Embedded Systems & Applications

Algorithmen im Chip-Entwurf

Ablaufplanung

Andreas Koch

FG Eingebettete Systeme und ihre Anwendungen
Informatik, TU Darmstadt

Wintersemester 2010/11

Gliederung

Einführung

Schaltungsmodell
- Sequenzgraph $G_S(V, E)$ (hier: flach!)
- Taktperiode
- Ressourcentyp von v_i ist $T(v_i)$
- Operationsverzögerungen $d_i = d(T(v_i))$ in Takten

Ablaufplanung
- Bestimmt Startzeitpunkte der Operatoren
- Erfüllt Zeit- und Flächenbeschränkungen

Ziel
- Abstimmung von Zeit- und Flächenbedarf (*trade-off*)

Allgemein

Ablaufplan

Funktion $\varphi : V \rightarrow \mathbf{N}$, mit $\varphi(v_i) = t_i$, so dass

$$\forall (v_i, v_j) \in E : t_j \geq t_i + d_i,$$

Feinere Differenzierung: Ablaufplan . . .

. . . ohne Ressourcenbeschränkung und minimaler Latenz

Ablaufplan mit minimalem t_n.

. . . mit Ressourcenbeschränkungen und minimaler Latenz

Zusätzlich muss für alle Ressourcetypen $k = 1, 2, \cdots, n_{res}$ und Ausführungsschritte $l = 1, 2, \cdots, t_n$ gelten:

$$|\{v_i : T(v_i) = k \wedge t_i \leq l < t_i + d_i\}| \leq a_k$$

Hinweis: Hier vereinfachtes Flächenmodell, a_k sind die maximalen Anzahlen von Ressourcen des Typs k.

Ablaufplanung ohne Ressourcenbeschränkung

- Dedizierte Ressource für jeden Operator
  - Paradigma der räumlich verteilten Berechnungen
  - Bindung hat vor Ablaufplanung stattgefunden

- Nützlich zur Bestimmung von Latenzuntergrenzen
  - Bei Planung mit beschränkten Ressourcen
  - Kann nicht besser werden als im dedizierten Fall

ASAP Algorithmus

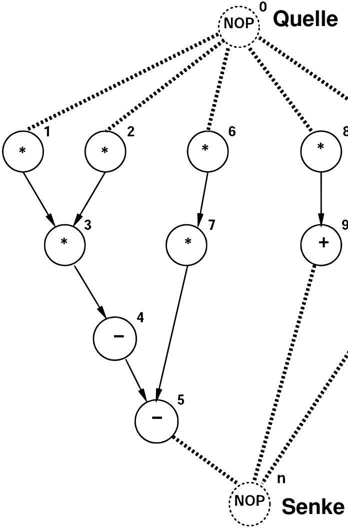
As Soon As Possible - „So früh wie möglich“

ASAP$(G_S(V,E))$

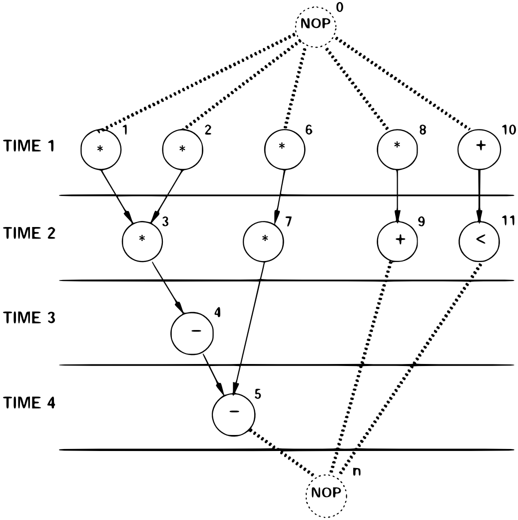
```
Starte $v_0$ bei $t_0^S = 1$;
repeat
        Wähle $v_i$ dessen Vorgänger alle schon geplant sind
        Starte $v_i$ bei $t_i^S = \max_{(v_j,v_i)\in E}\ t_j^S + d_j$
  until $v_n$ ist geplant
```

Beispiel Sequenzgraph

…nur zur Erinnerung



Beispiel: ASAP Ablaufplan



Ablaufplanung mit Latenzbeschränkung

…aber immer noch ohne Ressourcenbeschränkung!

- Maximale Latenz ist $\bar{\lambda}$
- Existenz eines gültigen Ablaufplans testbar mit ASAP
  - …dann muss gelten $t_n^S - t_0^S \leq \bar{\lambda}$
- Falls gültig, spätestmögliche Startzeitpunkte bestimmen

