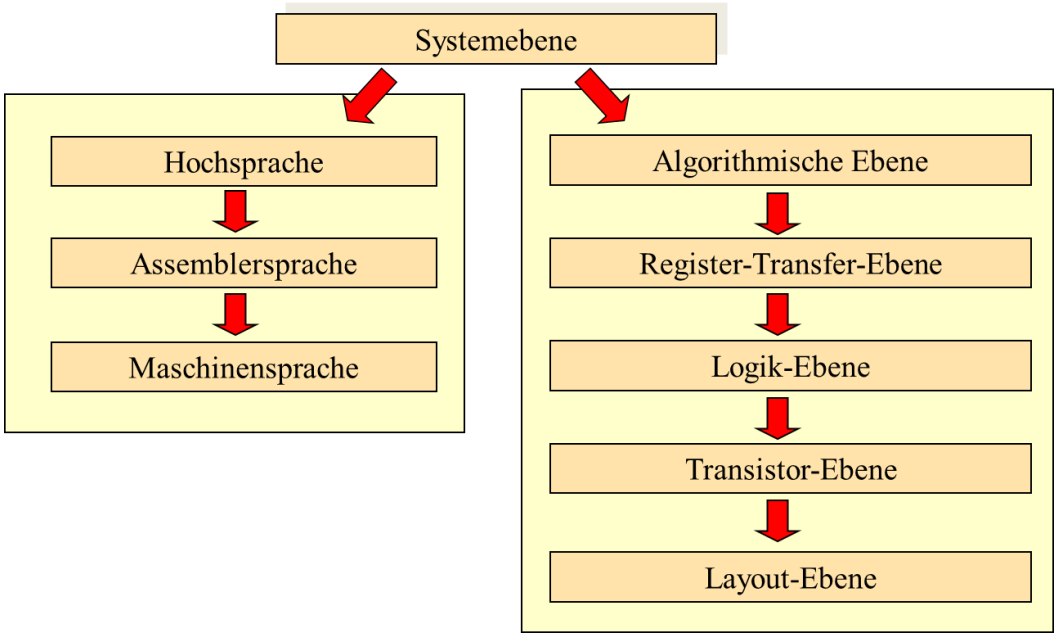
Einführung in die Technische Informatik

Systementwurf

Robert Wille

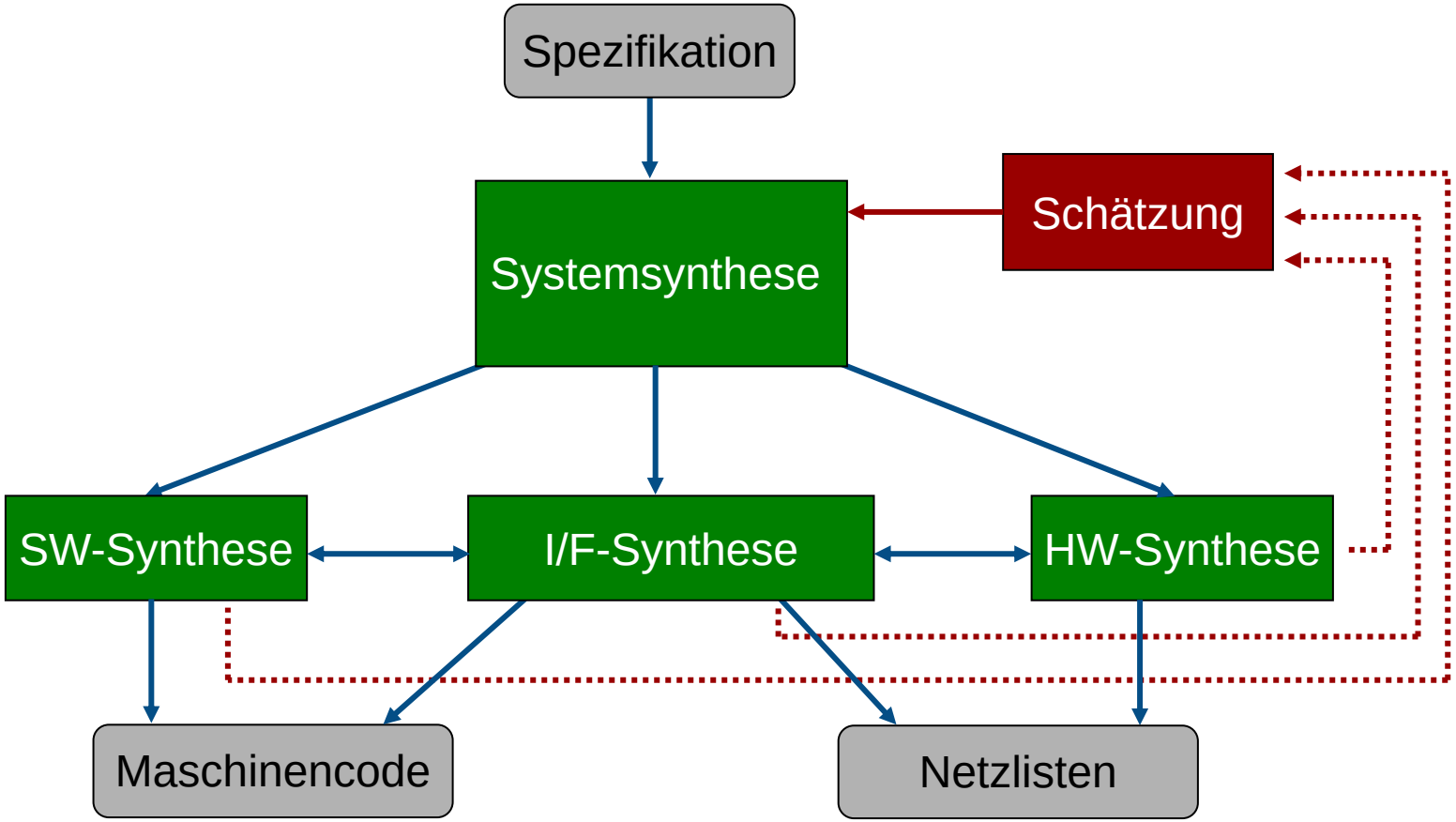
Motivation

- Bisher:
  Hardwareentwurf
- In der Praxis:
  Zusammenspiel zwischen Hard- und Software



- Hardware/Software Co-Design

Motivation


Spezifikation
Systemsynthese
Schätzung
SW-Synthese
I/F-Synthese
HW-Synthese
Maschinencode
Netzlisten

Modellierung von Systemen

- Ziele
  - Detaillierte Beschreibung des Systems
  - Exploration des Entwurfsraums
  - Automatisierte Weiterverarbeitung

