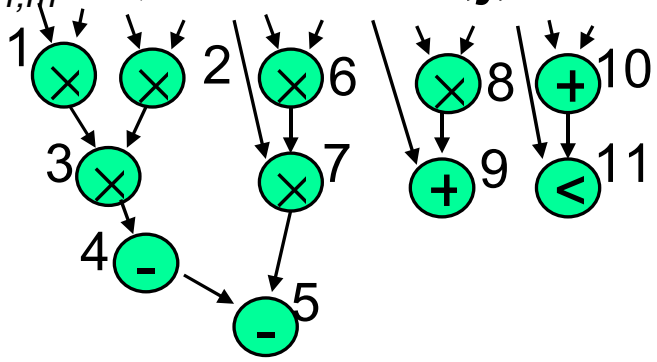
$p(v_1)=p(v_2)=4$

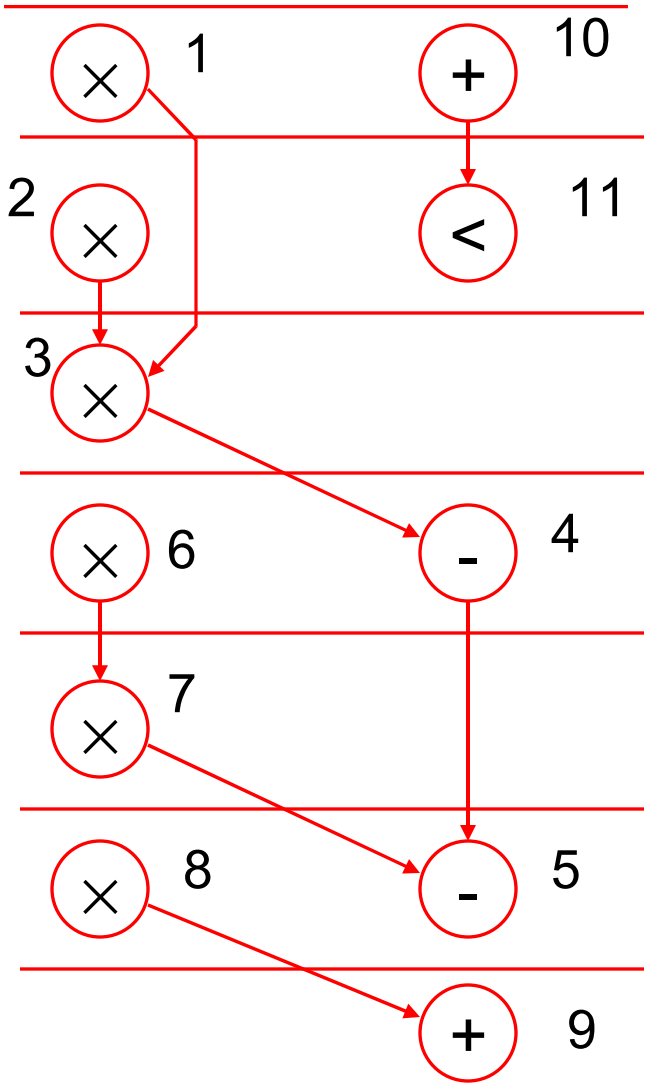
$p(v_3)=p(v_6)=3$

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

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

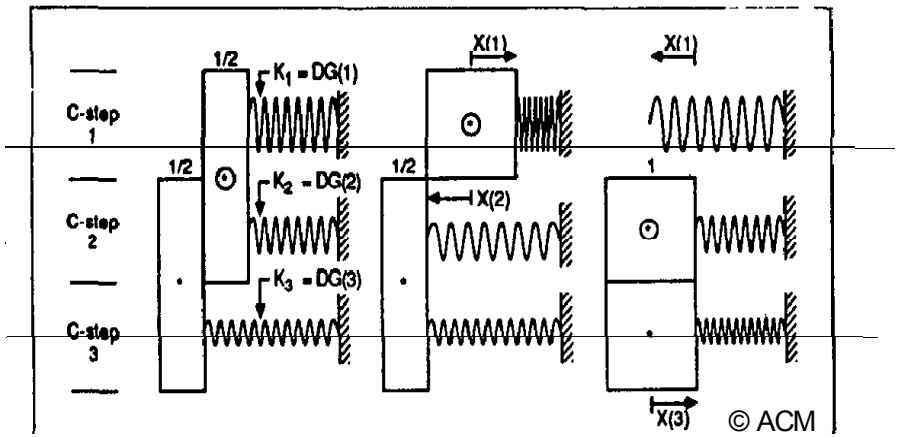
$G_{i,m}=0$, weil $\ell=1$ $\forall i,j,m$.