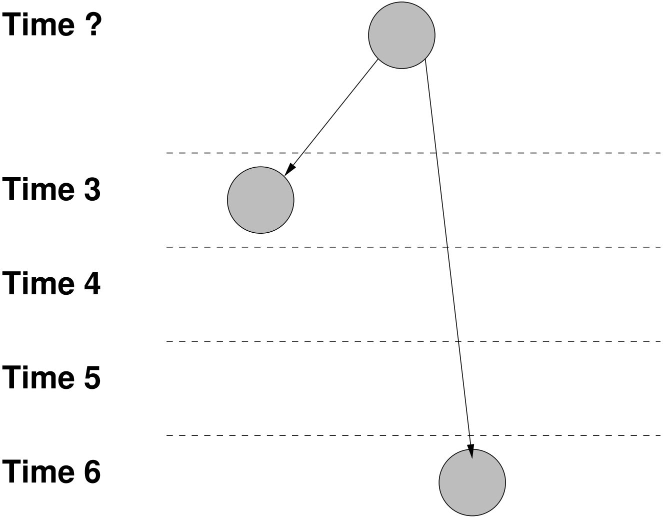
ALAP Algorithmus

As Late As Possible - „So spät wie möglich“

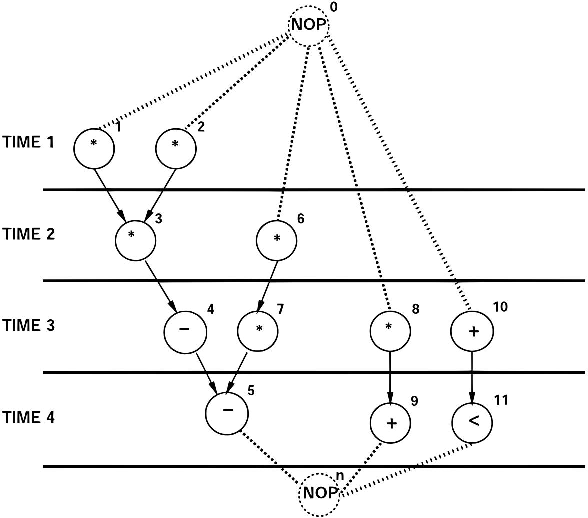
ALAP$(G_S(V,E),\bar{\lambda})$

```
Starte v_n bei t_n^L = λ̄ + 1;
repeat
    Wähle v_i dessen Nachfolger alle schon geplant sind
    Starte v_i bei t_i^L = min_{(v_i,v_j)∈E} t_j^L − d_i
until v_0 ist geplant
```

ALAP Idee



Beispiel: ALAP Ablaufplan



Mobilität von Operationen

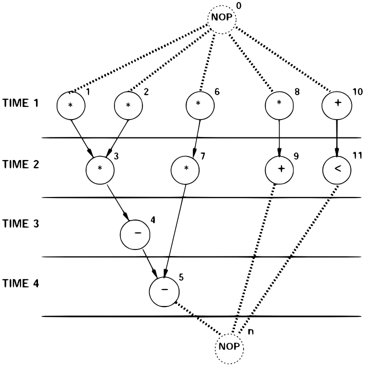
- Mögliche Startzeitpunkte liegen im Intervall $[t_i^S, t_i^L]$
- Mobilität $\mu_i = t_i^L - t_i^S$

  $\mu_i = 0$ Operation v_i kann nur zu einem Zeitpunkt gestartet werden
  - Operation liegt auf kritischem Pfad

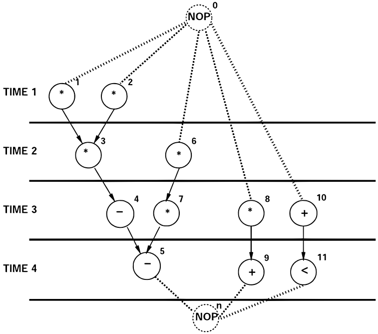
  $\mu_i > 0$ Start von v_i kann beliebig im Intervall geschoben werden

Beispiel: Mobilität

ASAP



ALAP

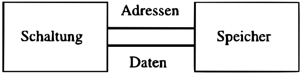


$\mu = 0 : \{v_1, v_2, v_3, v_4, v_5\}$
$\mu = 1 : \{v_6, v_7\}$
$\mu = 2 : \{v_8, v_9, v_{10}, v_{11}\}$

Ablaufplanung mit Zeitbeschränkungen

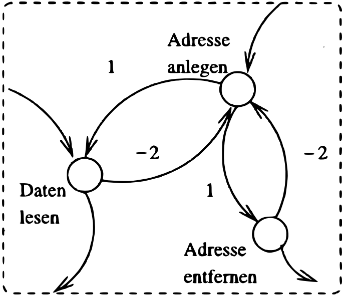
- Häufig durch Restsystem vorgegeben
  - **Absolut**: Spätester (=Deadline) und frühester (=Releasetime) Startzeitpunkt
  - **Relativ**: Zeitliche Relationen zwischen Operatorpaaren
- Absolute sind Spezialfälle von relativen Beschränkungen
  - Werden relativ zum Quellknoten formuliert
- Minimale/maximale Anzahl von Takten zwischen Startzeitpunkten
  - Durch Min=Max auch exakter Zeitpunkt fomulierbar.

Beispiel: Relative Zeitbeschränkungen



Adresse Muss mindestens 1 Takt, darf aber höchstens 2 Takte anliegen

Daten Erscheinen 1 Takt nach Anlegen der Adresse, sind danach 1 Takt lang gültig



Modellierung von Zeitbeschränkungen

Minimale $l_{ij} \geq 0$, mit $t_j \geq t_i + l_{ij}$

Maximale $u_{ij} \geq 0$, mit $t_j \leq t_i + u_{ij}$

Beschränkungsgraph

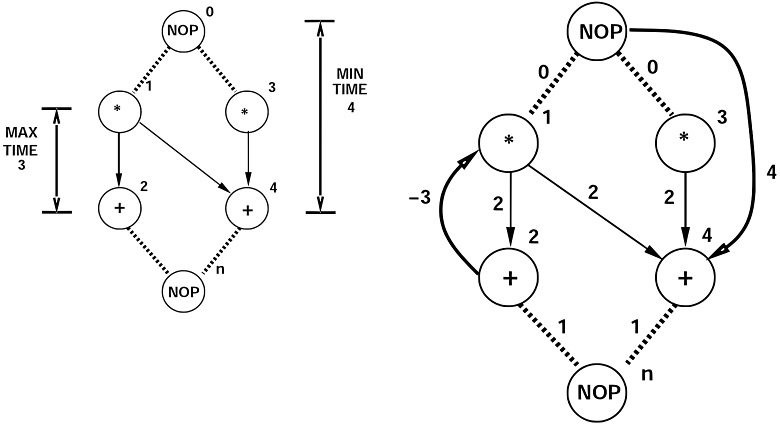
Erweiterung des Sequenzgraphen um Kantengewichte $w(e) \in \mathbf{Z}$ und zusätzliche Kanten e' für Zeitbeschränkungen.