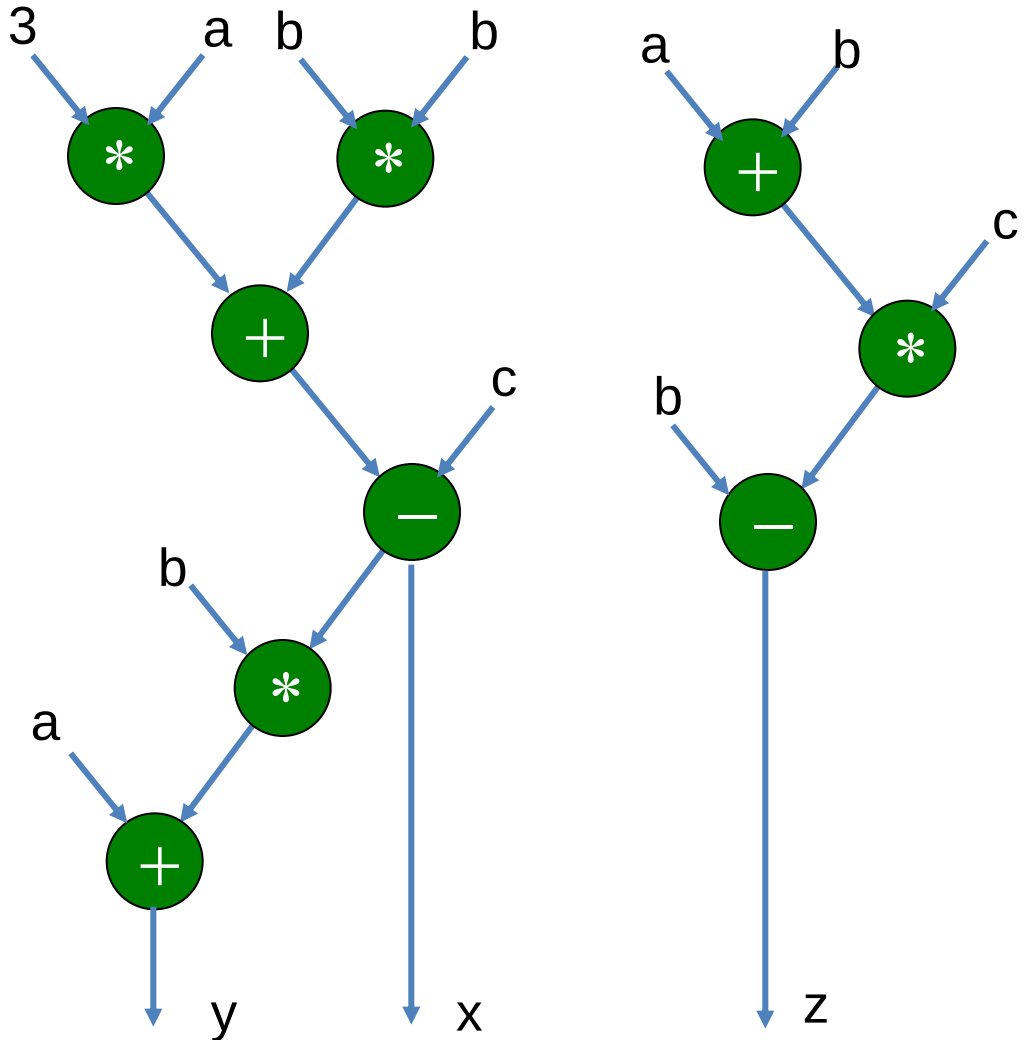
- Unterschiede verschiedener Modellierungsansätze
  - Universell
    - Zielarchitektur
    - Modellierte Aspekte (Funktion, Zeitverhalten, Kosten, …)
  - Formal
  - Synthetisierbar
  - Standardisiert
  - …

Kontroll/Datenflussmodelle

- Graph *G(V,E)*
  - Menge der Knoten *V*: Operationen, Tasks
  - Menge der Kanten *E*: Abhängigkeiten
- Abhängigkeiten
  - Datenabhängigkeit (data dependency)
  - Kontrollflussabhängigkeit (control dependency)
  - Ressourcenkonflikt (entsteht durch die Implementierung)
- Modelle
  - Datenflussgraphen (DFGs)
  - Kontrollflussgraphen (CFGs)
  - Gemischte Kontroll-/Datenflussgraphen (CDFGs) (Bsp.: Sequenzgraphen)

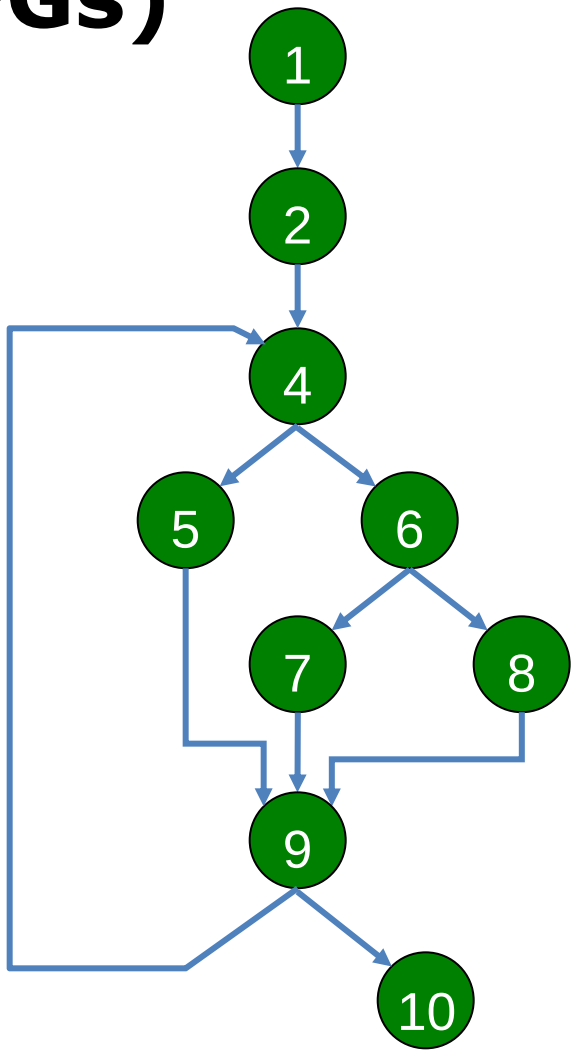
Datenflussgraphen (DFGs)

```
x = 3*a + b*b - c;
y = a + b*x;
z = b - c*(a + b);
```



Kontrollflussgraphen (CFGs)

```
    what_is_this {
 1    read (a,b);
 2    done = FALSE;
 3    REPEAT {
 4      IF (a>b)
 5         a = a-b;
 6      ELSE IF (b>a)
 7         b = b-a;
 8      ELSE done = TRUE;
 9    } UNTIL done;
10    write (a);
    }
```