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

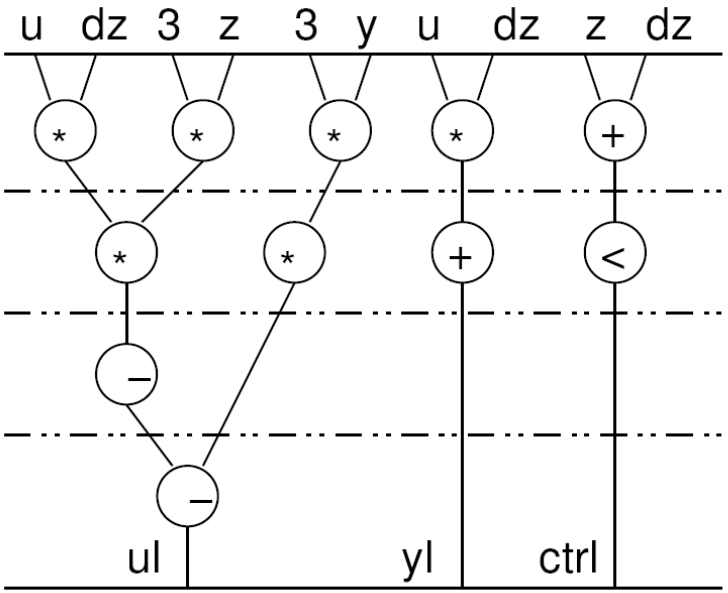
kann durch den folgenden Algorithmus gelöst werden:

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

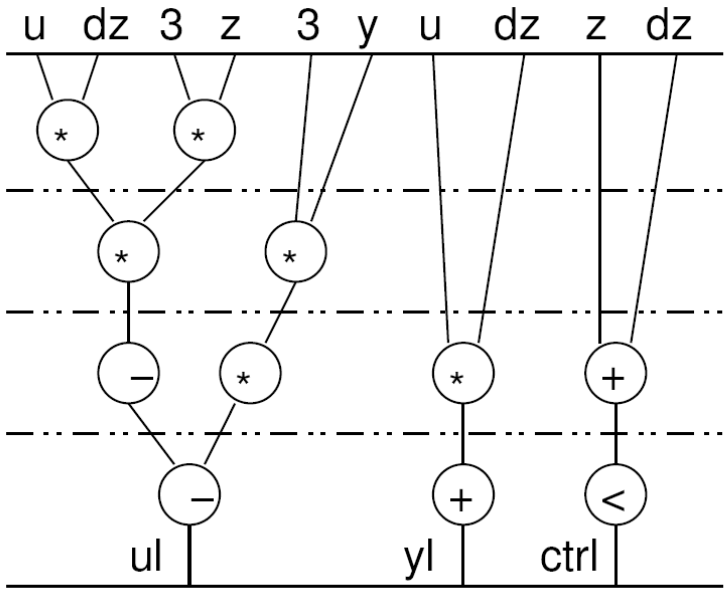
ASAP- und ALAP-*schedules* von *diffeq*

```
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 Ressourcentypen in Frage kommen:

$$\forall\, g \in H_p\ \forall\, g' \in H_p : \{m \mid g \in \beta(m)\} \cap \{m' \mid g' \in \beta(m')\} = \varnothing$$

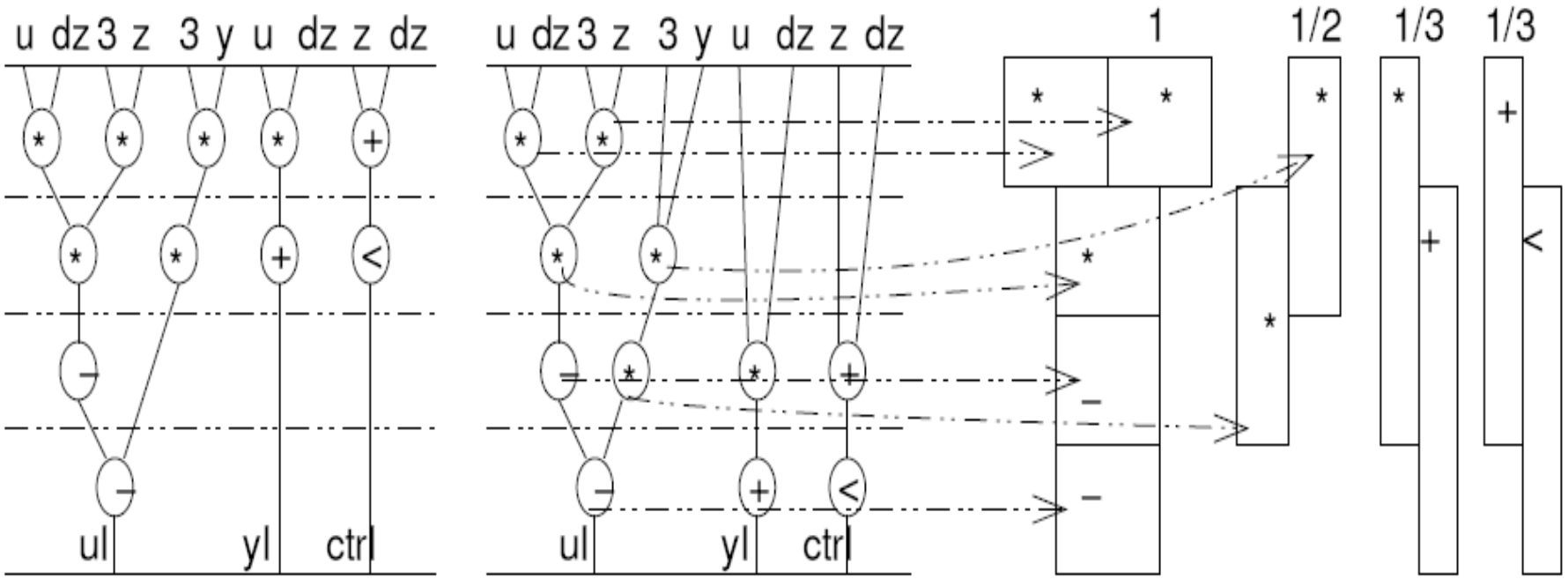
Beispiel:

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

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 $j \rightarrow i$

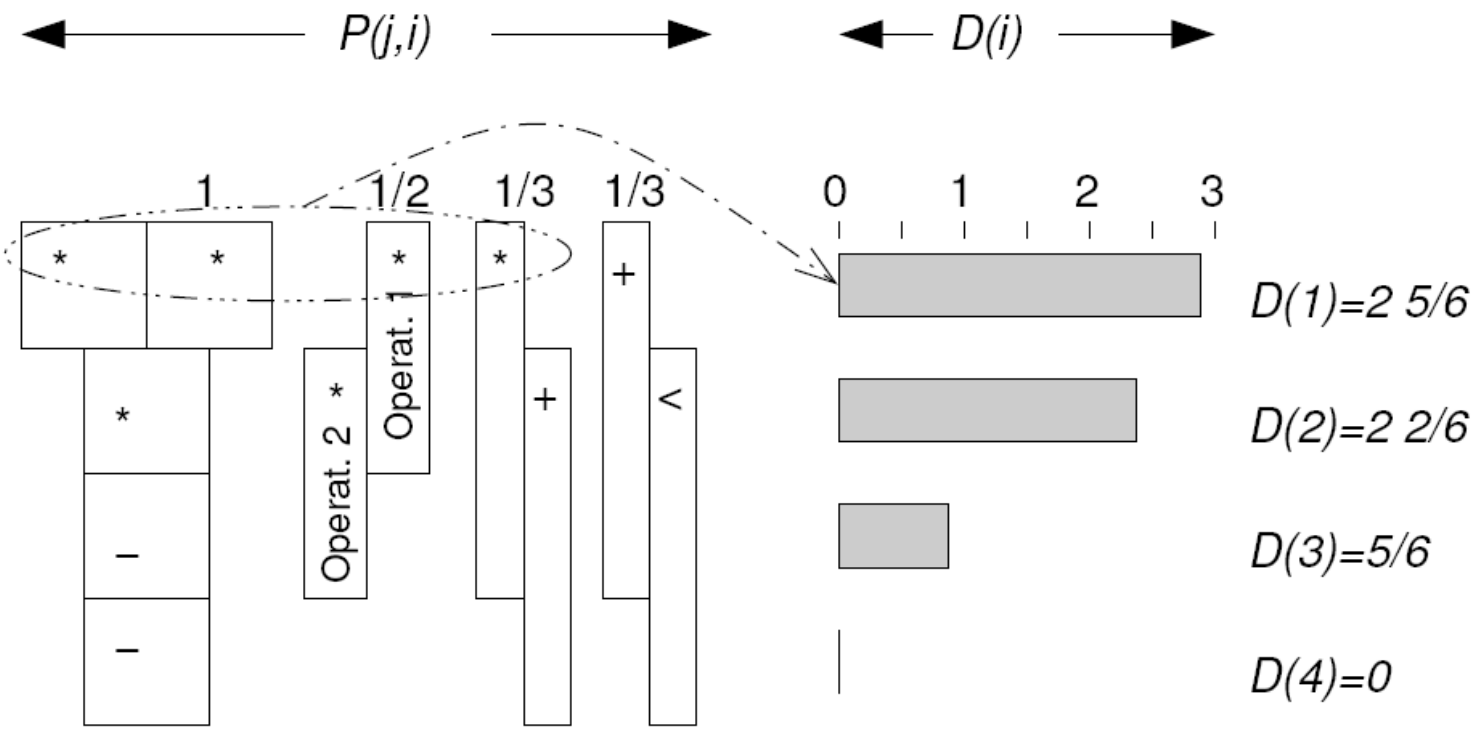


$R(j)$={ASAP-Kontrollschritt .. ALAP-Kontrollschritt}

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

3. Bestimmung einer „Verteilung" *D(i)* (Anzahl von Operationen aus *H* im Kontrollschritt *i*)