- $w(e) = w((v_i, v_j)) = d_i$
- Neue e' je Zeitbeschränkung zwischen v_i und v_j

  Minimum $e' = (v_i, v_j)$ mit $w(e') = l_{ij}$

  Maximum $e' = (v_j, v_i)$ mit $w(e') = -u_{ij}$

Beispiel: Beschränkungsgraph

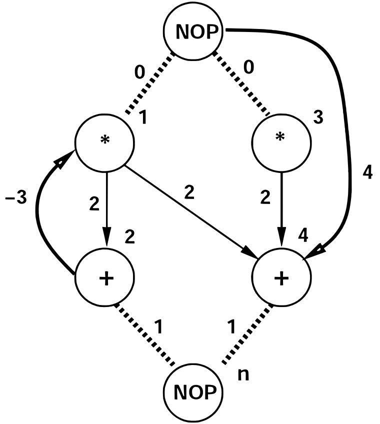


Erfüllbarkeit der Beschränkungen

- Beschränkungen können Ablaufplanung vereiteln
  - Konflikt zwischen u_{ij} und Operationslaufzeiten
  - Konflikt zwischen u_{ij} und l_{ij}

- Test auf Existenz einer gültigen Ablaufplanung
  - Bestimme für jedes u_{ij} den *längsten Pfad* von v_i nach v_j
  - Falls Pfad *länger* als u_{ij} ist, existiert *kein* gültiger Ablaufplan
  - Beschränkungsgraph darf keine *positiven* Zyklen haben
  - Durch Graphenalgorithmen überprüfbar
    - Bellman-Ford, Liao-Wong, etc.
    - Diese liefern auch gleichzeitig die ASAP-Startzeitpunkte

Beispiel: Ablaufplanung mit Zeitbeschränkungen



Knoten	Startzeitpunkt
v_0	1
v_1	1
v_2	3
v_3	1
v_4	5
v_n	6

Ablaufplanung mit Ressourcenbeschränkungen

- Feste Obergrenze für Fläche, minimiere Latenz
  - Flächen werden durch maximale Ressourcenanzahlen a_k beschränkt
- Problem ist $\mathscr{NP}$-hart
- Exakte Lösung mit ganzzahliger linearer Programmierung (ILP)
- Exakte Lösung in $\mathscr{P}$ unter stark eingeschränkten Umständen
- Allgemeinere Heuristiken in $\mathscr{P}$

Grundlagen des ILP-Modells

Allgemeine exakte Lösung des Problems

- Es gibt eine geschätzte Obergrenze für die Latenz $\bar{\lambda}$
  - In der Regel durch Heuristik bestimmt
- Entscheidungsvariablen $x_{il} \in \{0, 1\}$, für alle
  - Operatoren $1 \leq i \leq n_{ops}$
  - Schritte $1 \leq l \leq \bar{\lambda} + 1$
- $x_{il} = 1$ genau dann, wenn $t_i = l$.
  - Alternativ mit Kronecker-Symbol: $x_{il} = \delta_{t_i,l}$
- Aus ASAP/ALAP: $x_{il} = 0$ für $l < t_i^S \lor l > t_i^L$

Formulierung des Programmes 1

Als Ungleichungssystem

1. Jede Operation darf nur einmal gestartet werden

$$\sum_{l=t_i^S}^{t_i^L} x_{il} = 1, \; \forall v_i \in V$$

2. Umrechnung von Entscheidungsvariablen in Startzeitpunkt

$$\sum_{l=t_i^S}^{t_i^L} l \cdot x_{il} = t_i, \; \forall v_i \in V$$

Formulierung des Programmes 2

3. Datenabhängigkeiten einhalten

$$t_i \geq t_j + d_j,\ \forall (v_j, v_i) \in E$$

4. Von jeder Ressource k werden in jedem Zeitschritt l maximal a_k benutzt

$$\sum_{\{i:T(v_i)=k\}} \quad \sum_{m=l-d_i+1}^{l} x_{im} \leq a_k \ ,$$

$$\forall 1 \leq k \leq n_{res}, 1 \leq l \leq \bar{\lambda} + 1$$

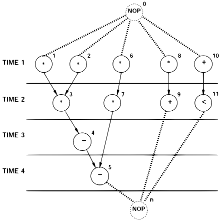
5. Optimierungsziel minimale Latenz: minimiere t_n

Beispiel: ILP - Annahmen

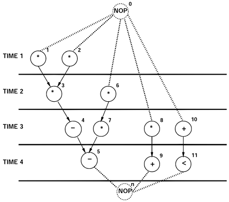
- Sequenzgraph zu `diffeq()`
- $r_1 = *$, $r_2 = +$, $d(r_1) = d(r_2) = 1$
- Ressourcenbeschränkung $a_1 = a_2 = 2$
- Heuristik (kommt später . . .) liefert Obergrenze $\bar{\lambda} = 4$ Schritte
- ASAP/ALAP-Algorithmen aus Problem ohne Ressourcenbeschränkung liefern Startintervalle

Beispiel: ILP - Basisdaten

ASAP



ALAP



Knoten	Intervall
v_0	[1,1]
v_1	[1,1]
v_2	[1,1]
v_3	[2,2]
v_4	[3,3]
v_5	[4,4]
v_6	[1,2]
v_7	[2,3]
v_8	[1,3]
v_9	[2,4]
v_{10}	[1,3]
v_{11}	[2,4]
$v_n = v_{12}$	[5,5]

