1 Hardwareentwurf

- **1.1** Überblick, Hardwareentwurfsschritte
- **1.2** Hardwarebeschreibungssprachen
- **1.3** Hardwaresimulation-/Verifikation
- **1.4** Hardwaresynthese
  - Highlevel Synthese
  - Gattersynthese
- **1.5** Plazierung und Verdrahtung

Literatur

- G. De Micheli
  Synthesis and Optimization of Digital Circuits
- J. Teich
  Digitale Hardware/Software Systeme
- T. Kropf
  VLSI Entwurf

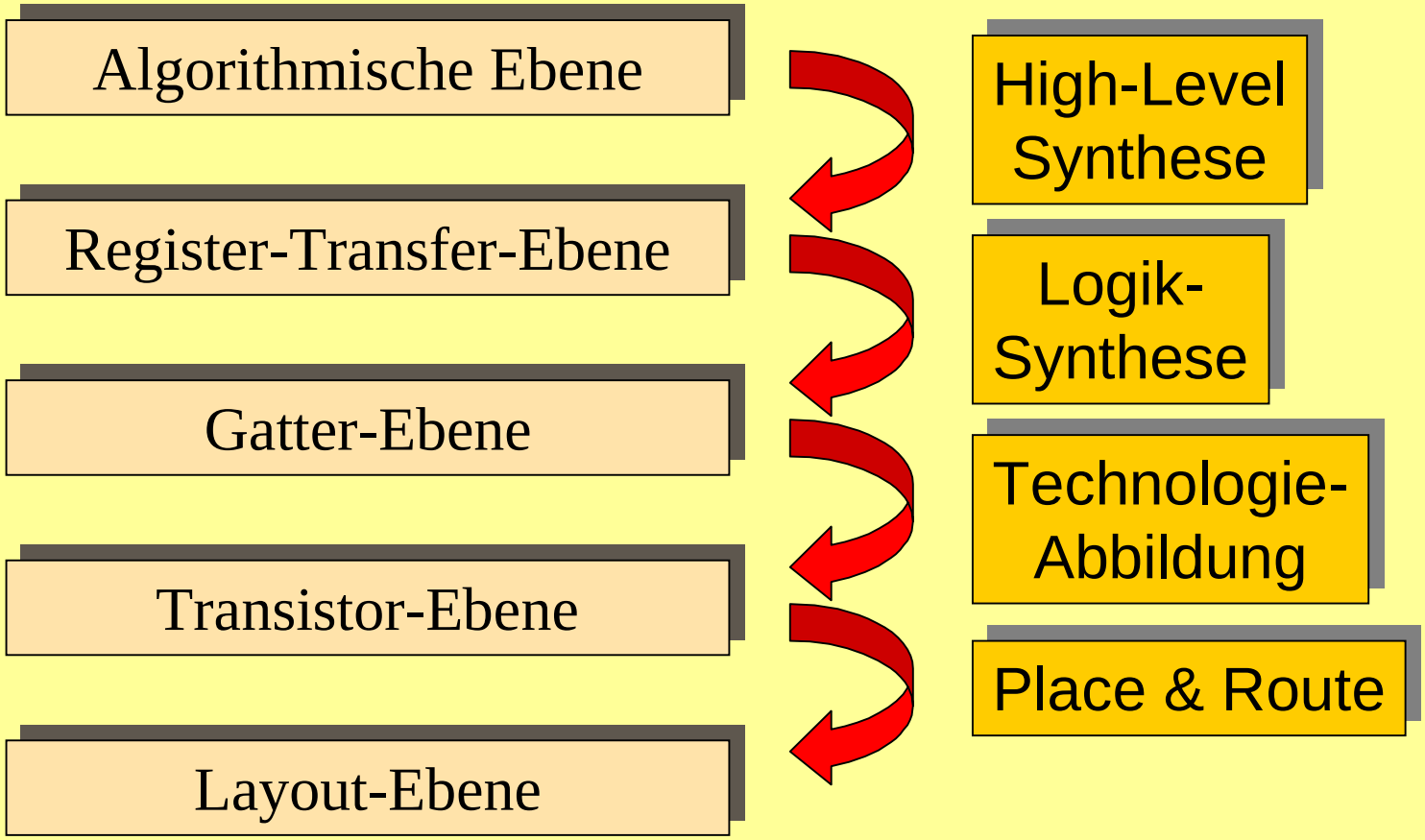
HW-Synthese

Automatische Abbildung von Beschreibungen auf hoher Abstraktionsebenen in niedrige Abstraktionsebenen

- Highlevel-Synthese
  algorithmische Ebene $\Rightarrow$ Registertransfer Ebene

- Gattersynthese
  Registertransfer Evene $\Rightarrow$ Gatterebene

Hardwaresynthese

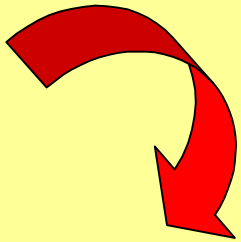


High-Level Synthese

```
int dgl(int x,y,u,dx,a) {
  int x1, y1, u1;
  do {
    x1 = x+dx;
    u1 = u-(3*x*u*dx)-(3*y*dx);
    y1 = y+(u * dx);
    x = x1; u = u1; y = y1;
  while (x1<=a);
  return y;
}
```

High-Level Synthese

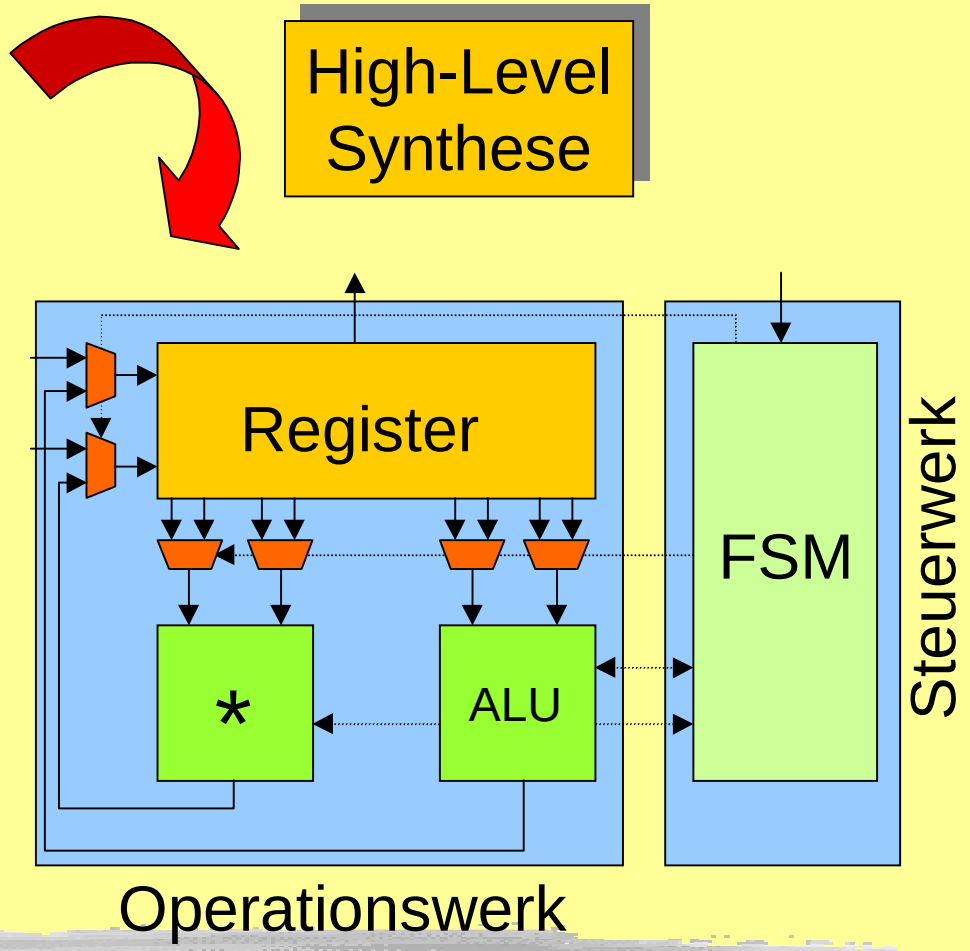
```
int dgl(int x,y,u,dx,a) {
  int x1, y1, u1;
  do {
    x1 = x+dx;
    u1 = u-(3*x*u*dx)-(3*y*dx);
    y1 = y+(u * dx);
    x = x1; u = u1; y = y1;
  while (x1<=a);
  return y;
}
```



High-Level Synthese

High-Level Synthese

```
int dgl(int x,y,u,dx,a) {
  int x1, y1, u1;
  do {
    x1 = x+dx;
    u1 = u-(3*x*u*dx)-(3*y*dx);
    y1 = y+(u * dx);
    x = x1; u = u1; y = y1;
  while (x1<=a);
  return y;
}
```



