

1 Hardwareentwurf

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

Literatur

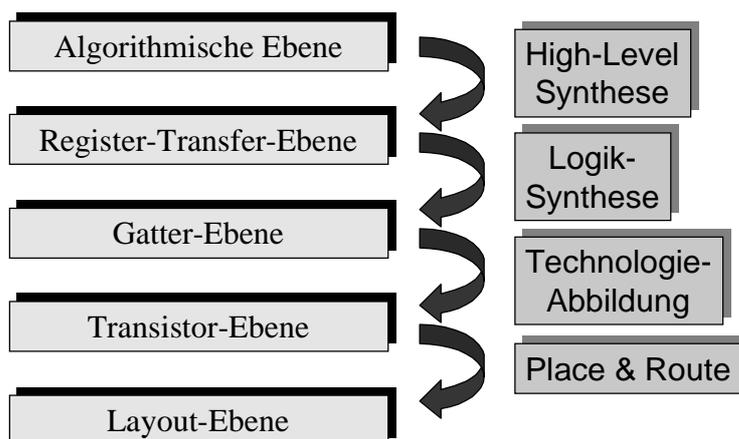
- G. De Micheli
Synthesis and Optimization of Digital Circuits
- J. Teich
Digitale Hardware/Software Systeme
- T. Kropf
VLSI Entwurf

HW-Synthese

Automatische Abbildung von Beschreibungen auf hoher Abstraktionsebenen in niedrige Abstraktionsebenen

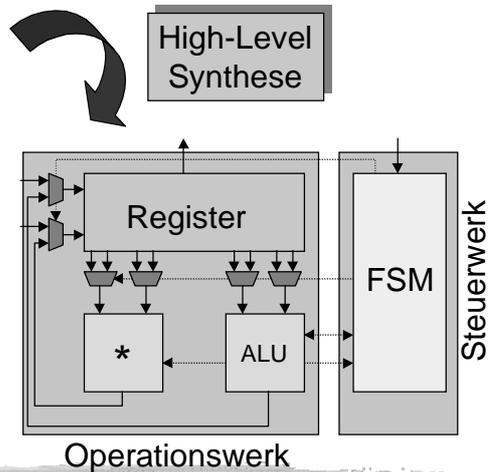
- Highlevel-Synthese
algorithmische Ebene \Rightarrow Registertransfer Ebene
- Gattersynthese
Registertransfer Ebene \Rightarrow Gatterebene