Beispiel: ILP - Gleichungen 1

Operationen dürfen nur einmal starten

$$
\begin{aligned}
x_{0,1} &= 1 && (1)\\
x_{1,1} &= 1 && (2)\\
x_{2,1} &= 1 && (3)\\
x_{3,2} &= 1 && (4)\\
x_{4,3} &= 1 && (5)\\
x_{5,4} &= 1 && (6)\\
x_{6,1} + x_{6,2} &= 1 && (7)\\
x_{7,2} + x_{7,3} &= 1 && (8)\\
x_{8,1} + x_{8,2} + x_{8,3} &= 1 && (9)\\
x_{9,2} + x_{9,3} + x_{9,4} &= 1 && (10)\\
x_{10,1} + x_{10,2} + x_{10,3} &= 1 && (11)\\
x_{11,2} + x_{11,3} + x_{11,4} &= 1 && (12)\\
x_{n,5} &= 1 && (13)
\end{aligned}
$$

Beispiel: ILP - Gleichungen 2

Datenabhängigkeiten (nur nicht-triviale!)

$$\begin{aligned}
2x_{7,2}+3x_{7,3} &\geq 1x_{6,1}+2x_{6,2}+1 && (14)\\
2x_{9,2}+3x_{9,3}+4x_{9,4} &\geq 1x_{8,1}+2x_{8,2}+3x_{8,3}+1 && (15)\\
2x_{11,2}+3x_{11,3}+4x_{11,4} &\geq 1x_{10,1}+2x_{10,2}+3x_{10,3}+1 && (16)\\
4x_{5,4} &\geq 2x_{7,2}+3x_{7,3}+1 && (17)\\
5x_{n,5} &\geq 2x_{9,2}+3x_{9,3}+4x_{9,4}+1 && (18)\\
5x_{n,5} &\geq 2x_{11,2}+3x_{11,3}+4x_{11,4}+1 && (19)
\end{aligned}$$

Trivial: Beide Operationen haben feste Zeit

$$\begin{aligned}
2x_{3,2} &\geq 1x_{1,1}+1 && (20)\\
2\cdot 1 &\geq 1+1 && (21)\\
2 &\geq 2 && (22)
\end{aligned}$$

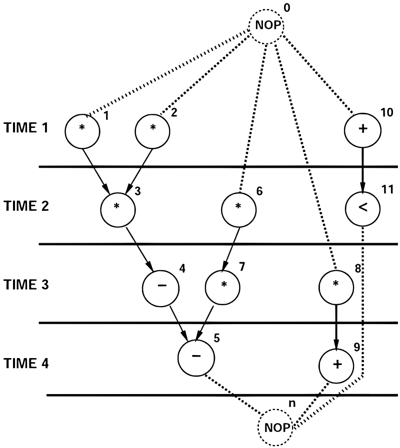
Beispiel: ILP - Gleichungen 3

Ressourcenbeschränkungen

$$
\begin{aligned}
&\text{Multiplizierer} && (23)\\
x_{1,1} + x_{2,1} + x_{6,1} + x_{8,1} &\leq 2 && (24)\\
x_{3,2} + x_{6,2} + x_{7,2} + x_{8,2} &\leq 2 && (25)\\
x_{7,3} + x_{8,3} &\leq 2 && (26)\\
&\text{ALUs} && (27)\\
x_{10,1} &\leq 2 && (28)\\
x_{9,2} + x_{10,2} + x_{11,2} &\leq 2 && (29)\\
x_{4,3} + x_{9,3} + x_{10,3} + x_{11,3} &\leq 2 && (30)\\
x_{5,4} + x_{9,4} + x_{11,4} &\leq 2 && (31)
\end{aligned}
$$

Beispiel: ILP - Lösung mit Solver

Var.	Wert
xn_5	1
x0_1	1
x1_1	1
x2_1	1
x3_2	1
x4_3	1
x5_4	1
x6_1	0
x6_2	1
x7_2	0
x7_3	1
x8_1	0
x8_2	0
x8_3	1
x9_2	0
x9_3	0
x9_4	1
x10_1	1
x10_2	0
x10_3	0
x11_2	1
x11_3	0
x11_4	0



Relative Zeitbeschränkungen im ILP

Können modelliert werden.

Minimale Zeitbeschränkung l_{ij} zwischen v_i und v_j

$$\sum_{l=t_j^S}^{t_j^L} l \cdot x_{jl} \geq \left(\sum_{l=t_i^S}^{t_i^L} l \cdot x_{il}\right) + l_{ij}$$

Maximale Zeitbeschränkung u_{ij} zwischen v_i und v_j

$$\sum_{l=t_j^S}^{t_j^L} l \cdot x_{jl} \leq \left(\sum_{l=t_i^S}^{t_i^L} l \cdot x_{il}\right) + u_{ij}$$

Duales Problem

Minimale Fläche mit Latenzbeschränkung

- Formeln 1,2,3,4 bleiben
  - In 4 sind die a_k jetzt aber freie Variablen
- Zusätzlich

$$\sum_{l=t_n^S}^{\bar{\lambda}+1} l \cdot x_{nl} \leq \bar{\lambda} + 1$$

- Minimiere nun echte Flächen, z.B. bei Fläche(Mult)=5 und Fläche(ALU)=1

$$\textbf{minimiere}: \; 5 \cdot a_1 + 1 \cdot a_2$$

Algorithmus von Hu

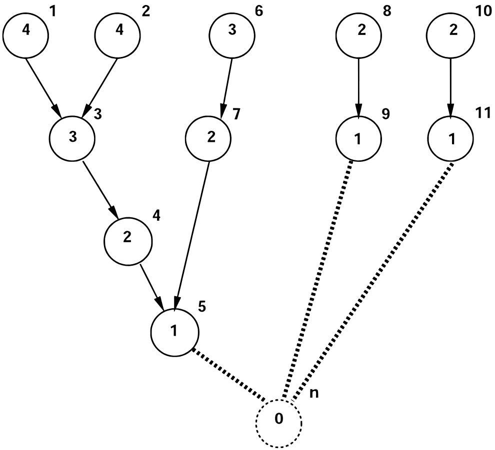
Problem: Lösung von ILPs ist $\mathscr{NP}$-hart
Für eingeschränktere Eingaben aber schneller möglich

- Ignoriert v_0
- G_S ohne v_0 ist ein *Baum*
- Alle Operationen haben denselben Typ
- Es gibt a Ressourcen des Typs, alle mit Verzögerung 1

Vorbereitung:
Beschrifte Knoten $v \in V \setminus \{v_0\}$ mit ihrer Entfernung $p(v) \in N_0$ von der Senke v_n.