High-Level Synthese

- Allokation

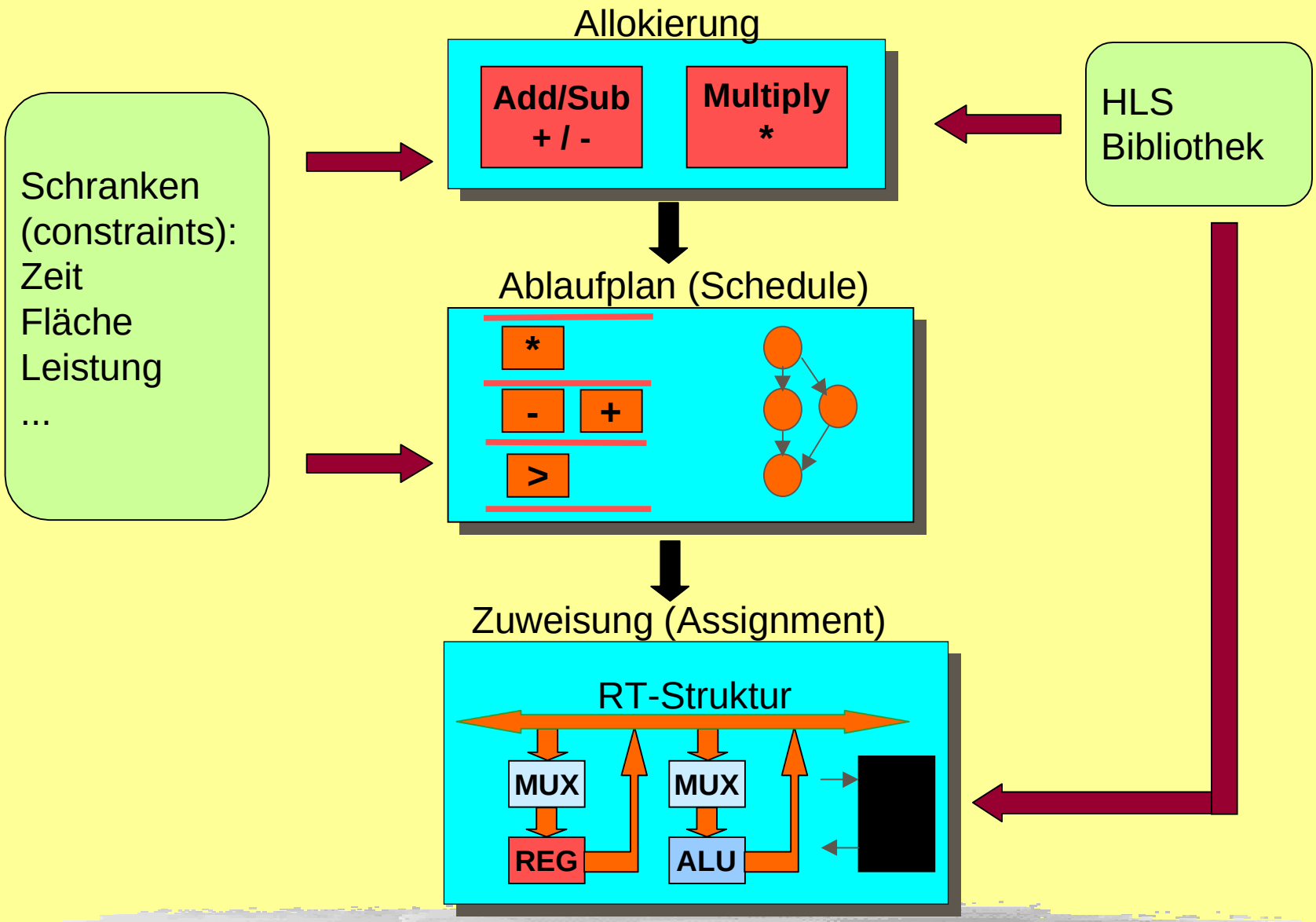
  Auswahl der HW-Komponenten für die Operationen

- Ablauflanung

  Bestimmung der Zeitpunkte an denen die Operationen auszuführen sind

- Bindung

  Zuordnung von Variablen zu Registern, Operationen zu funktionalen Einheiten und Kommunikationskanälen zu Bussen


Allokierung
Add/Sub
+ / -
Multiply
*
HLS
Bibliothek
Schranken
(constraints):
Zeit
Fläche
Leistung
...
Ablaufplan (Schedule)
*
-
+
>
Zuweisung (Assignment)
RT-Struktur
MUX
MUX
REG
ALU

Allokation

- Abbildung von Operatoren der Programmiersprache in HW-Komponenten
- Die HW-Komponenten liegen in generischen Bibliotheken vor, z.B.
  - ALUs mit generischer Bitbreite
  - Multiplizierer mit generischer Bitbreite
  - ...

Allokation

```
int dgl(int x,y,u,dx,a) {
  int x1, y1, u1;
  do {
    x1 = x+dx;
    u1 = u-(3*x*u*dx)-(3*y*dx);
    y1 = y+(u * dx);
    x = x1; u = u1; y = y1;
  while (x1<=a);
  return y;
}
```

Allokation

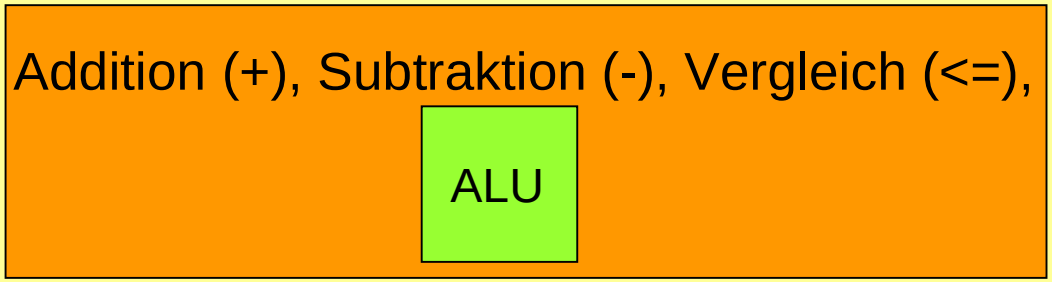
```
int dgl(int x,y,u,dx,a) {
  int x1, y1, u1;
  do {
    x1 = x+dx;
    u1 = u-(3*x*u*dx)-(3*y*dx);
    y1 = y+(u * dx);
    x = x1; u = u1; y = y1;
  while (x1<=a);
  return y;
}
```

⇒ Addition (+), Subtraktion (-), Vergleich (<=), Multiplikation (*)

Allokation

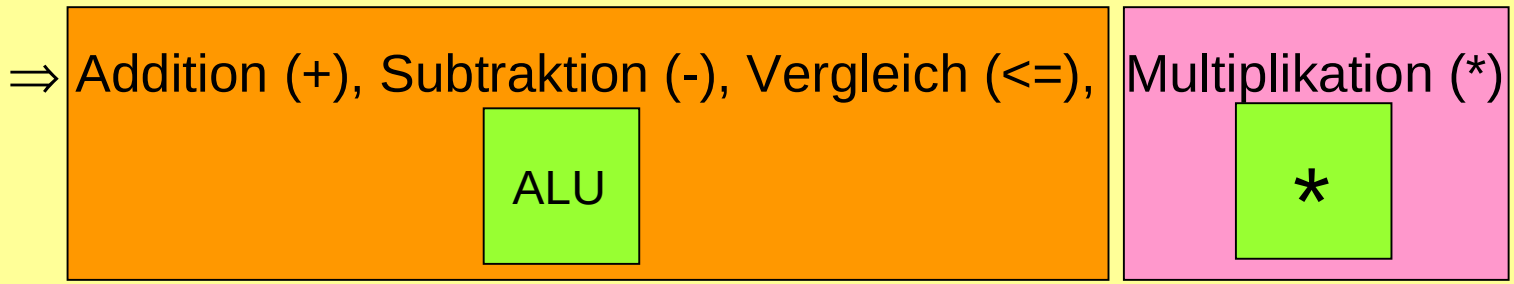
```
int dgl(int x,y,u,dx,a) {
  int x1, y1, u1;
  do {
    x1 = x+dx;
    u1 = u-(3*x*u*dx)-(3*y*dx);
    y1 = y+(u * dx);
    x = x1; u = u1; y = y1;
  while (x1<=a);
  return y;
}
```

⇒ Addition (+), Subtraktion (-), Vergleich (<=), Multiplikation (*)



Allokation

```
int dgl(int x,y,u,dx,a) {
  int x1, y1, u1;
  do {
    x1 = x+dx;
    u1 = u-(3*x*u*dx)-(3*y*dx);
    y1 = y+(u * dx);
    x = x1; u = u1; y = y1;
  while (x1<=a);
  return y;
}
```