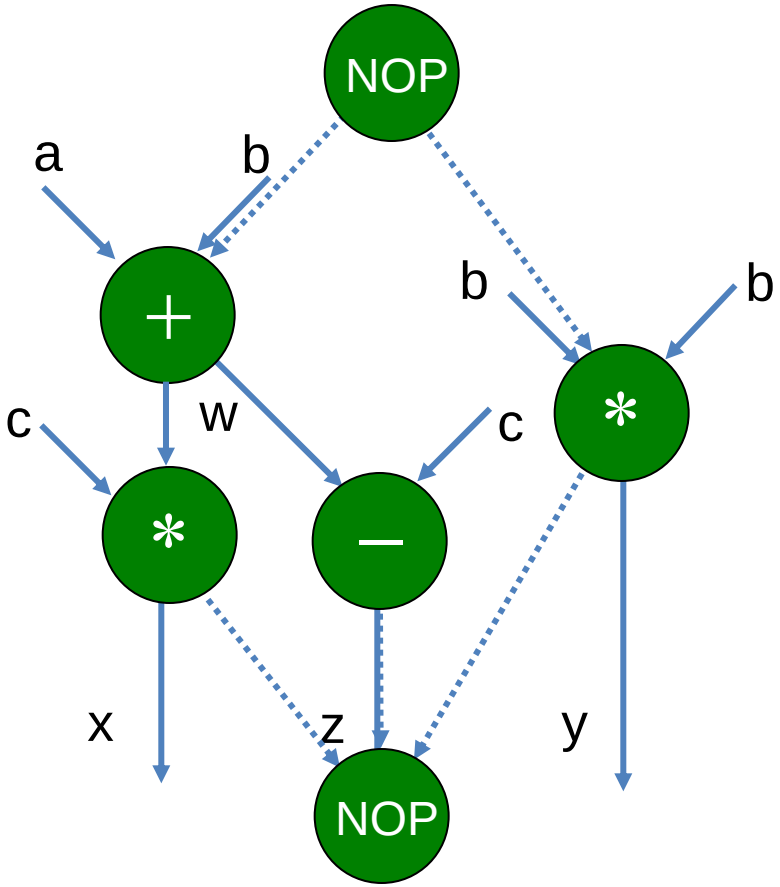


Sequenzgraphen

- Hierarchie von verketteten Einheiten
  - Einheiten modellieren den Datenfluss
  - Hierarchie modelliert den Kontrollfluss
- Spezielle Knoten
  - Start/Endknoten: NOP (No operation)
  - Verzweigungsknoten (BRANCH)
  - Iterationsknoten (LOOP)
  - Modulaufrufknoten (CALL)
- Attribute
  - Knoten: Berechnungszeiten, Kosten, …
  - Kanten: Bedingungen für Verzweigungen und Iterationen

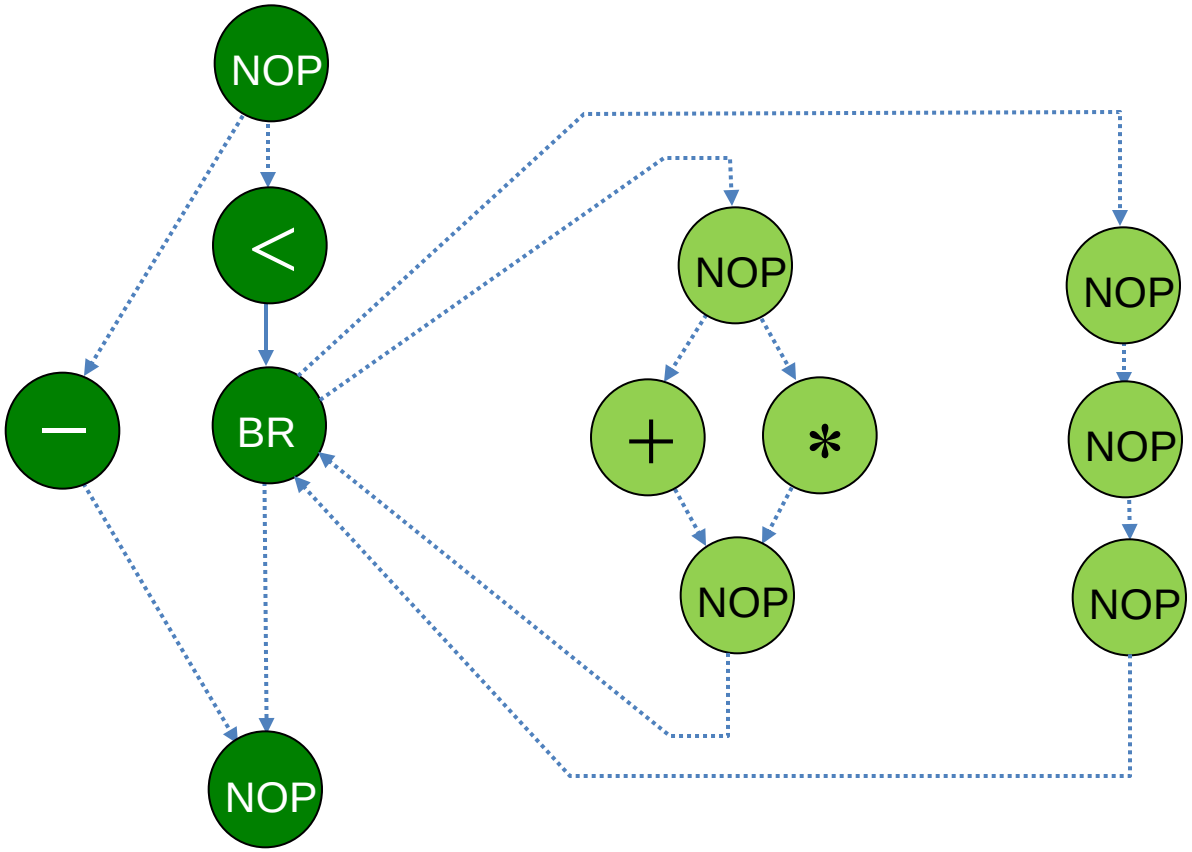
Sequenzgraphen - Einheit

```
w = a + b;
x = w * c;
y = b * b;
z = w - c;
```



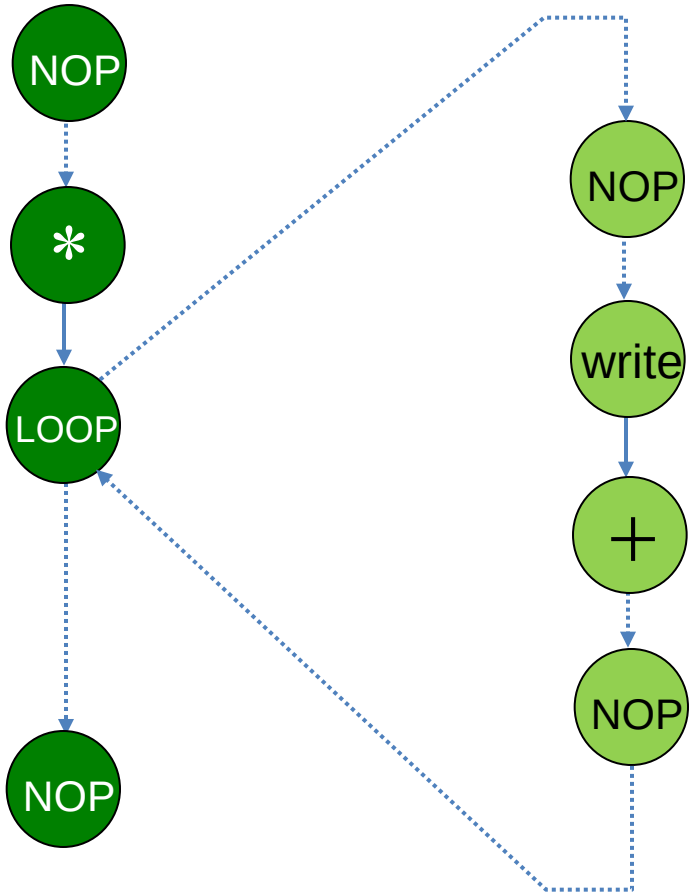
Sequenzgraphen - Branch

```
c = a < b;
IF (c) THEN
  p = m + n;
  q = m * n;
ENDIF
x = a - b;
```