Beschriftungsphase von Hus Algorithmus

$p(v)$ wirken als *Priorität*, je größer $p(v)$ desto höher.



Ablauf von Hus Algorithmus

Minimiere Latenz bei Ressourcenbeschränkungen

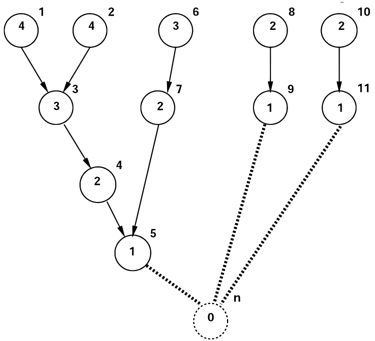
HU$(G_S(V,E),a)$

Beschrifte Knoten $V\setminus\{v_0\}$ mit Priorität;
$l = 1$;
repeat
  U ist Menge aller Knoten ohne Vorgänger
    oder nur mit geplanten Vorgängern;
  Wähle $S \subseteq U$, so dass $|S| \leq a$
    und $\sum_{v\in S}\ p(v)$ maximal;
  Plane Operationen in S bei Schritt l
    durch $t_i = l\ \forall v_i \in S$;
  $l = l + 1$;
until v_n ist geplant;

Vorgehen: Greedy-Schema

Beispiel: Hus Algorithmus

Minimiere Latenz mit $a = 3$ Ressourcen



1. $S = \{v_1, v_2, v_6\}$
2. $S = \{v_3, v_7, v_8\}$
3. $S = \{v_4, v_9, v_{10}\}$
4. $S = \{v_5, v_{11}\}$

Optimales Ergebnis in $O(|V|)$.

Listen-basierte Ablaufplanung

- Familie von Heuristiken in $\mathscr{P}$
- Auch bei Sequenz*graphen*, mehreren Ressourcetypen und längeren Ausführungszeiten
  - Minimiere Latenz bei Ressourcenbeschränkungen
  - Minimiere Ressourcen bei Latenzbeschränkungen
- Erweitern Hus Algorithmus
- Erreichen aber nicht immer das Optimum

Algorithmenskelett bei Ressourcenbeschränkung

LISTSKEL$(G_S(V,E),\mathbf{a})$

$l = 1$;
repeat
    for Ressource $k \in \{1,\cdots,n_{res}\}$ **do**
        Bestimme Kandidaten
        $U_{l,k} = \{v_i \in V : T(v_i) = k \wedge t_j + d_j \leq l \, \forall (v_j, v_i) \in E\}$
        Bestimme nicht-beendete Operationen
        $T_{l,k} = \{v_i \in V : T(v_i) = k \wedge t_i + d_i > l\}$;
        Wähle $S_k \subseteq U_{l,k}$, so dass $|S_k| + |T_{l,k}| \leq a_k$;
        Plane Operationen in S_k bei Schritt l
        durch $t_i = l \, \forall v_i \in S_k$;

    $l = l + 1$;
until v_n ist geplant;

Listen-basierte Ablaufplanung zur Latenzminimierung

- LISTSKEL hat $O(|V|)$ und beachtet bereits Ressourcenbeschränkung
- Versucht aber nicht, die Latenz zu minimieren
- Fehlt: Beachtung der Dringlichkeit von Operationen
- Eine Lösung: LISTMINLAT$(G_S(V,E),\mathbf{a})$
    - Gleicher Aufbau wie LISTSKEL
    - Zeile 6: Knoten nach absteigender Entfernung zur Senke wählen
- Bei $n_{res} = 1$ und $d(r_1) = 1$: Identisch zu Hus Algorithmus, wenn Sequenzgraph ein Baum ist.

Umgang mit relativen Zeitbeschränkungen

Listen-basierte Ablaufplanung unterstützt Zeitbeschränkungen

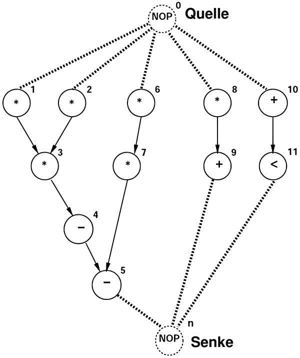
Minimale *Verzögere* die Aufnahme eines Kandidaten v_j nach S_k solange, bis ein l erreicht ist, bei dem alle l_{ij} erfüllt sind.

Maximale Berechne die Priorität eines Kandidaten v_j aus der *Nähe* zu seiner spätesten Ausführungszeit, bestimmt durch das anwachsende l und die u_{ij}.