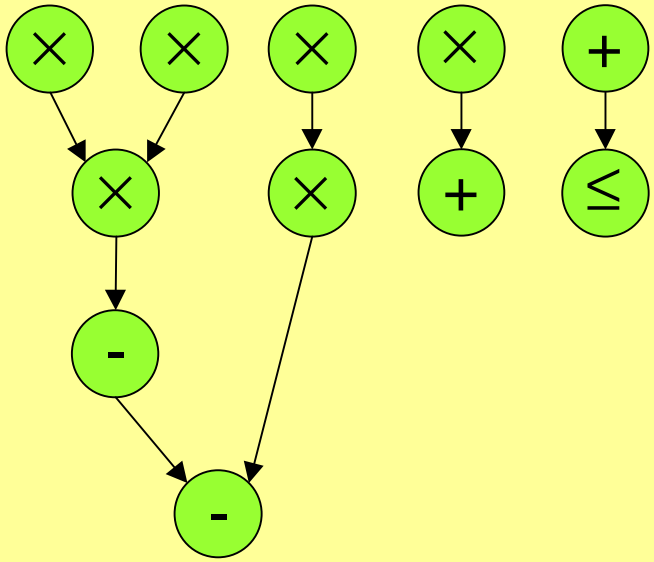
Ablaufplanung

- Bestimmung der Zeitpunkte zu denen die Operationen auszuführen sind unter Berücksichtigung von
  - Resourcenbeschränkungen (z.B. nur ein Multiplizierer)
  - Zeitbeschränkungen (maximale Latenz)

Proplemgraph

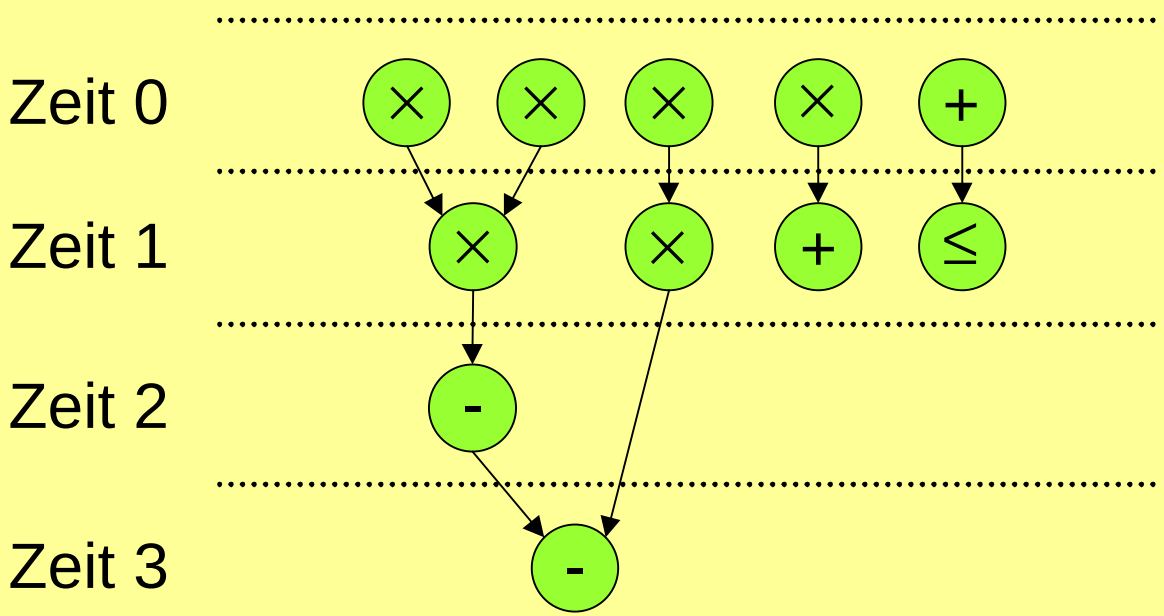
- Operationen
- Datenabhängigkeiten

```
int dgl(int x,y,u,dx,a) {
  int x1, y1, u1;
  do {
    x1 = x+dx;
    u1 = u-(3*x*u*dx)-(3*y*dx);
    y1 = y+(u * dx);
    x = x1; u = u1; y = y1;
  while (x1<=a);
  return y;
}
```



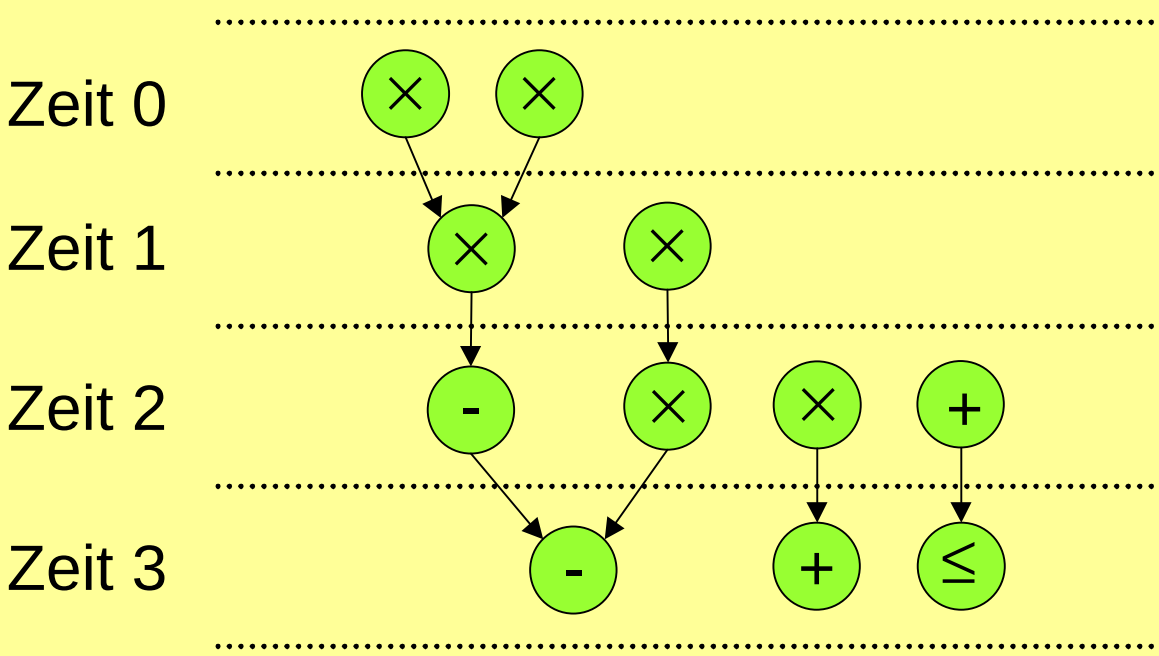
Ablaufplanung

- Ordnet den Knoten im Problemgraph Ausführungszeitpunkte zu
- z.B. ASAP (as soon as possible)



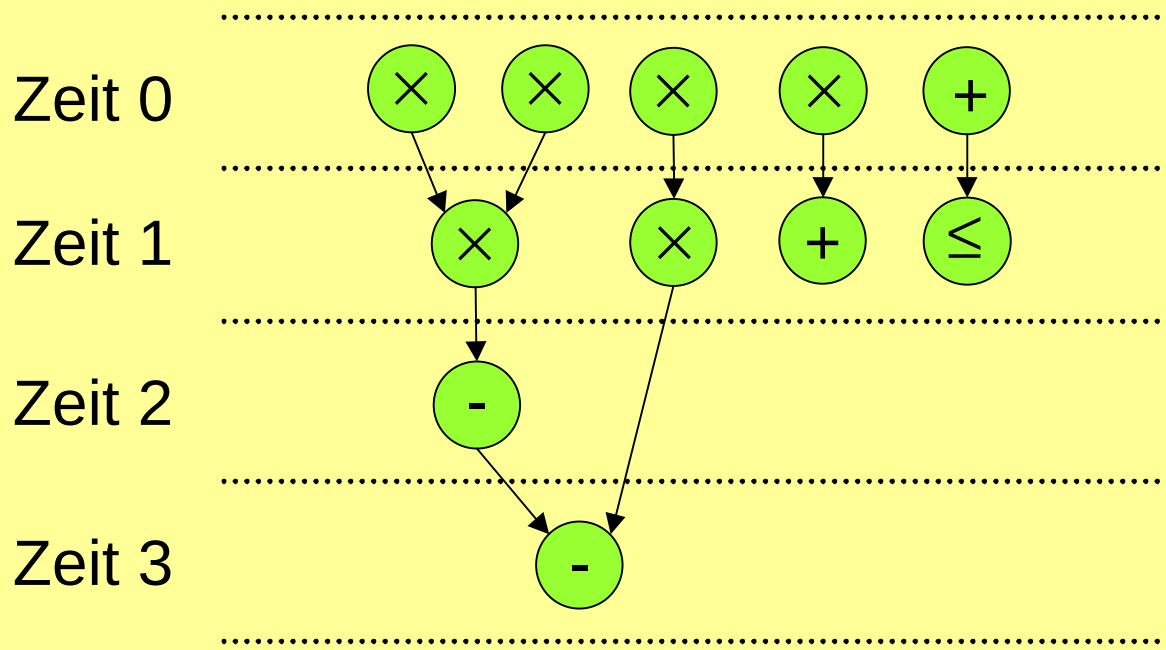
Ablaufplanung

- ALAP (as late as possible)