Sequenzgraphen - Loop

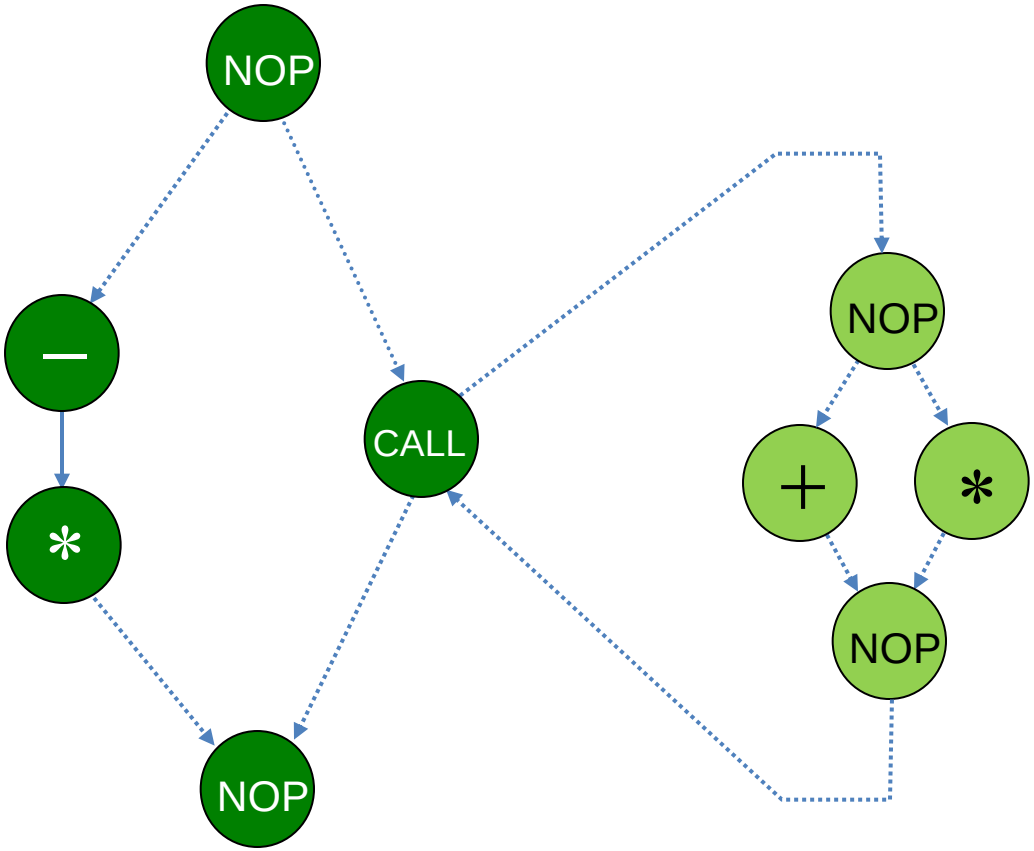
```
d = 2*x;
WHILE (d<5) DO
  write(d);
  d = d + 1;
ENDWHILE
```



Sequenzgraphen - Call

```
d = x - y;
e = d * x;
sub(x, y);
...
```

```
PROCEDURE sub(m,n)
  p = m + n;
  q = m * n;
END sub
```



Synthese

- Gegeben:
  Spezifikation und Randbedingungen

- Synthese:
  Automatisches Erzeugen einer Implementierung

- Typische Aufgaben:
  - Allokation (Allocation)
  - Bindung (Binding)
  - Ablaufplanung (Scheduling)

Ablaufplanung

- statisch / dynamisch
- preemptive / non-preemptive
- mit / ohne Ressourcenbeschränkungen
- aperiodisch / periodisch (iterativ)

Ohne Ressourcenbeschränkungen

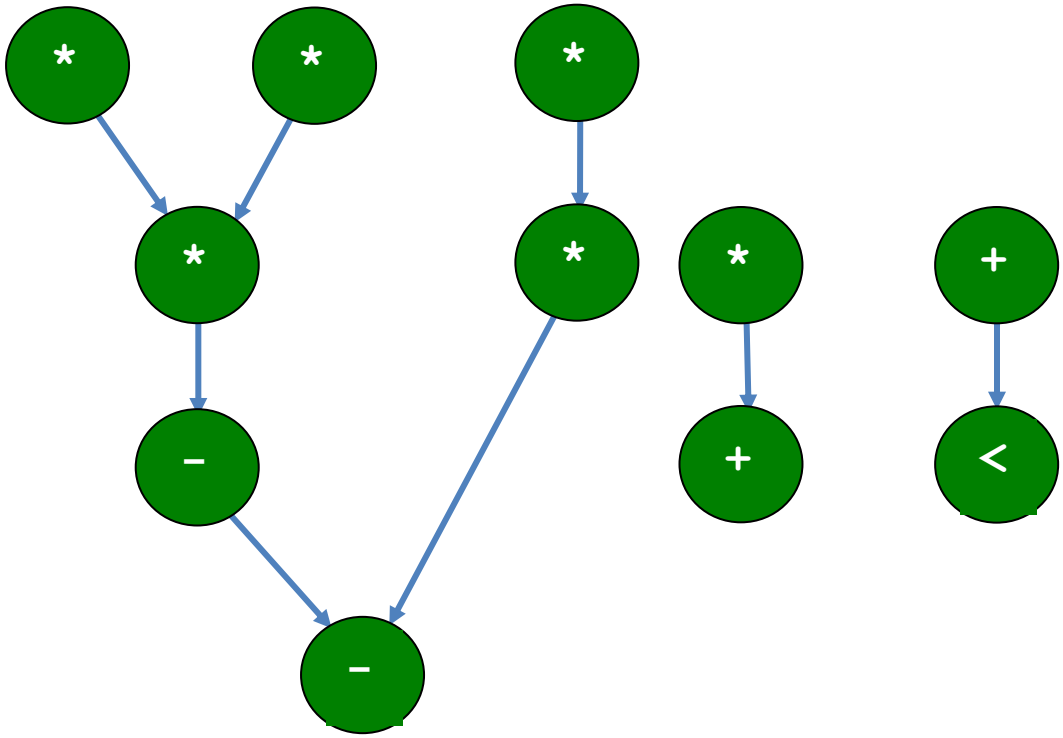
- ASAP (as soon as possible)
  - bestimmt die frühest möglichen Startzeitpunkte für die Tasks
  - liefert minimale Latenz
- ALAP (as late as possible)
  - bestimmt die spätest möglichen Startzeitpunkte für die Tasks bei vorgegebener Latenz
- Mobilität eines Tasks ist die Differenz der Startzeitpunkte ALAP-ASAP
  - Mobilität 0 → Task liegt auf dem kritischen Pfad

Problemgraph

Definition:

Ein **Problemgraph** G(V,E) ist ein gerichteter, azyklischer Graph mit Knotenmenge V und Kantenmenge E, in dem jeder Knoten $v_i \in V$ eine Aufgabe (Task, Prozess, Anweisung, Elementaroperation) und jede Kante $e=(v_i, v_j) \in E$ eine Datenabhängigkeit darstellt.

Problemgraph



Ablaufplan

Sei $d_i \in N$ die Bearbeitungszeit für den einen Knoten $v_i \in V$.