Beispiel: Listen-basierte Ablaufplanung

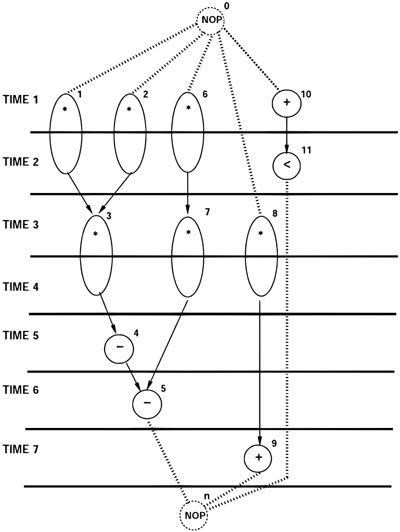
Annahmen

- $a_1 = 3$ Multiplizierer mit $d(r_1) = 2$
- $a_2 = 1$ ALU mit $d(r_2) = 1$
- Priorität entspricht Pfadlänge zur Senke

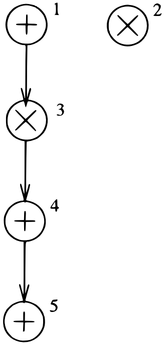


Startzeit	Multiplizierer	ALU
1	$\{v_1, v_2, v_6\}$	v_{10}
2	—	v_{11}
3	$\{v_3, v_7, v_8\}$	—
4	—	—
5	—	v_4
6	—	v_5
7	—	v_9

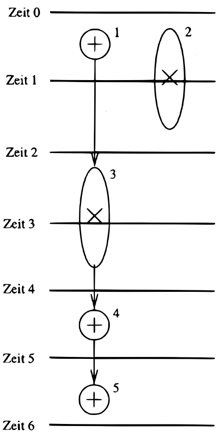
Beispiel: Listen-basierte Ablaufplanung

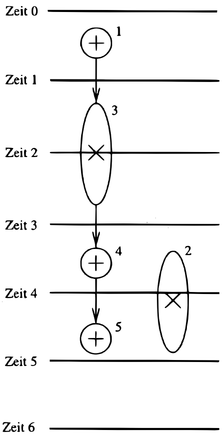


Suboptimalität bei listen-basierter Ablaufplanung



$$a_+ = a_* = 1,$$
$$d(+) = 1, d(*) = 2$$





Listen-basierte Ablaufplanung zur Ressourcenminimierung

…bei Latenzbeschränkung $\bar{\lambda}$

Ideen:

- Beginne mit $a_k = 1$ für alle Ressourcetypen k
- Berechne den Schlupf jedes Operators v_i zur Zeit l als $s_{i,l} = t_i^L - l$
- Wenn $s_{i,l} = 0$, muss der Operator zu diesem Zeitpunkt ausgeführt werden
- Auch, wenn dafür eine zusätzliche Ressource aufgebracht werden muss

Algorithmus LISTMINRES

LISTMINRES$(G_S(V,E),\bar{\lambda})$

$a_k = 1 \;\; \forall k \in \{1,\cdots,n_{res}\}$;
Berechne t_i^L durch ALAP$(G_S(V,E),\bar{\lambda})$;
if $t_0^L \leq 0$ **then**
  return $\emptyset$;

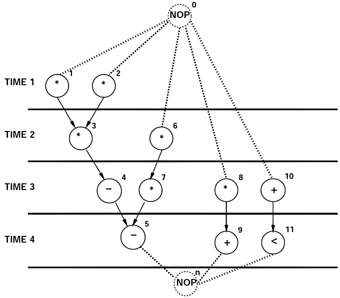
$l = 1$;
repeat
  for Ressource $k \in \{1,\cdots,n_{res}\}$ **do**
    Bestimme Kandidaten
      $U_{l,k} = \{v_i \in V : T(v_i) = k \wedge t_j + d_j \leq l \, \forall (v_j, v_i) \in E\}$;
    Bestimme nicht-beendete Operationen
      $T_{l,k} = \{v_i \in V : T(v_i) = k \wedge t_i + d_i > l\}$;
    Berechne Schlupf $s_{i,l} = t_i^L - l \;\; \forall v_i \in U_{l,k}$;
    Plane Operationen aus $S_{l,k} = \{v_i : s_{i,l} = 0\}$ in Schritt l;
    Setze Ressource auf $a_k = \max(a_k, |S_{l,k}| + |T_{l,k}|)$;
    Plane $a_k - (|S_{l,k}| + |T_{l,k}|)$ weitere Operationen $A_{l,k} \subseteq (U_{l,k} \setminus S_{l,k})$ in l;

  $l = l + 1$;
until v_n ist geplant;

Beispiel: ListMinRes

$d_i = 1$,
$\bar{\lambda} = 4$

ALAP



Schritt	$U_{l,1}$	$S_{l,1}$	$A_{l,1}$	a_1	$U_{l,2}$	$S_{l,2}$	$A_{l,2}$	a_2
1	$\{v_1, v_2, v_6, v_8\}$	$\{v_1, v_2\}$	$\emptyset$	2	$\{v_{10}\}$	$\emptyset$	$\{v_{10}\}$	1
2	$\{v_3, v_6, v_8\}$	$\{v_3, v_6\}$	$\emptyset$	2	$\{v_{11}\}$	$\emptyset$	$\{v_{11}\}$	1
3	$\{v_7, v_8\}$	$\{v_7, v_8\}$	$\emptyset$	2	$\{v_4\}$	$\{v_4\}$	$\emptyset$	1
4	$\emptyset$	$\emptyset$	$\emptyset$	2	$\{v_5, v_9\}$	$\{v_5, v_9\}$	$\emptyset$	2

Verfeinerung: Kräftegesteuerte Ablaufplanung

- Idee: Aktualisiere Prioritäten während des Ablaufs
- Berücksichtige Abhängigkeiten über Datenfluss hinaus
- Führt im allgemeinen zu besseren Ergebnissen
- Kann beide Probleme lösen
  - Minimiere Latenz bei Ressourcenbeschränkungen
  - Minimiere Ressourcen bei Latenzbeschränkungen

- Zunächst einige Definitionen …

Grundlegende Definitionen

1. **Mobilitätsintervall** M_i einer Operation $v_i \in V$ bestimmt via ASAP/ALAP

$$M_i = [t_i^S, t_i^L]$$

2. **Ausführungswahrscheinlichkeit** $p_{i,l}$ einer Operation v_i zum Zeitpunkt l ist