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;
}
```



JR - RA - SS02

Kap. 1.4

5

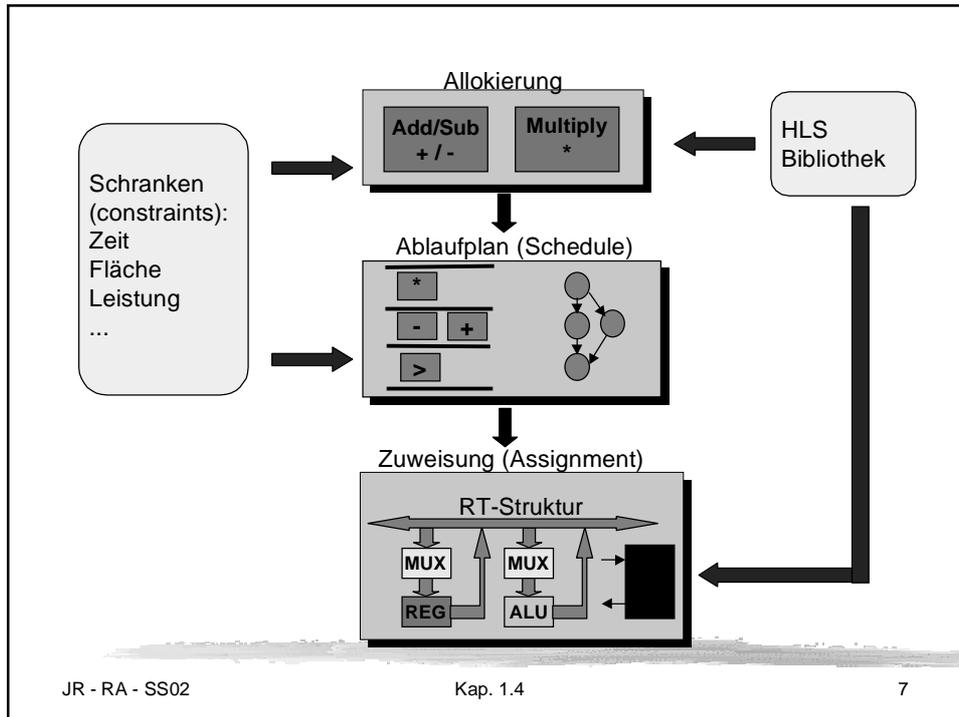
High-Level Synthese

- Allokation
Auswahl der HW-Komponenten für die Operationen
- Ablaufflanung
Bestimmung der Zeitpunkte an denen die Operationen auszuführen sind
- Bindung
Zuordnung von Variablen zu Registern, Operationen zu funktionalen Einheiten und Kommunikationskanälen zu Bussen

JR - RA - SS02

Kap. 1.4

6



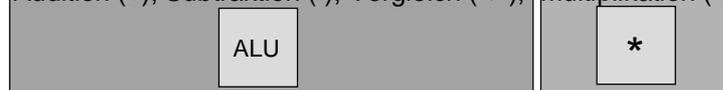
Allokation

- 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;
}
```

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



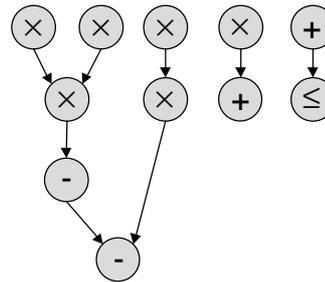
Ablaufplanung

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

Problemmgraph

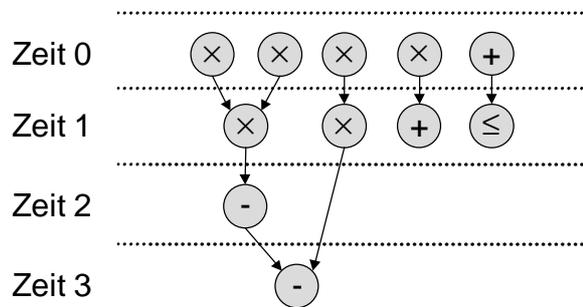
- Operationen
- Datenabhängigkeiten

```
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;
}
```



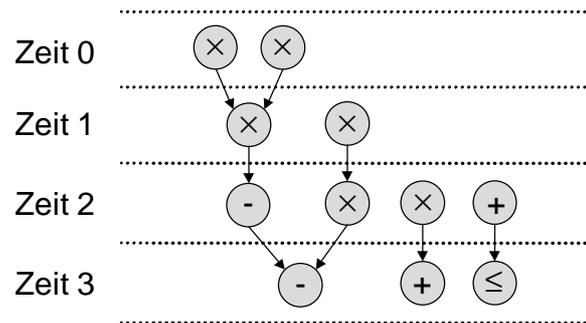
Ablaufplanung

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



Ablaufplanung

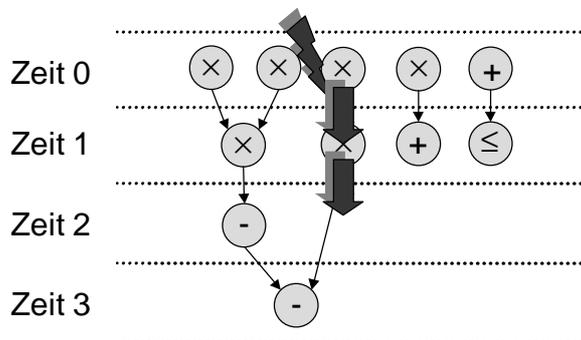
■ ALAP (as late as possible)