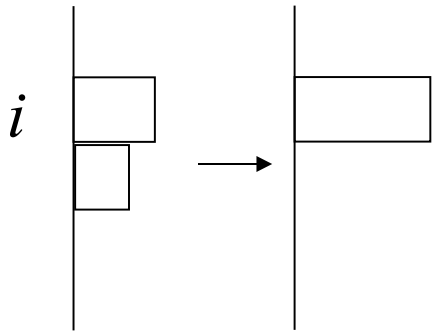
$$D(i) = \sum_{j, type(j) \in H} P(j,i)$$



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 $P(j, i)$.

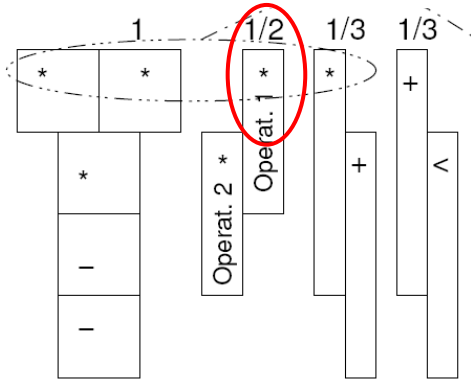
☞ $$\Delta P_i(j, i') = \begin{cases} 1 - P(j, i) \text{ falls } i = i' \\ -P(j, i') \text{ sonst} \end{cases}$$

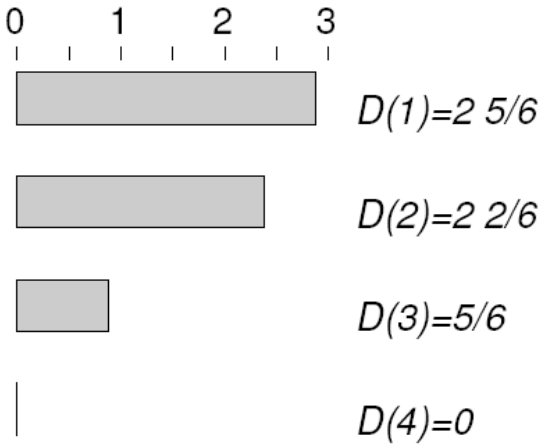
4. Berechnung von direkten Kräften (2)

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

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

Beispiel



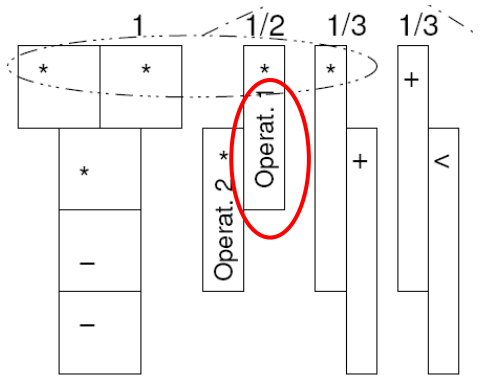


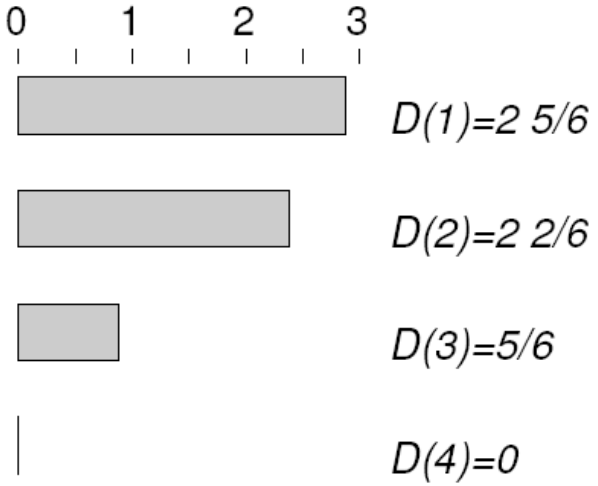
SF(1,1)=2 5/6 (1–1/2) – 2 2/6 (1/2)=

1/2 (17/6–14/6) = 1/2 (3/6) = 1/4

4. Berechnung von direkten Kräften (3)

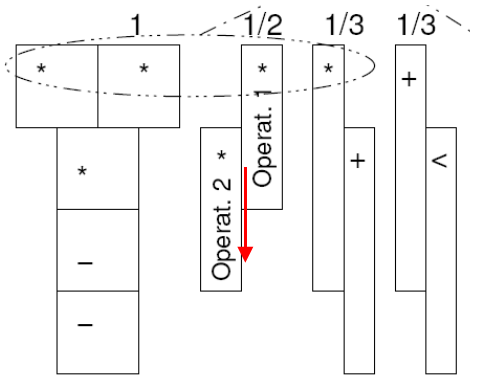