Definition:
Ein **Ablaufplan** (Schedule) eines Problemgraphen
G(V, E) ist eine Funktion $T : V \rightarrow N$, die jedem Knoten $v_i \in V$ die Startzeit $t_i = T(v_i)$ zuordnet, so dass gilt:

$$T(v_i) \geq T(v_j) + d_j \text{ für alle } (v_j, v_i) \in E$$

Latenz

Definition:

Die **Latenz** L eines Ablaufplans T eines Problemgraphen G(V, E) ist definiert als

$$L = \max_{v_i \in V}\{T(v_i)+d_i\} - \min_{v_i \in V}\{T(v_i)\}$$

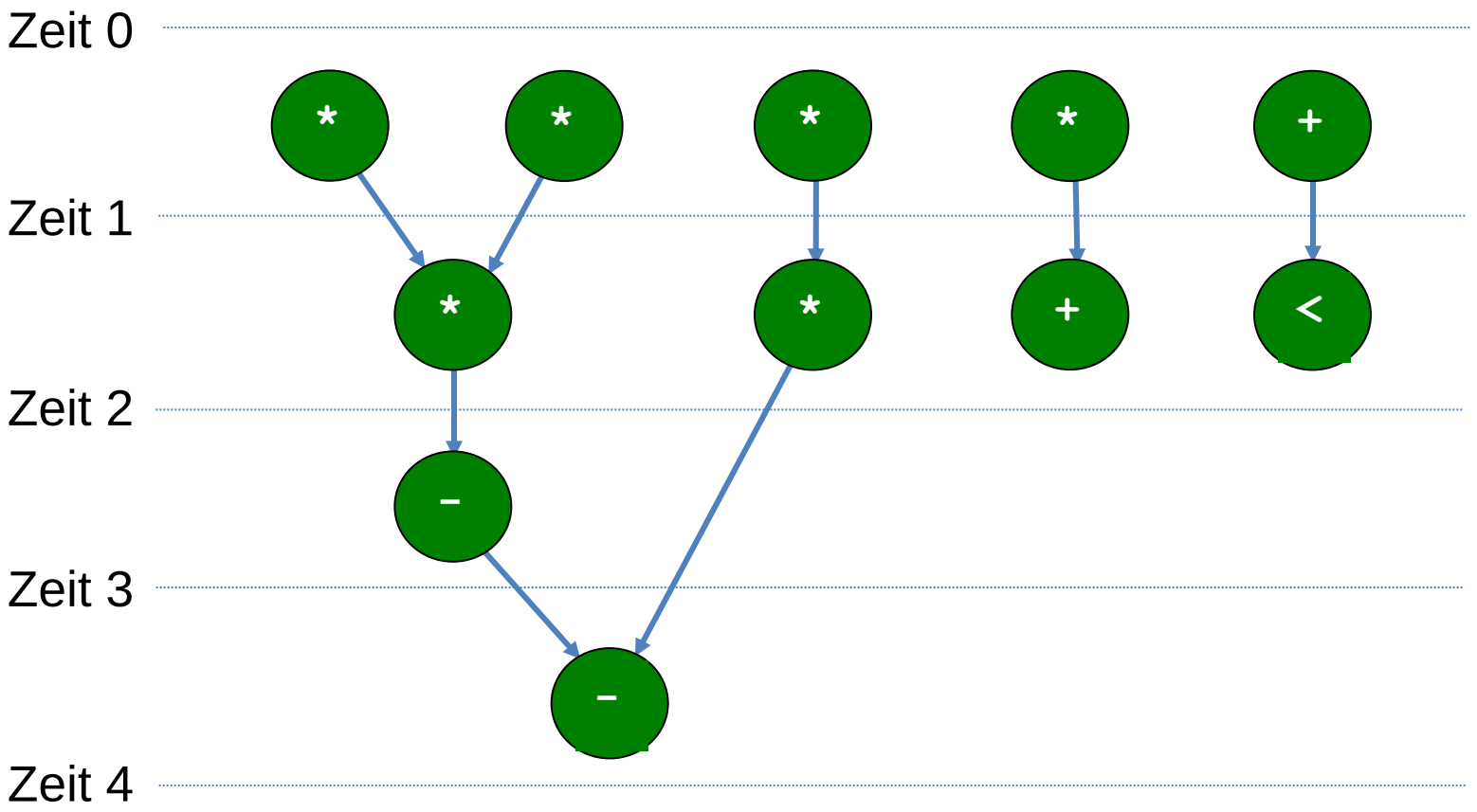
und bezeichnet damit die Anzahl der Zeitschritte des kleinsten Intervalls, das die Ausführungsintervalle aller Knoten $v_i \in V$ einschließt.

ASAP (as soon as possible)

ASAP(G(V,E),d) {
    FOREACH(v_i ohne Vorgänger)
        $T(v_i)^S := 0$;
    REPEAT {
        Wähle einen Knoten v_i aus, dessen Vorgänger alle geplant sind;
        $T(v_i)^S := \max_{j:(v_j,v_i) \in E} \{T(v_j)^S + d_j\}$;
    }
    UNTIL (Alle Knoten v_i geplant);
    RETURN T^S
}

ASAP (as soon as possible)

- latenzoptimaler Ablaufplan: *L = 4*

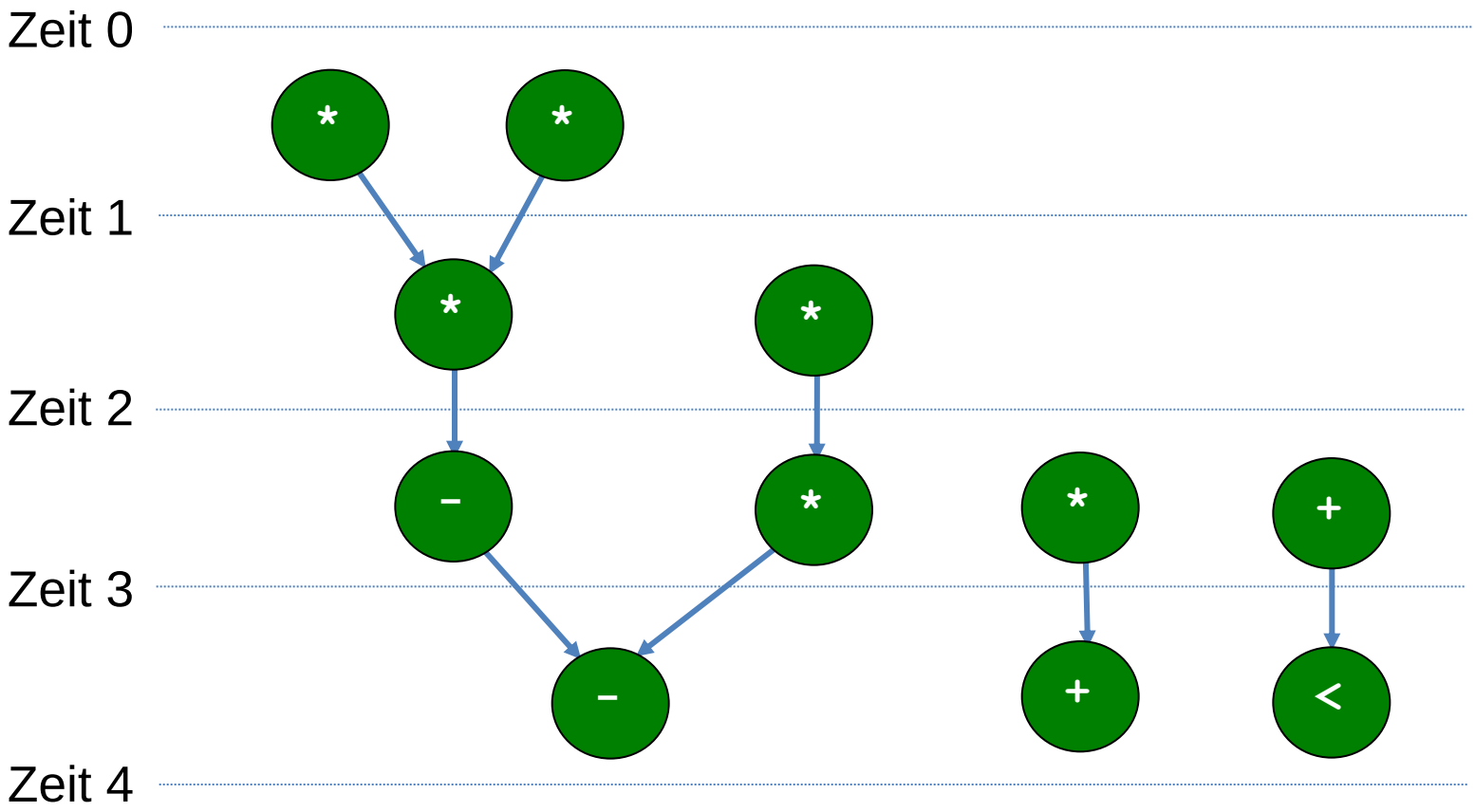


ALAP (as late as possible)