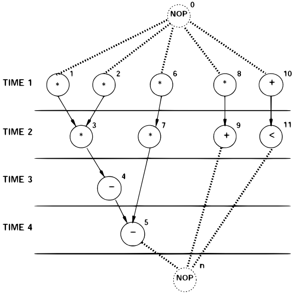
$$p_{i,l} = \begin{cases} \frac{1}{\mu_i + 1} & : \quad \forall l \in M_i \\ 0 & : \quad sonst \end{cases}$$

3. **Belegung** $q_{k,l}$ des Ressourcetyps r_k zum Zeitpunkt l

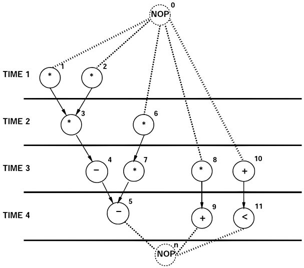
$$q_{k,l} = \sum_{\{v_i : T(v_i) = k\}} p_{i,l}$$

Beispiele für `diffeq()`

ASAP



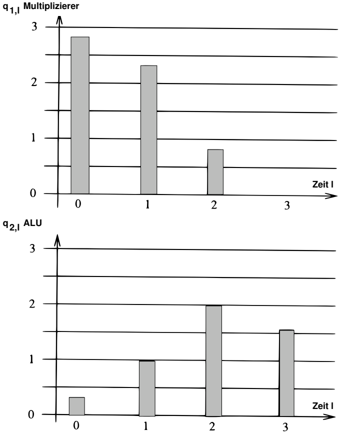
ALAP



$$\bar{\lambda} = 4$$
$$\mu_1 = 0, M_1 = [1,1], p_{1,1} = 1, p_{1,2} = p_{1,3} = p_{1,4} = 0$$
$$\mu_2 = 0, M_2 = [1,1], p_{2,1} = 1, p_{2,2} = p_{2,3} = p_{2,4} = 0$$
$$\mu_6 = 1, M_6 = [1,2], p_{6,1} = 1/2, p_{6,2} = 1/2, p_{6,3} = p_{6,4} = 0$$
$$\mu_8 = 2, M_8 = [1,3], p_{8,1} = 1/3, p_{8,2} = 1/3, p_{8,3} = 1/3, p_{8,4} = 0$$
$$q_{1,1} = 1 + 1 + 1/2 + 1/3 = 2.8\overline{3}$$

Belegungsgraph

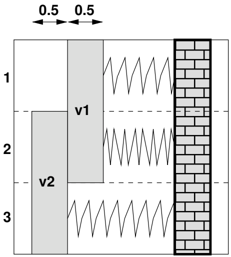
Stellt $q_{k,l}$ für alle Ressourcen k für l auf ganzer Latenz dar



Gleichmäßige Verteilung $\rightarrow$ bessere Auslastung.

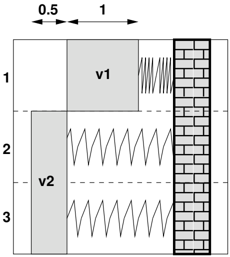
Idee: Mechanisches Modell

Federkraft $F = cx$ (Hookesches Gesetz)



Ungespannter Zustand

$$p_{1,1} = p_{1,2} = 1/2, p_{1,3} = 0$$
$$p_{2,1} = 0, p_{2,2} = p_{2,3} = 1/2$$
$$q_{1,1} = 1/2, q_{1,2} = 1, q_{1,3} = 1/2$$
$$c \approx q$$

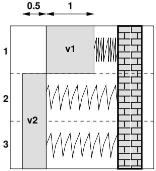


v_1 probeweise in Schritt 1

$x \approx$ Änderung der $p_{i,l}$

$$F^S_{i,l} = \sum_{m=t^S_i}^{t^L_i} q_{T(v_i),m}(\delta_{l,m} - p_{i,m})$$

Selbstkraft



$$
\begin{aligned}
F_{1,1}^{S} &= \sum_{m=t_i^S}^{t_i^L} q_{T(v_i),m}\ \left(\delta_{l,m} - p_{i,m}\right) \\
&= q_{1,1}(1 - p_{1,1}) + q_{1,2}(0 - p_{1,2}) \\
&= 1/2 \cdot (1 - 1/2) + 1 \cdot (0 - 1/2) \\
&= 1/4 - 1/2 = -1/4
\end{aligned}
$$

Umformung Selbstkraftformel

$$
\begin{aligned}
F_{i,l}^{S} &= \sum_{m=t_i^S}^{t_i^L} q_{T(v_i),m} \; (\delta_{l,m} - p_{i,m}) \\
&= \sum_{m=t_i^S}^{t_i^L} q_{T(v_i),m} \; (\delta_{l,m} - \frac{1}{\mu_i + 1}) \\
&= q_{T(v_i),l} - \frac{1}{\mu_i + 1} \sum_{m=t_i^S}^{t_i^L} q_{T(v_i),m}
\end{aligned}
$$

Interpretation: Nach Probeplanung von v_i auf Zeitpunkt l die Änderung zur *durchschnittlichen* Belegung der Ressource k im Mobilitätsintervall von v_i.

Beispiel: Selbstkraft auf v_6 in `diffeq()`

Zur Erinnerung: $M_6 = [1,2]$, $q_{1,1} = 2.8\bar{3}$, $q_{1,2} = 2.\bar{3}$

1. Plane v_6 probeweise auf $l = 1$:

$$F^S_{6,1} = 2.8\bar{3} \cdot (1 - 1/2) + 2.\bar{3} \cdot (0 - 1/2) = 0.25$$

   Interpretation: Über der durchschnittlichen Belegung, höherer Grad an Parallelität und damit Ressourcenbedarf.