Ablaufplanung

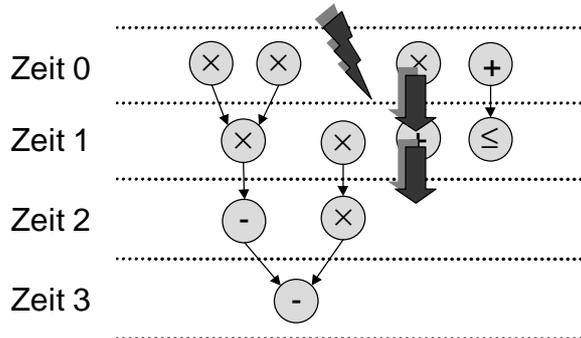
■ ASAP mit bedingter Verschiebung

■ z.B. 2 Multiplizierer, 2 ALUs



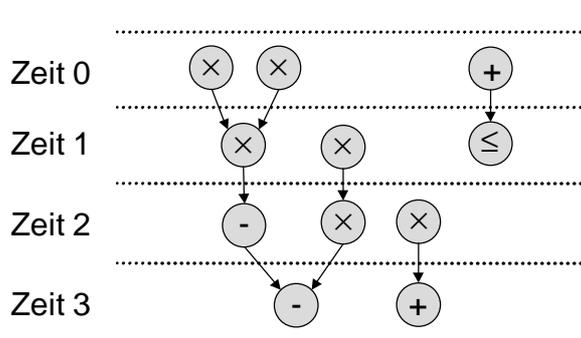
Ablaufplanung

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



Ablaufplanung

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



Ablaufplanung

Problem ist für kombinierte Zeit- und Ressourcenbeschränkung NP-vollständig

Weitere Algorithmen

- Listenscheduling
- Force-direct scheduling
- Ganzzahlige lineare Programmierung

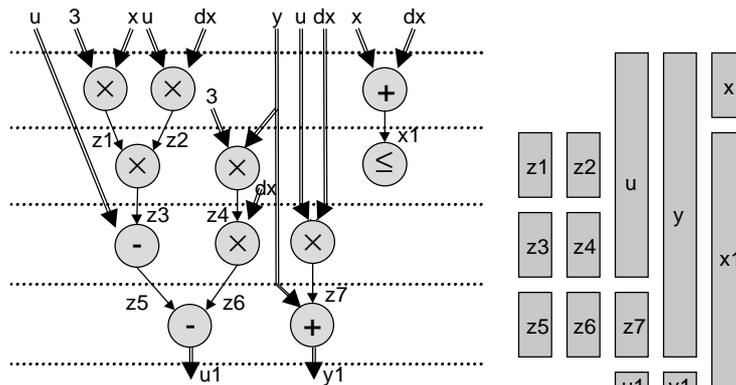
Bindung

Zuordnung von Ressourcen 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



JR - RA - SS02

Kap. 1.4

19

Synthese des Operstionswerks

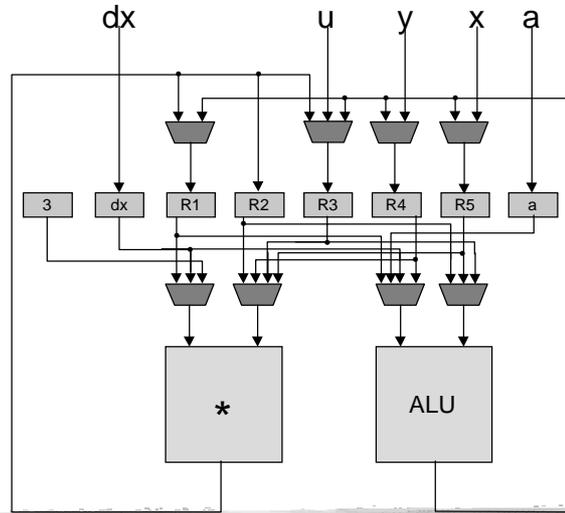
- = Datenpfad
- Auswahl der Kommunikationsstrukturen
 - Multiplexern
 - hoher Verdrahtungsaufwand
 - paralleler Datentransport
 - Bussen
 - geringer Verdrahtungsaufwand
 - wird eventuell zum Flaschenhals
 - Mischformen