ALAP(G(V,E),d) {
    FOREACH(v_i ohne Nachfolger)
        $T(v_i)^L := L - d_i$;
    REPEAT {
        Wähle einen Knoten v_i aus, dessen Nachfolger
            alle geplant sind;
        $T(v_i)^L := \min_{j:(v_i,v_j) \in E} \{T(v_j)^L\} - d_i$;
    }
    UNTIL (Alle Knoten v_i geplant);
    RETURN T^L
}

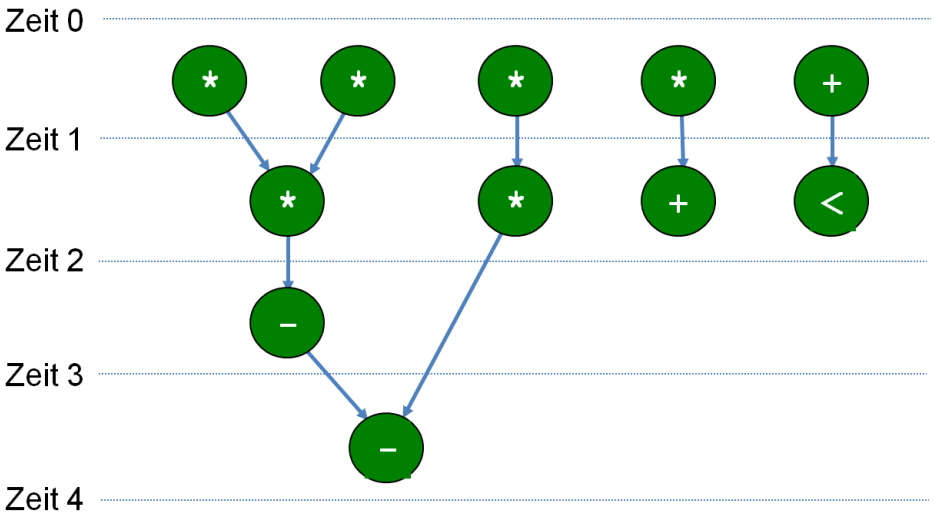
ALAP (as late as possible)