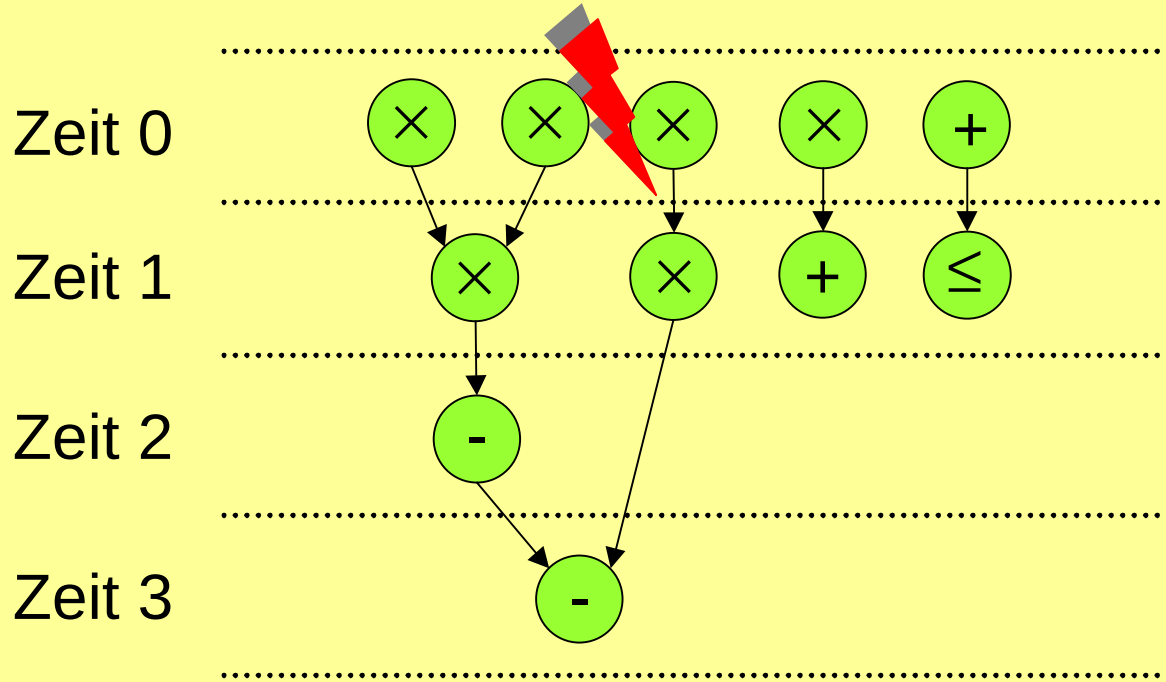
Ablaufplanung

- ASAP mit bedingter Verschiebung
- z.B. 2 Multiplizierer, 2 ALUs



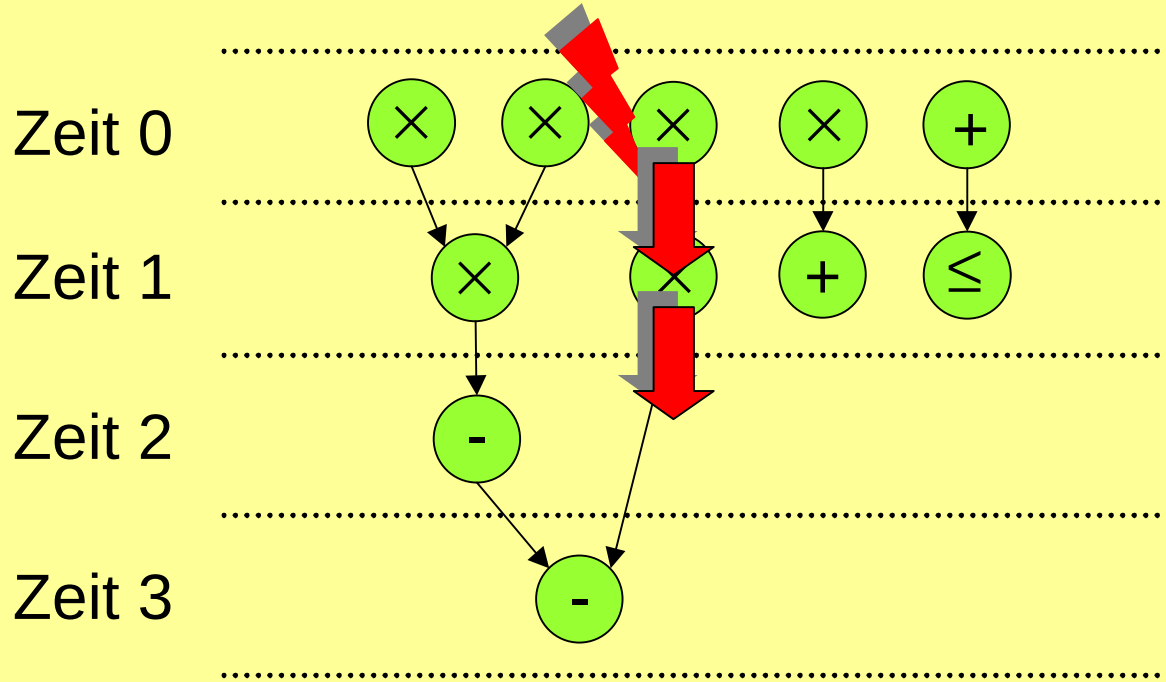
Ablaufplanung

- ASAP mit bedingter Verschiebung
- z.B. 2 Multiplizierer, 2 ALUs



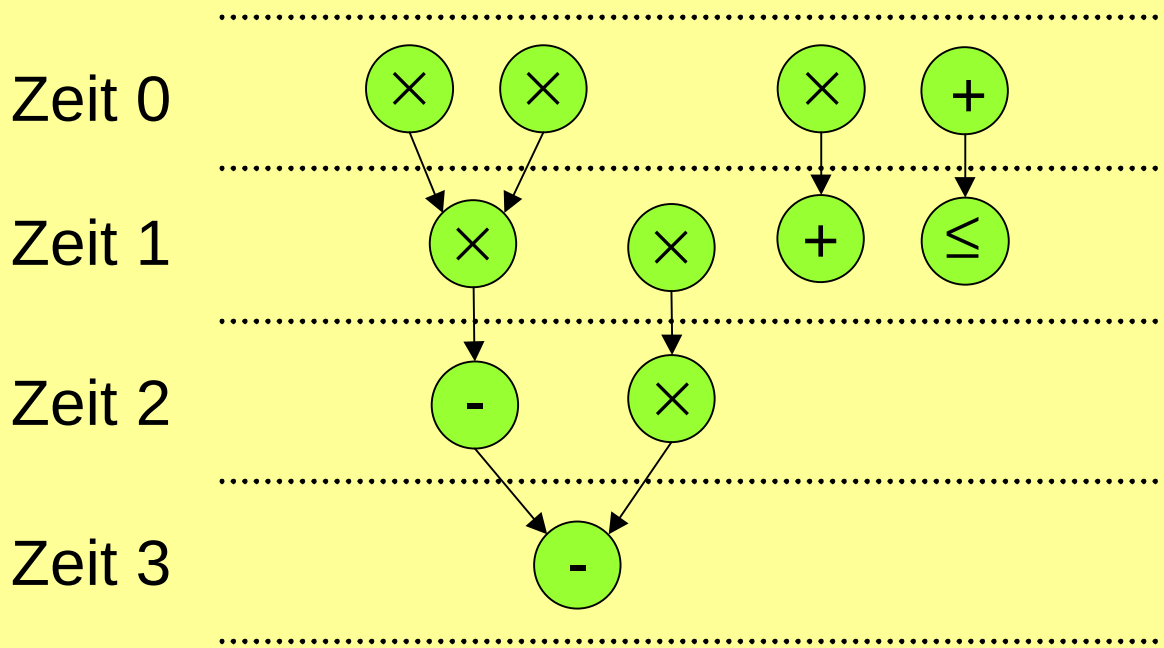
Ablaufplanung

- ASAP mit bedingter Verschiebung
- z.B. 2 Multiplizierer, 2 ALUs



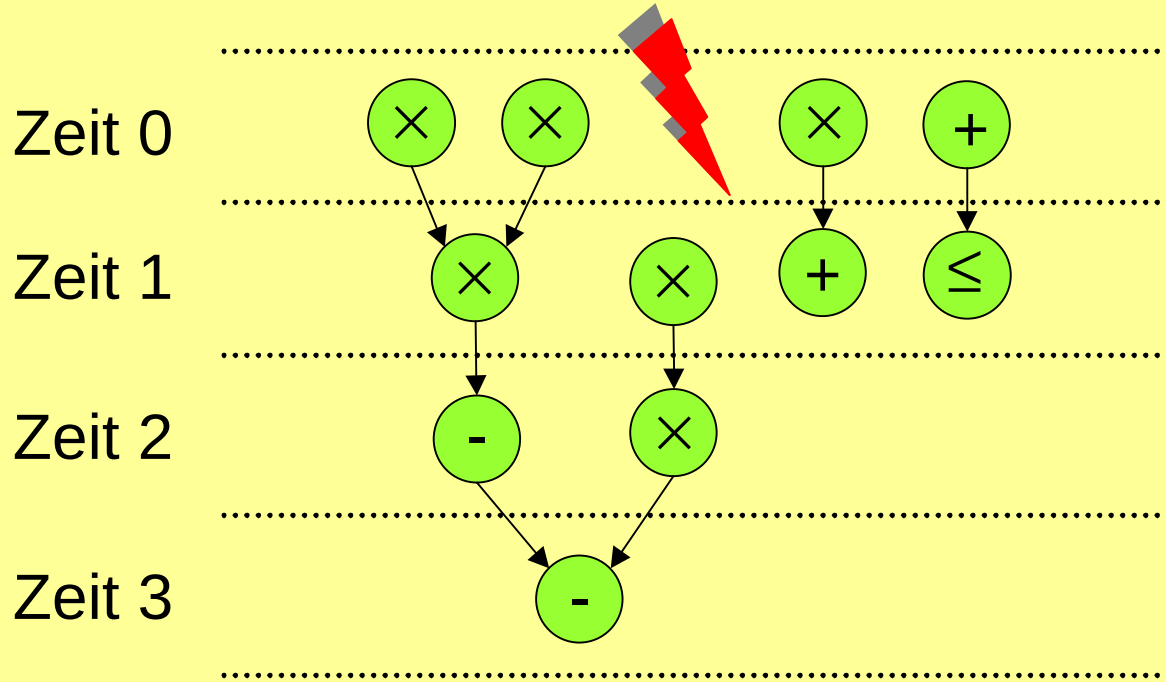
Ablaufplanung

- ASAP mit bedingter Verschiebung
- z.B. 2 Multiplizierer, 2 ALUs



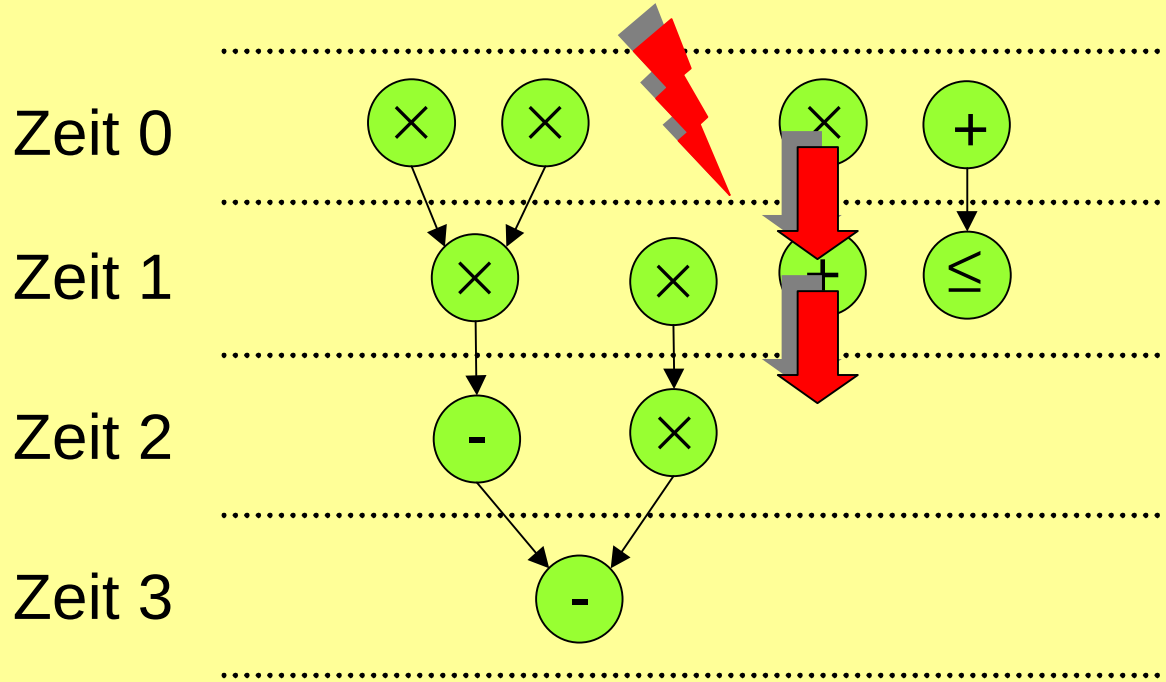
Ablaufplanung

- ASAP mit bedingter Verschiebung
- z.B. 2 Multiplizierer, 2 ALUs



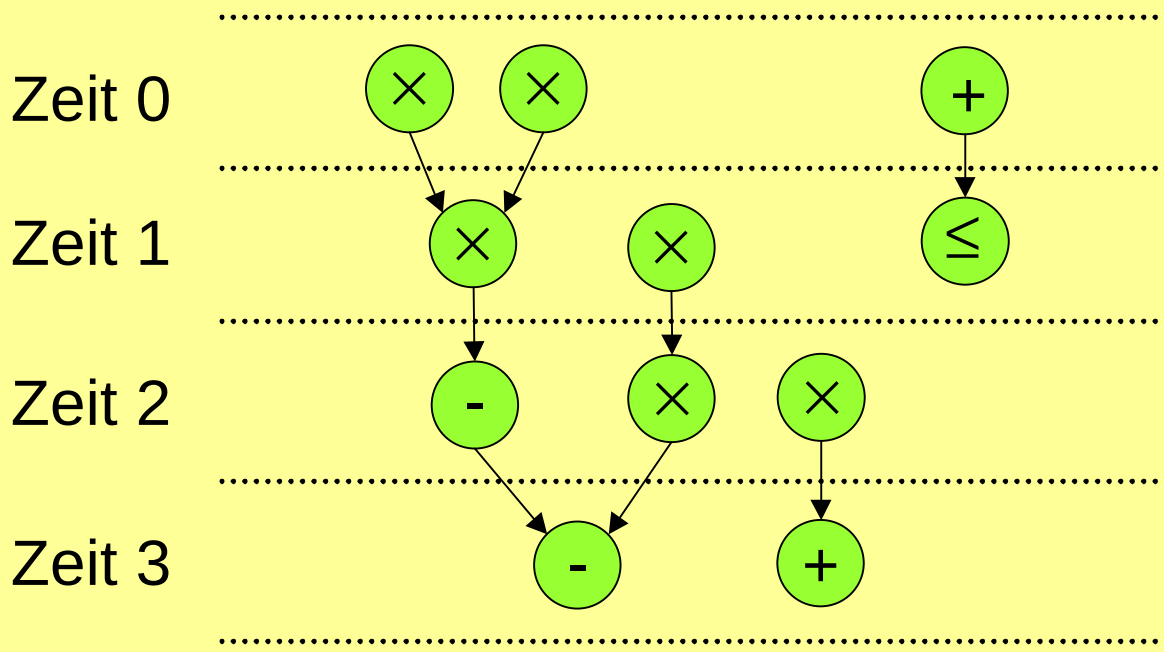
Ablaufplanung