Berechnung der direkten Kraft für die Zuordnung von Operation 1 in Kontrollschritt 2.





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

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

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

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

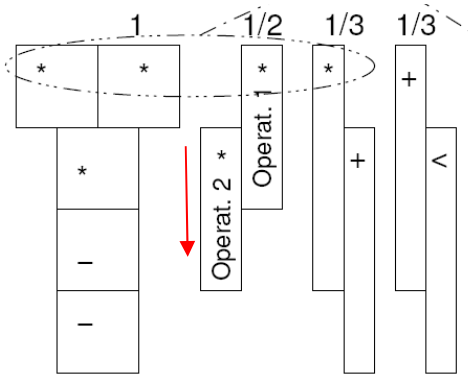
$\Delta P_{j,\,i}(j',i')$ ist die Änderung der Wahrscheinlichkeit der Zuordung von j' zu i' aufgrund der Zuordnung von j zu i

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:

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

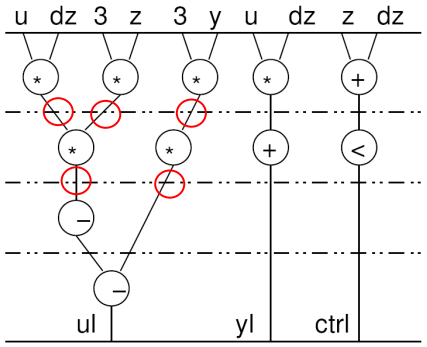
Der niedrige Wert lässt die Zuordnung von Operation 1 zu CS 2 sehr vorteilhaft erscheinen.

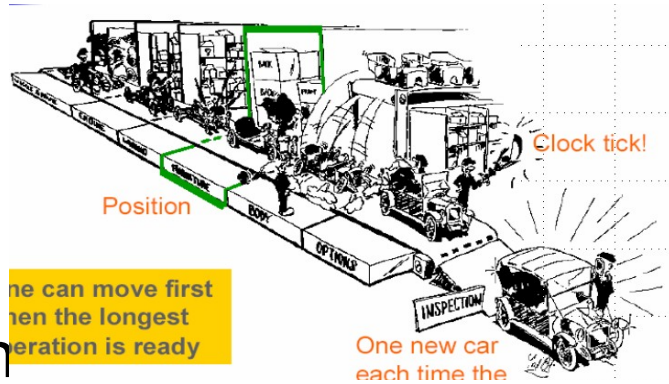
Gesamtablauf