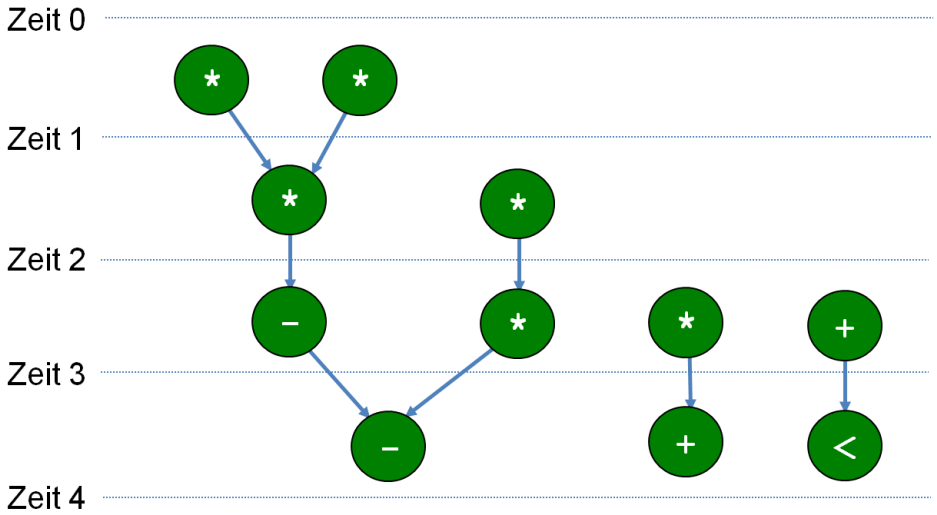
- Latenzschranke: *L = 4*



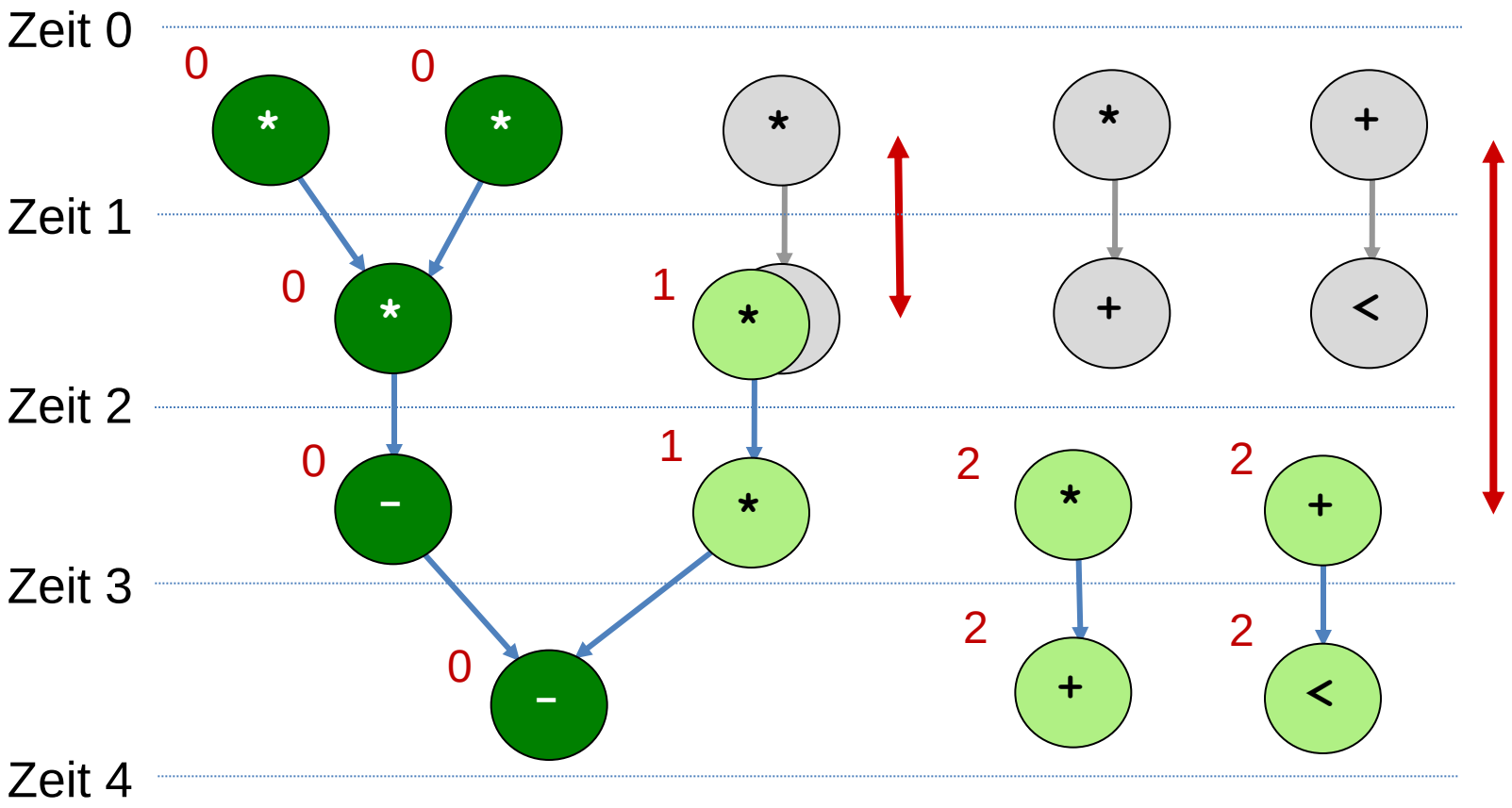
Mobilität

- Mobilität eines Tasks ist die Differenz der Startzeitpunkte ALAP-ASAP





Mobilität



Ressourcegraph

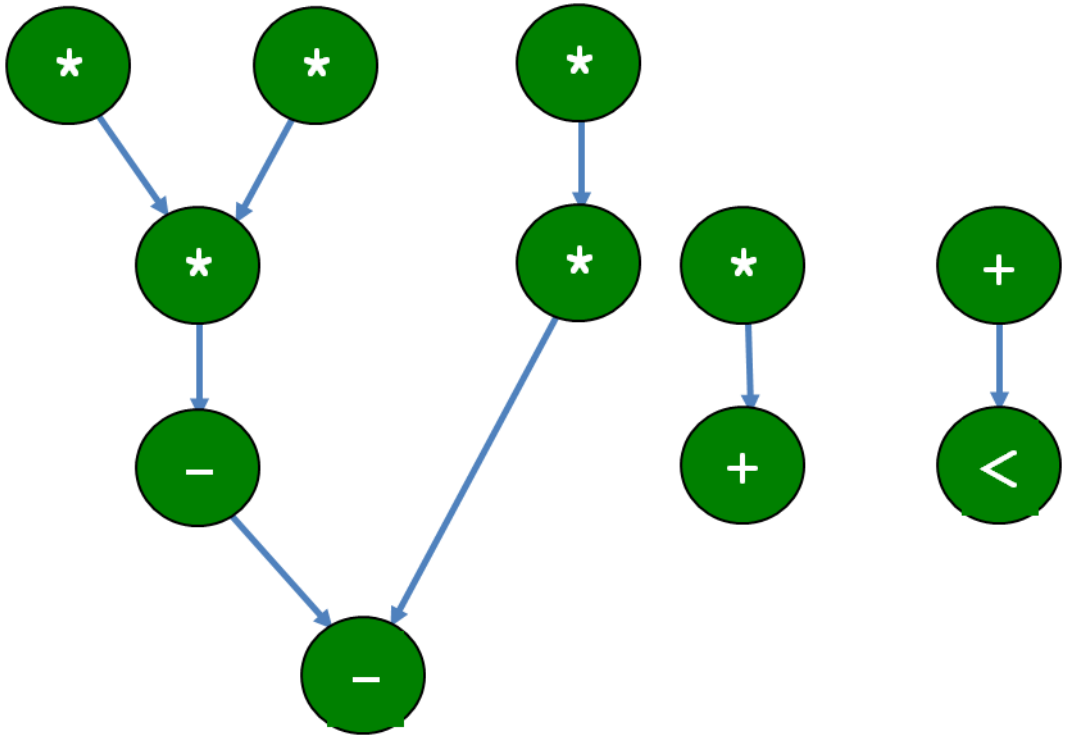
Definition:

Ein **Ressourcegraph** $G_R(V_R,E_R)$ ist ein bipartiter Graph. Die Knotenmenge $V_R = V \cup V_T$ enthält die Knotenmenge des Problemgraphen G(V,E). Jeder Knoten $r_k \in V_T$ stellt einen Ressourcetypen dar (z.B. Prozessor, ALU, Addierer, Multiplizierer, Speicher, Bus etc.). Eine Kante $(v_i,r_k) \in E_R$ mit $v_i \in V$ und $r_k \in V_T$ modelliert die Realisierbarkeit von v_i auf einer Instanz des Ressourcetypen r_k. Ferner gibt es eine

- Kostenfunktion $c: V_T \rightarrow Z^+_0$, die jedem Ressourcetyp $r_k \in V_T$ die Kosten $c(r_k)$ zuordnet sowie eine
- Gewichtsfunktion $w:E_R \rightarrow Z^+_0$, die jeder Kante (v_i,r_k) in E_R das Gewicht $w(v_i,r_k)$ abgekürzt $w_{i,k}$ zuordnet, die die Berechnungszeit von v_i auf r_k darstellt.

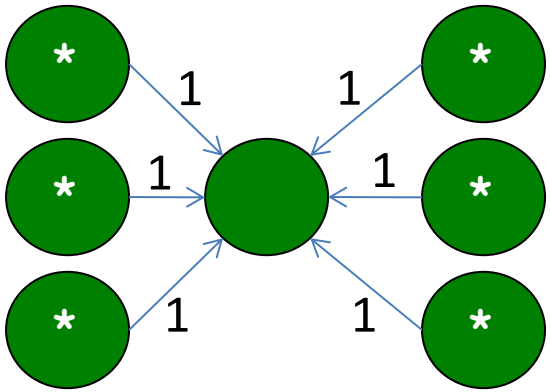
Ressourcegraph

- Ressourcen: 2 Multiplizierer, 2 ALUs (+, –, <)



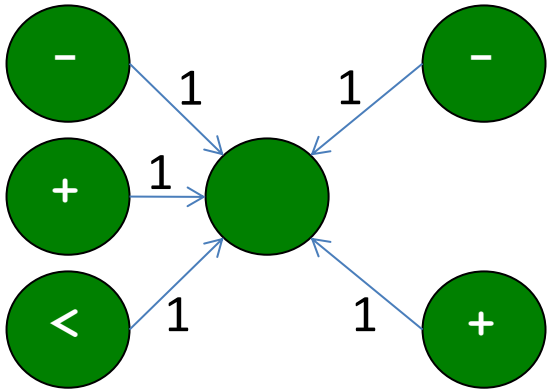
Ressourcegraph

- Ressourcen: 2 Multiplizierer, 2 ALUs (+, –, <)



Multiplizierer

$\alpha(r_1)=2$



ALU

$\alpha(r_2)=2$

Allokation und Bindung

Ein Problemgraph G(V,E) und ein Ressourcegraph $G_R(V_R,E_R)$ stellen zusammen eine Spezifikation $(G(V,E), G_R(V_R,E_R))$ dar.

Definition:

Gegeben sei eine Spezifikation $(G(V,E), G_R(V_R,E_R))$. Eine **Allokation** ist eine Funktion $\alpha : V_T \rightarrow Z^+_0$, die jedem Ressourcentypen $r_k \in V_T$ die Anzahl $\alpha(r_k)$ verfügbarer Instanzen zuordnet.

Definition:

Gegeben sei eine Spezifikation $(G(V,E), G_R(V_R,E_R))$ sowie eine Allokation α. Eine **Bindung** ist ein Paar von Funktionen

- $\beta : V \rightarrow V_T$ mit $\beta(v_i) = r_k \in V_T$ und $(v_i, \beta(v_i)) \in E_R$
- $\gamma : V \rightarrow Z^+$ mit $\gamma(v_i) \leq \alpha(\beta(v_i))$.