JR - RA - SS02

Kap. 1.4

20

Operationswerk mit Multiplexern

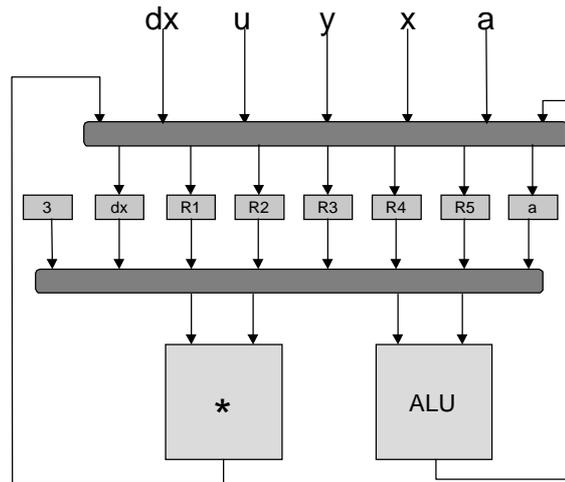


JR - RA - SS02

Kap. 1.4

21

Operationswerk mit Bussen



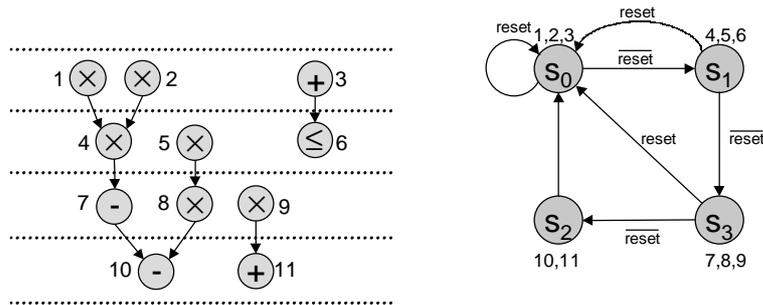
JR - RA - SS02

Kap. 1.4

22

Synthese des Steuerwerks

- Umsetzen des Ablaufplans in einen endlichen Automaten



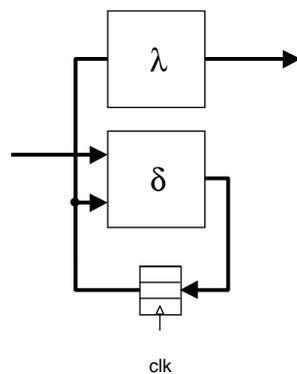
JR - RA - SS02

Kap. 1.4

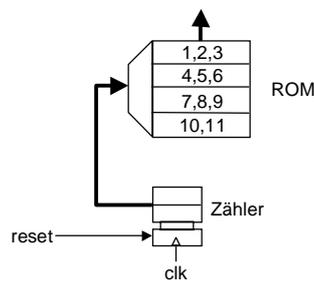
23

Implementierung der Steuerung

- Als Schaltnetz



- Als Mikroprogramm (ROM)



JR - RA - SS02

Kap. 1.4

24

1 Hardwareentwurf

- **1.1** Überblick, Hardwareentwurfsschritte
- **1.2** Hardwarebeschreibungssprachen
- **1.3** Hardwaresimulation-/Verifikation
- **1.4** Hardwaresynthese
 - ┆ Highlevel Synthese
 - ┆ Gattersynthese
- **1.5** Platzierung 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