- ASAP mit bedingter Verschiebung
- z.B. 2 Multiplizierer, 2 ALUs



Ablaufplanung

- ASAP mit bedingter Verschiebung
- z.B. 2 Multiplizierer, 2 ALUs



Ablaufplanung

Problem ist für kombinierte Zeit- und Resourcenbeschränkung NP-vollständig

Weitere Algorithmen

- Listenscheduling
- Force-direct scheduling
- Ganzzahlige lineare Programmierung

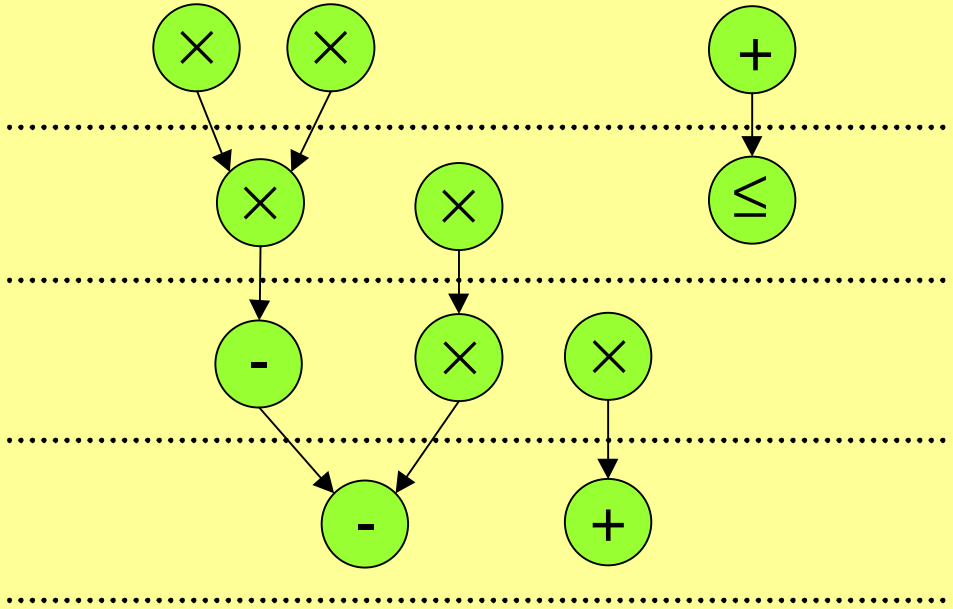
Bindung

Zuordnung von Resourcen zu Operationen

- Bindung kann vor, während oder nach der Ablaufplanung durchgeführt werden
- Bindung nach der Ablaufplanung ist in polynomieller Zeit möglich
- Zuordnung von Speicherstellen/Register zu Programmvariablen gehört auch zur Bindung