2. Plane v_6 probeweise auf $l = 2$

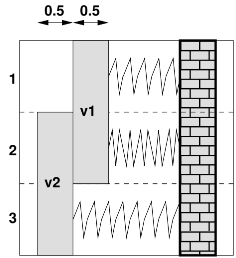
$$F^S_{6,2} = 2.8\bar{3} \cdot (0 - 1/2) + 2.\bar{3} \cdot (1 - 1/2) = -0.25$$

   Interpretation: Unter durchschnittlicher Belegung, braucht nicht mehr Ressourcen.

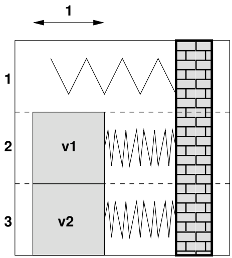
Auswirkungen auf andere Operationen

- Probeweises Planen eines Operators i auf Schritt l schränkt Mobilitätsintervalle seiner Vorgänger und Nachfolger ein
  - Frühester Start von Nachfolger j
    - $\tilde{t}_j^S = \max(t_j^S, t_i + d_i)$
    - $\tilde{M}_j = [\tilde{t}_j^S, t_j^L]$
  - Spätester Start von Vorgänger j
    - $\tilde{t}_j^L = \min(t_j^L, t_i - d_j)$
    - $\tilde{M}_j = [t_j^S, \tilde{t}_j^L]$
- Analog: Berechnung von $\tilde{\mu}_j$
- Modelliere Effekte durch Vorgänger- und Nachfolgerkräfte

Beispiel: Mechanisches Modell



Annahme: $(v_1, v_2) \in E$



v_1 probeweise in Schritt 2
Mobilitätsintervall von v_2 eingeschränkt

$$M_2 = [2, 3] \rightarrow \tilde{M}_2 = [3, 3]$$

$q_{1,3}$ erhöht sich!

Vorgänger-/Nachfolgerkräfte für v_j

…wenn v_i probeweise auf Schritt l geplant ist.

Idee: Berechne die Änderung der *mittleren* Belegung von $T(v_j)$ von M_j zu $\tilde{M}_j$,

$$F^N_{j,l} = \frac{1}{\tilde{\mu}+1}\sum_{m=\tilde{t}^S_j}^{\tilde{t}^L_j} q_{T(v_j),m} \;-\; \frac{1}{\mu_j+1}\sum_{m=t^S_j}^{t^L_j} q_{T(v_j),m}$$

Interpretation: Wie stark ändert sich durch die Probeplanung von v_i die mittlere Nachfrage nach den Ressourcen seiner Vorgänger / Nachfolger?

Beispiel: `diffeq()`

- Probeweise Planung von v_8 auf Schritt 2
- v_9 ist Nachfolger, da $(v_8, v_9) \in E$
- $M_9 = [2,4]$, aber jetzt $t_9^S < t_8 + d_8$
- $\tilde{M}_9 = [3,4]$

Damit

$$\begin{aligned} F_{9,1}^N &= \frac{1}{2}\left(q_{2,3} + q_{2,4}\right) - \frac{1}{3}\left(q_{2,2} + q_{2,3} + q_{2,4}\right) \\ &= 0.5 \cdot (2 + 1.\bar{6}) - 0.\bar{3} \cdot (1 + 2 + 1.\bar{6}) \\ &= 0.2\bar{7} \end{aligned}$$

Die Nachfrage nach Ressource 2 steigt also.

Gesamtkraft auf v_i geplant in Schritt l

Summe von Selbst-, Vorgänger- und Nachfolgerkräften

$$F_{i,l} = F_{i,l}^{S} + \sum_{(v_j,v_i)\in E} F_{j,l}^{N} + \sum_{(v_i,v_j)\in E} F_{j,l}^{N}$$

Beispiel `diffeq()`:

- Probeweise Planung von v_6 in Schritt 2
- Impliziert Planung von v_7 in Schritt 3
- $F_{7,2}^{N} = q_{1,3} - 1/2\,(q_{1,2} + q_{1,3}) = -0.75$
- $F_{6,2} = F_{6,2}^{S} + F_{7,2}^{N} = -0.25 - 0.75 = -1$

Kräftegesteuerte Listenablaufplanung

- Berechnet Ablaufplan mit minimaler Latenz bei beschränkten Ressourcen
- Grobstruktur wie ListSkel, also Vorgehen in Zeitschritten
- Selektion der $S_k \subseteq U_{l,k}$ nun kräftegesteuert
  - Verzögere Operationen mit kleinen $F_{i,l}$ solange, bis a_k eingehalten werden
  - Verzögern geschieht durch Verkürzen der M_i
- Idee: Maximale Parallelität (niedrige Latenz) unter Wahrung der Ressourcenbeschränkungen
- Bei jedem Zeitschritt müssen Kräfte neu berechnet werden, $O(|V|^2)$
- Falls Operationen mit $\mu = 0$ verzögert werden müssen, erhöhe $\bar{\lambda}$ um 1 und berechne damit Kräfte noch ungeplanter Operationen neu

Kräftegesteuerte Ablaufplanung

- Berechnet Plan mit minimalen Ressourcen bei Latenzbeschränkung
- Geht operationsweise vor, $O(|V|^3)$, mit Trick $O(|V|^2)$

FORCEDIRECTED$(G_S(V,E),\bar{\lambda})$

repeat
    Bestimme M_i aller noch nicht geplanten v_i
    Bestimme $p_{i,l}$ und $q_{k,l}$ für alle l und k
    Berechne $F_{i,l}$ aus $F_{i,l}^S$ und $F_{j,l}^N$ für alle i und l
    Plane v_i mit der geringsten Kraft $F_{i,l}$ in Schritt l
until alle Operationen sind geplant

Idee: Minimiere Parallelität (=Ressourcen) bei garantierter Einhaltung der Latenzbeschränkung (alle v_i werden *immer* innerhalb ihrer M_i geplant).