```
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/dsi/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.

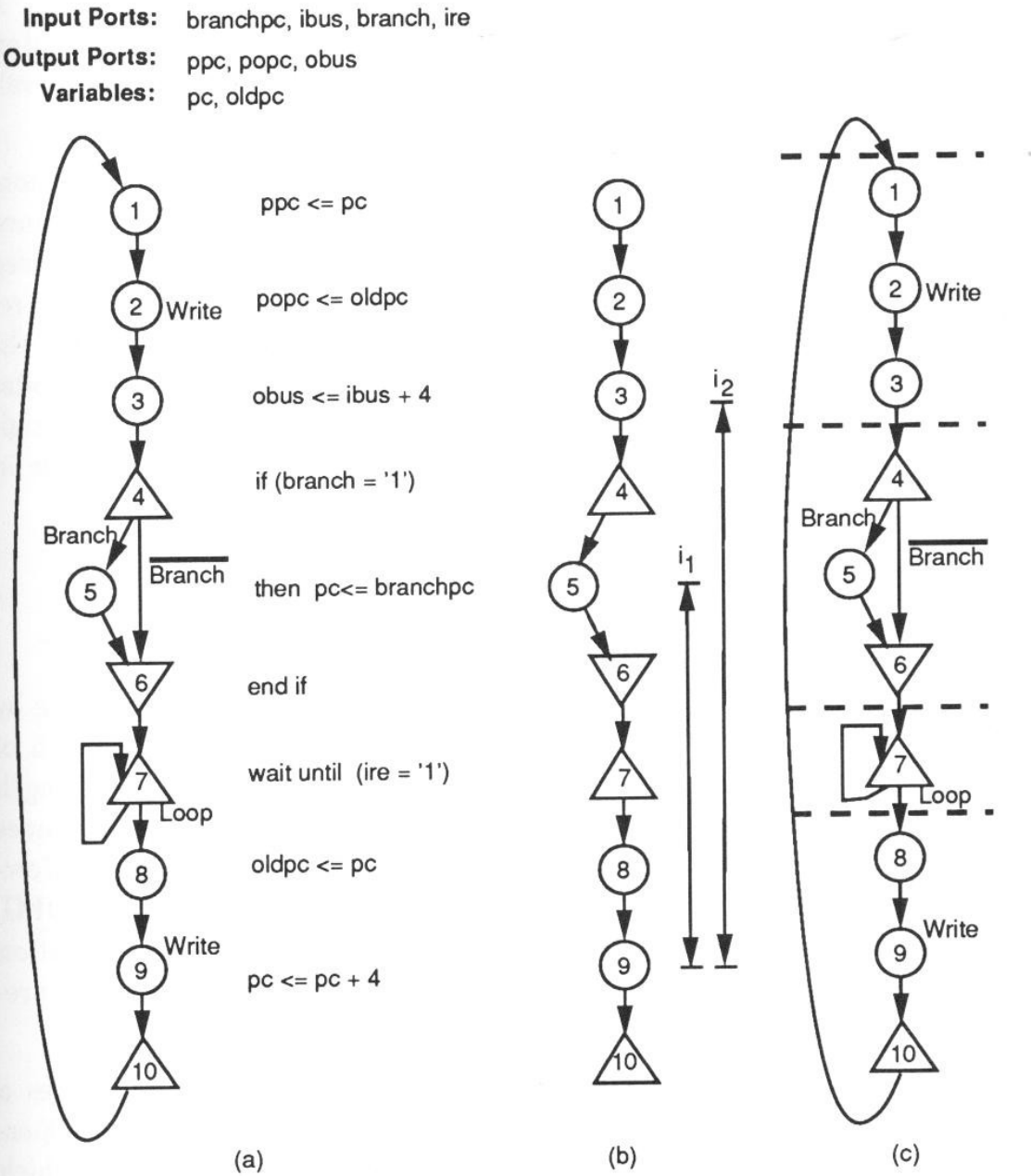


Figure 7.14: Path-based scheduling: (a) an example CDFG, (b) a path in the CDFG with constraint intervals, (c) scheduled CDFG.

Bewertung der Scheduling-Verfahren

- Erlaubt Behandlung heterogener Ressourcen bzw. Multiprozessorsysteme
- Schwerpunkt auf der Berücksichtigung von Abhängigkeiten
- Primär Betrachtung diskreter Zeitabschnitte
- Variable Ausführungszeiten häufig nur als Erweiterung
- Kaum Nutzung globaler Kenntnisse zu Perioden usw.
- V.a. Heuristiken, wenige Aussagen zur Optimalität

Zusammenfassung

- Analogie Multiprozessor-Scheduling
  $\leftrightarrow$ Scheduling in Mikroarchitektur-Synthese
- Verfahren
  - ASAP
  - ALAP
  - *List scheduling* (LS)
  - *Force-directed scheduling* (FDS)
- Bewertung