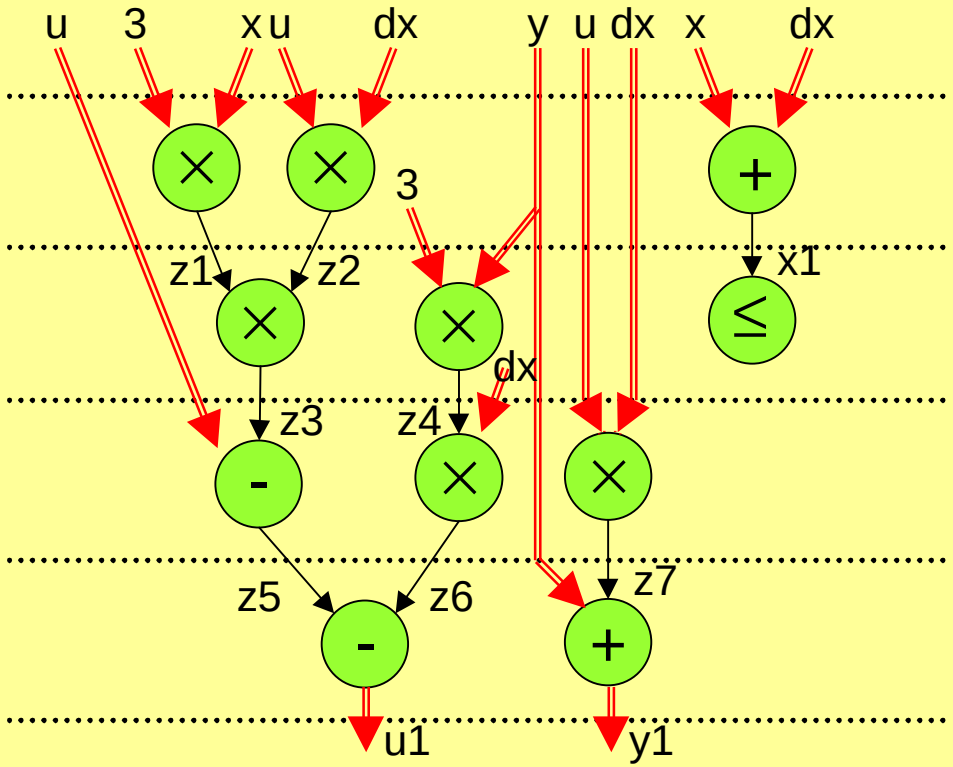
Bindung

- Lebenszeit von Variablen



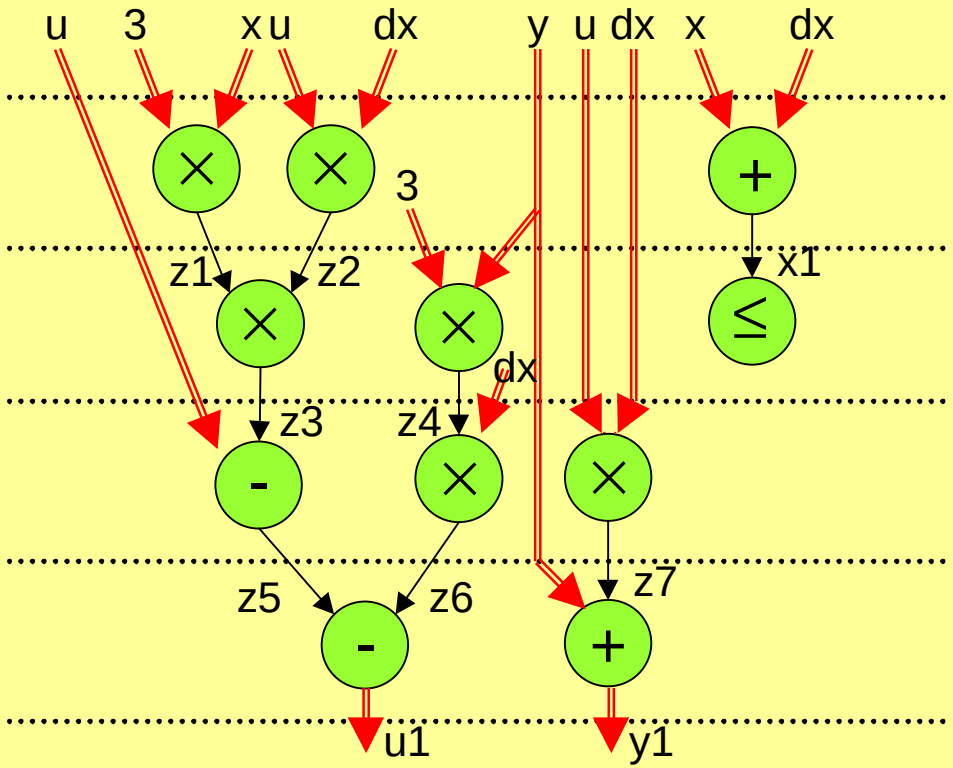
Bindung

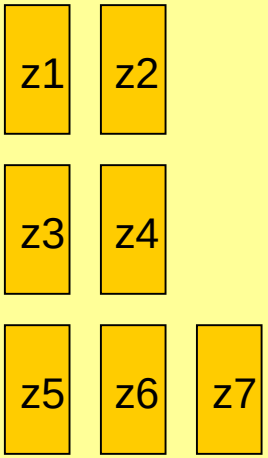
- Lebenszeit von Variablen



Bindung

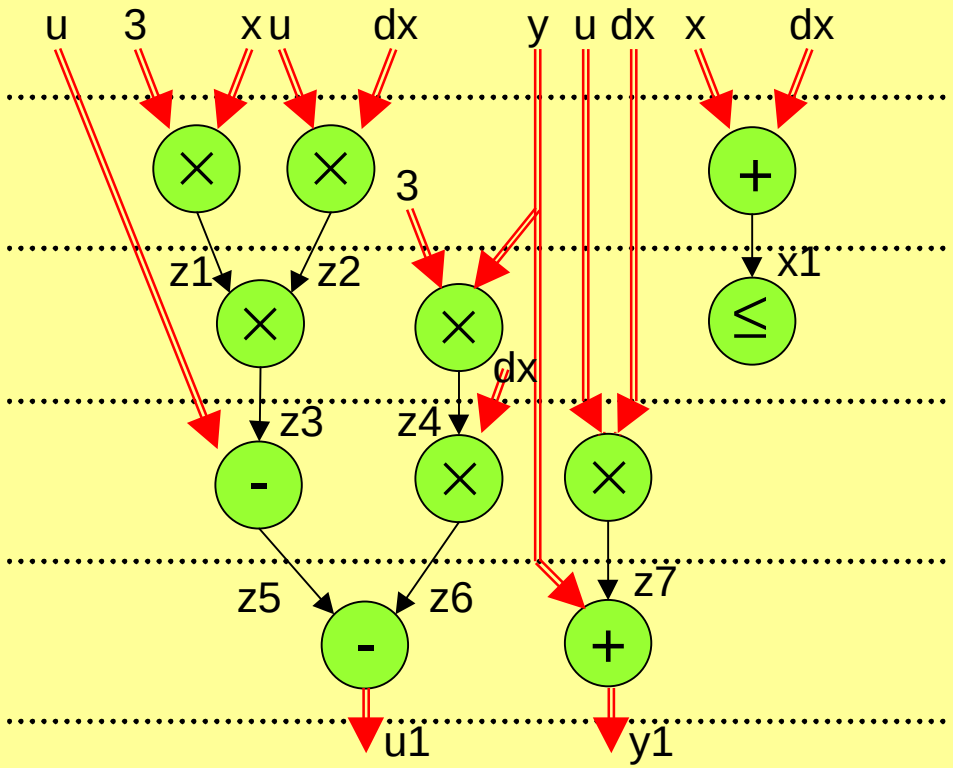
- Lebenszeit von Variablen

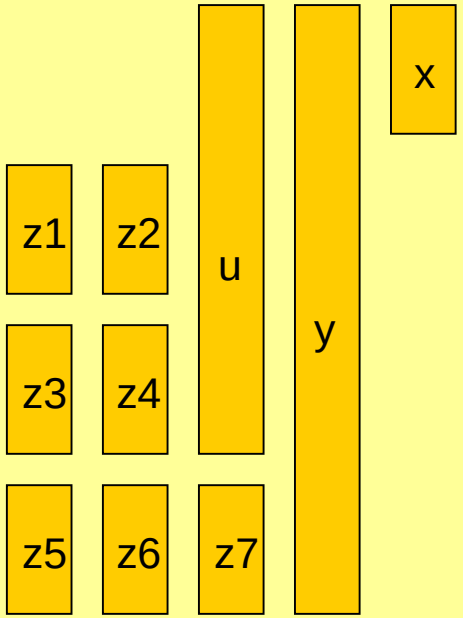




Bindung

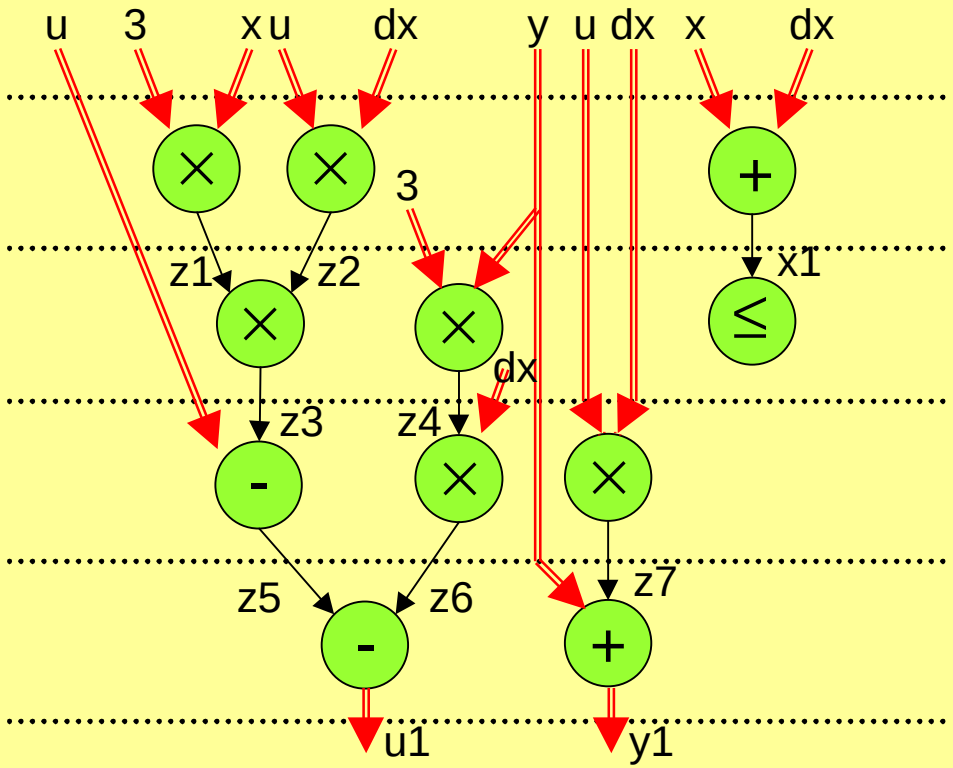
- Lebenszeit von Variablen

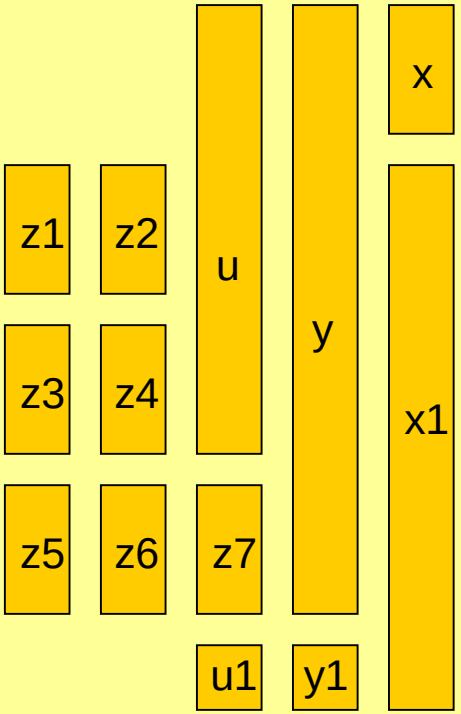




Bindung

- Lebenszeit von Variablen

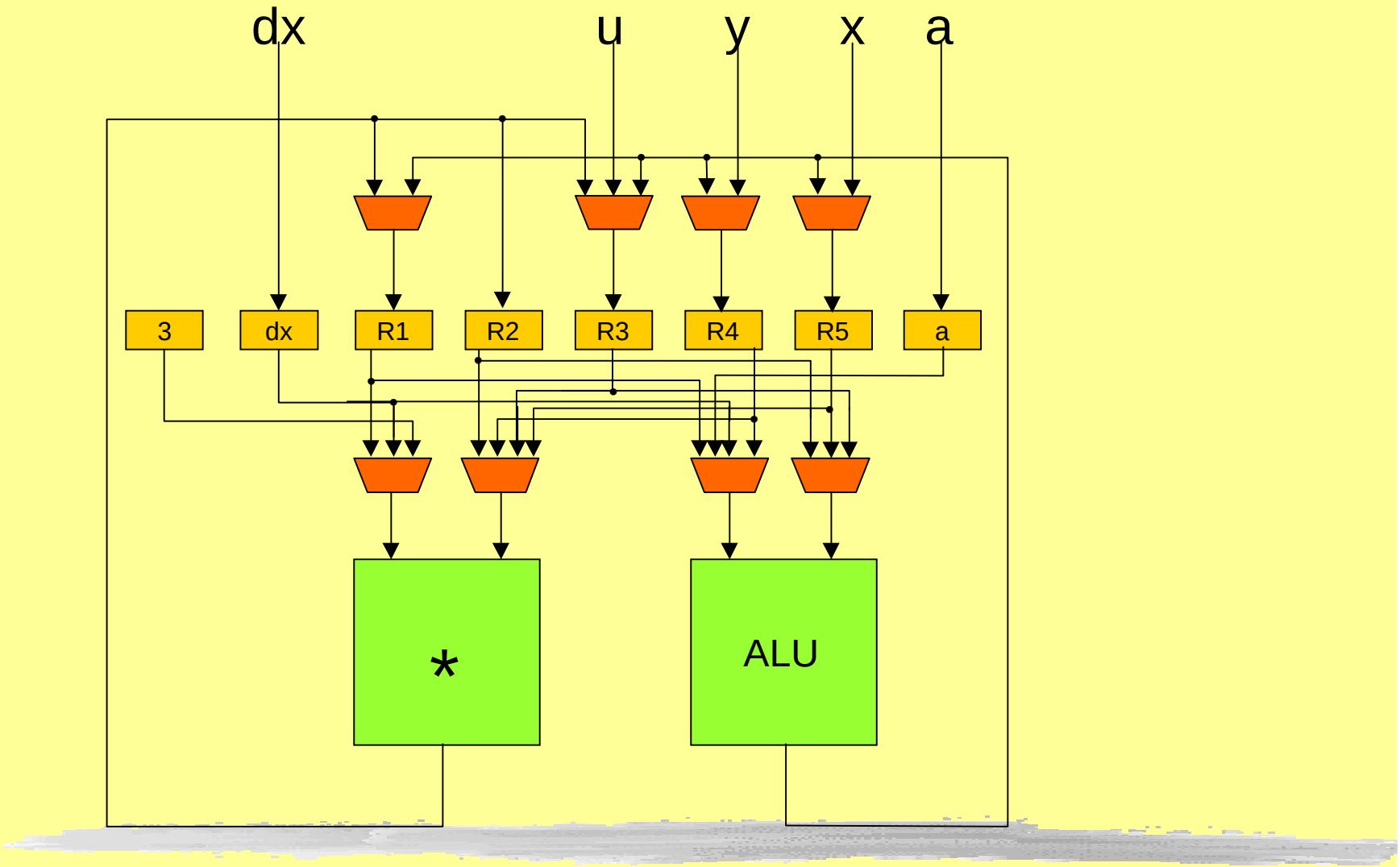




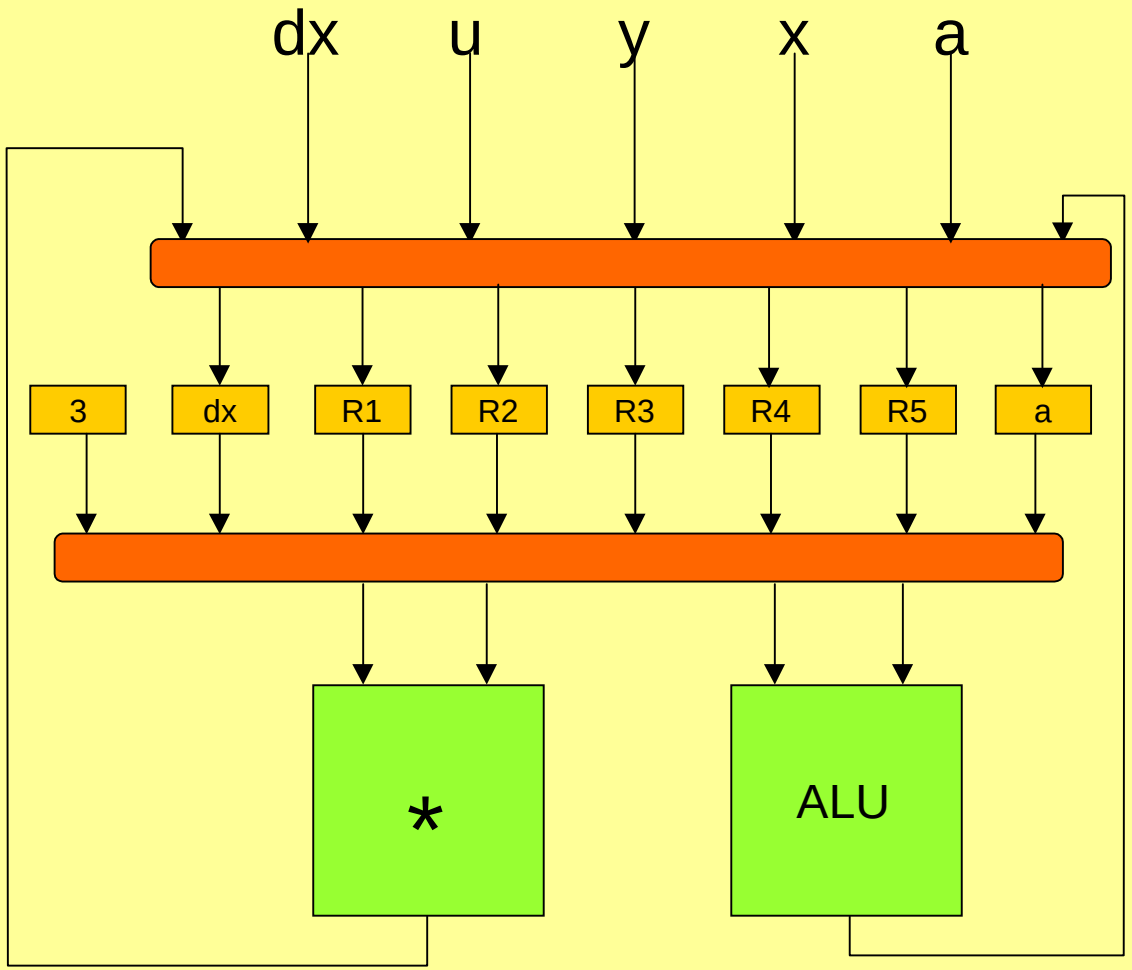
Synthese des Operstionswerks

- = Datenpfad
- Auswahl der Kommunikationsstrukturen
  - Multiplexern
    - hoher Verdrahtungsaufwand
    - paralleler Datentransport
  - Bussen
    - geringer Verdrahtungsaufwand
    - wird eventuell zum Flaschenhals
  - Mischformen

Operationswerk mit Multiplexern

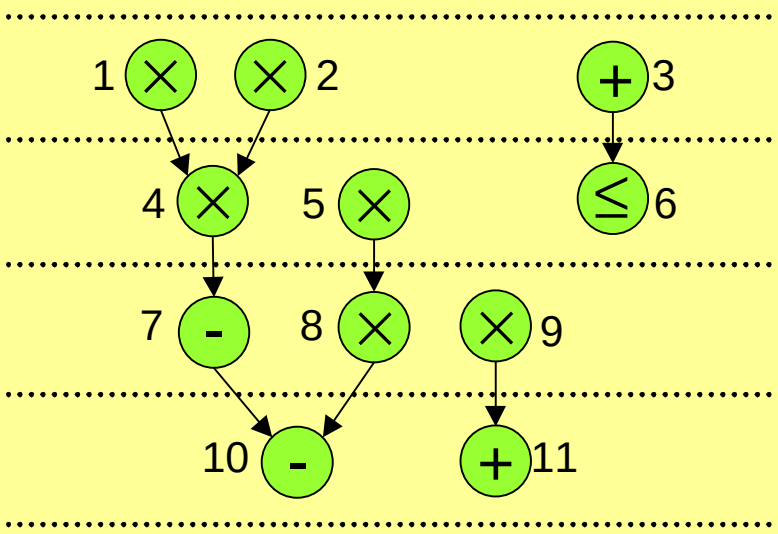


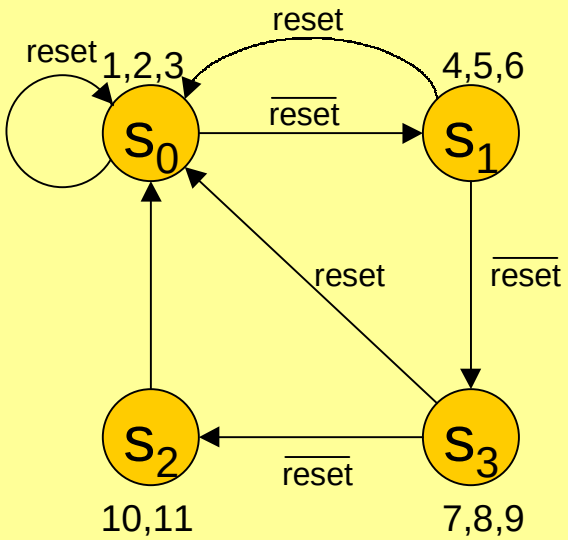
Operationswerk mit Bussen



Synthese des Steuerwerks

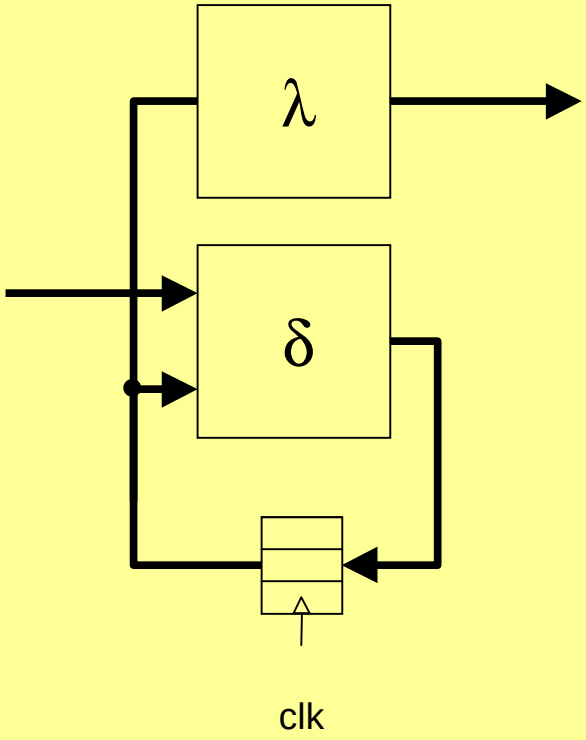
- Umsetzem des Ablaufplans in einen endlichen Automaten



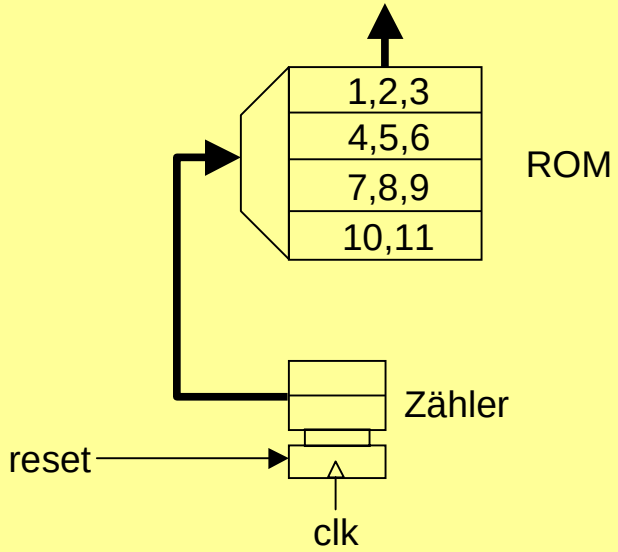


Implementierung der Steuerung

- Als Schaltnetz



- Als Mikroprogramm (ROM)



1 Hardwareentwurf

- **1.1** Überblick, Hardwareentwurfsschritte
- **1.2** Hardwarebeschreibungssprachen
- **1.3** Hardwaresimulation-/Verifikation
- **1.4** Hardwaresynthese
  - Highlevel Synthese
  - Gattersynthese
- **1.5** Plazierung und Verdrahtung

Gattersynthese

- Ersetzen der RT-Module durch Gatterbeschreibungen
  - FSM-Synthese (Steuerwerk)
  - Synthese arithmetischer Einheiten
- Logikoptimierungen
  - Minimierung (2-stufig, mehrstufig)
  - Retiming
- Einfügen von Scanpfaden (Design for Test)

→ TI 1 und TI 2