Implementierung

Definition:

Gegeben sei eine Spezifikation ($G(V,E)$,$G_R(V_R,E_R)$) (und evtl. eine Latenzschranke L‘ und/oder eine Allokation α). Eine (gültige) **Implementierung** ist ein Quadrupel (T, β, γ, α) bestehend aus Ablaufplan T, Bindung , und Allokation, das die folgenden Bedingungen erfüllt:

- $L \leq L'$ im Falle einer gegebenen Latenzbeschränkung und
- Für alle $r_k \in V_T$ und für alle $t \in [\min_{v_i \in V} \{T(v_i)\}, \max_{v_i \in V} \{T(v_i)+d_i\}]$: $|\{v_i : \beta(v_i)= r_k \text{ und } T(v_i) \leq t \leq T(v_i)+d_i\}| \leq \alpha(r_k)$ im Fall von Ressourcenbeschränkungen

Ohne Ressourcenbeschränkung

- Einfacher Fall:
  - Es gibt genau eine Ressource je Aufgabe, d.h. für $v_i \in V$ gibt es genau einen Ressourcentyp also genau eine Kante $e= (v_i,r_k) \in E_R$.
  - Dann ist die Berechnungszeit $d_i= w(v_i,r_k)$.

- Keine Ressourcenbeschränkung liegt vor, falls für jede Aufgabe mindestens eine Instanz der entsprechenden Ressource verfügbar ist, d.h.

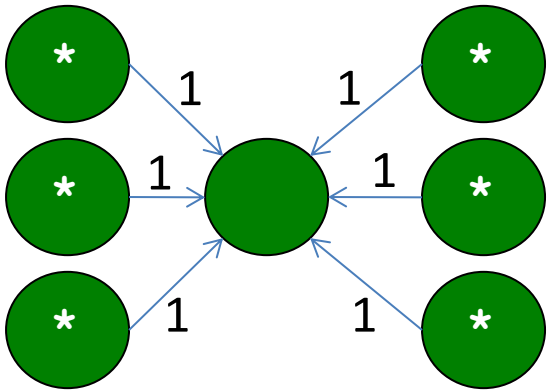
$$\alpha(r_k) \geq |\{e : e= (v_i,r_k) \in E_R \text{ und } v_i \in V \}|$$

Ressourcenbeschränkungen

- Erweitertes ASAP, ALAP
  - zuerst ASAP oder ALAP
  - dann Verschiebung von Tasks nach vorne (ALAP) oder nach hinten (ASAP), bis die Ressourcenbeschränkungen erfüllt sind
- Listscheduling
  - Tasks werden nach einem bestimmten Kriterium in eine Prioritätsliste eingereiht
  - in jedem Zeitschritt wird jeder freien Ressource diejenige ausführbare Operation zugeteilt, die die höchste Priorität hat
  - Kriterien: Anzahl der Nachfolgerknoten, Mobilität, ...

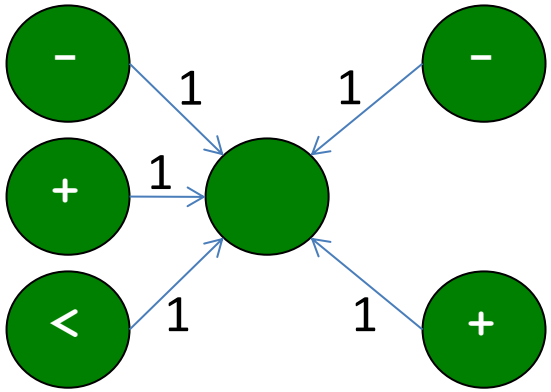
Ressourcegraph

- Ressourcen: 2 Multiplizierer, 2 ALUs (+, –, <)



Multiplizierer

$\alpha(r_1)=2$

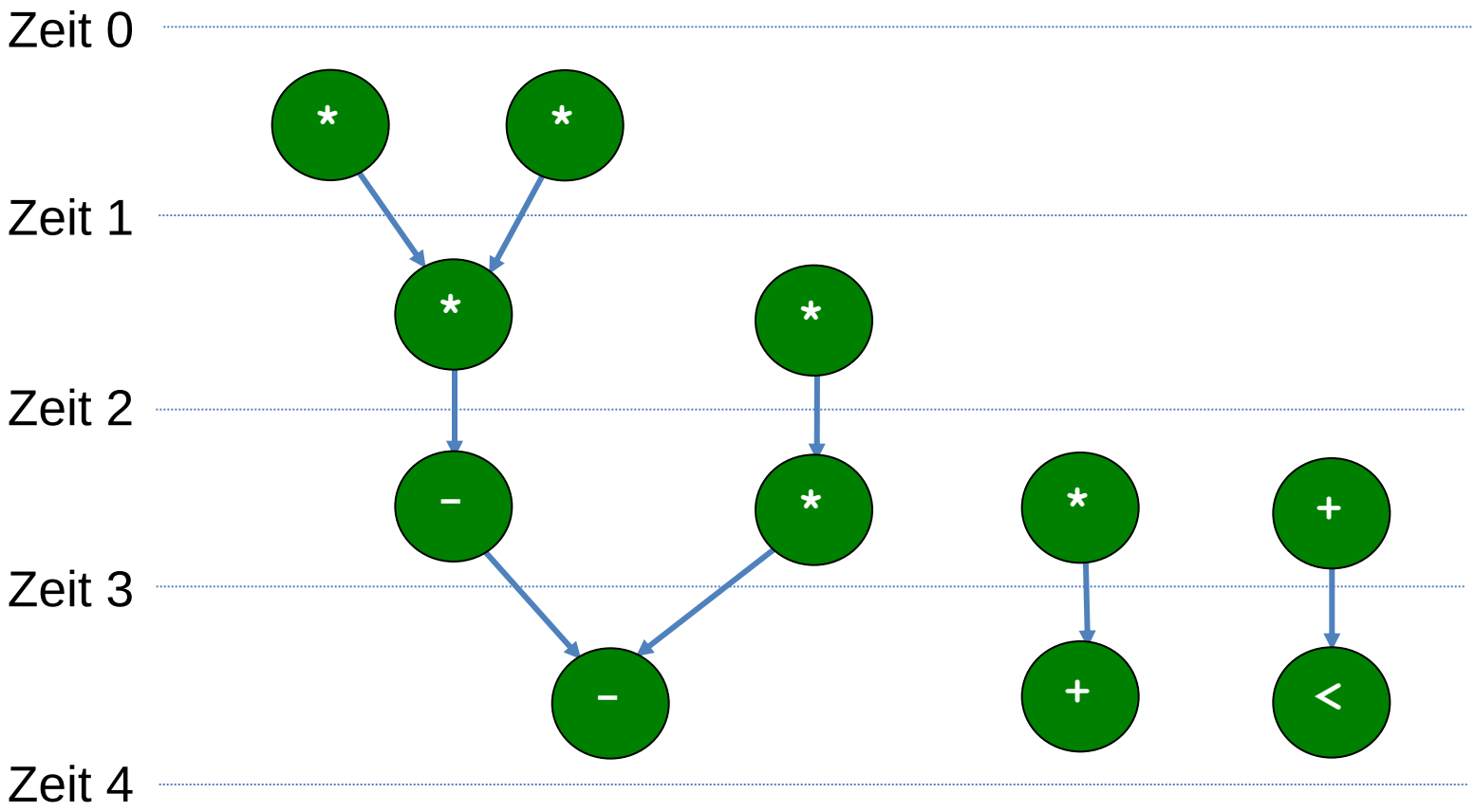


ALU

$\alpha(r_2)=2$

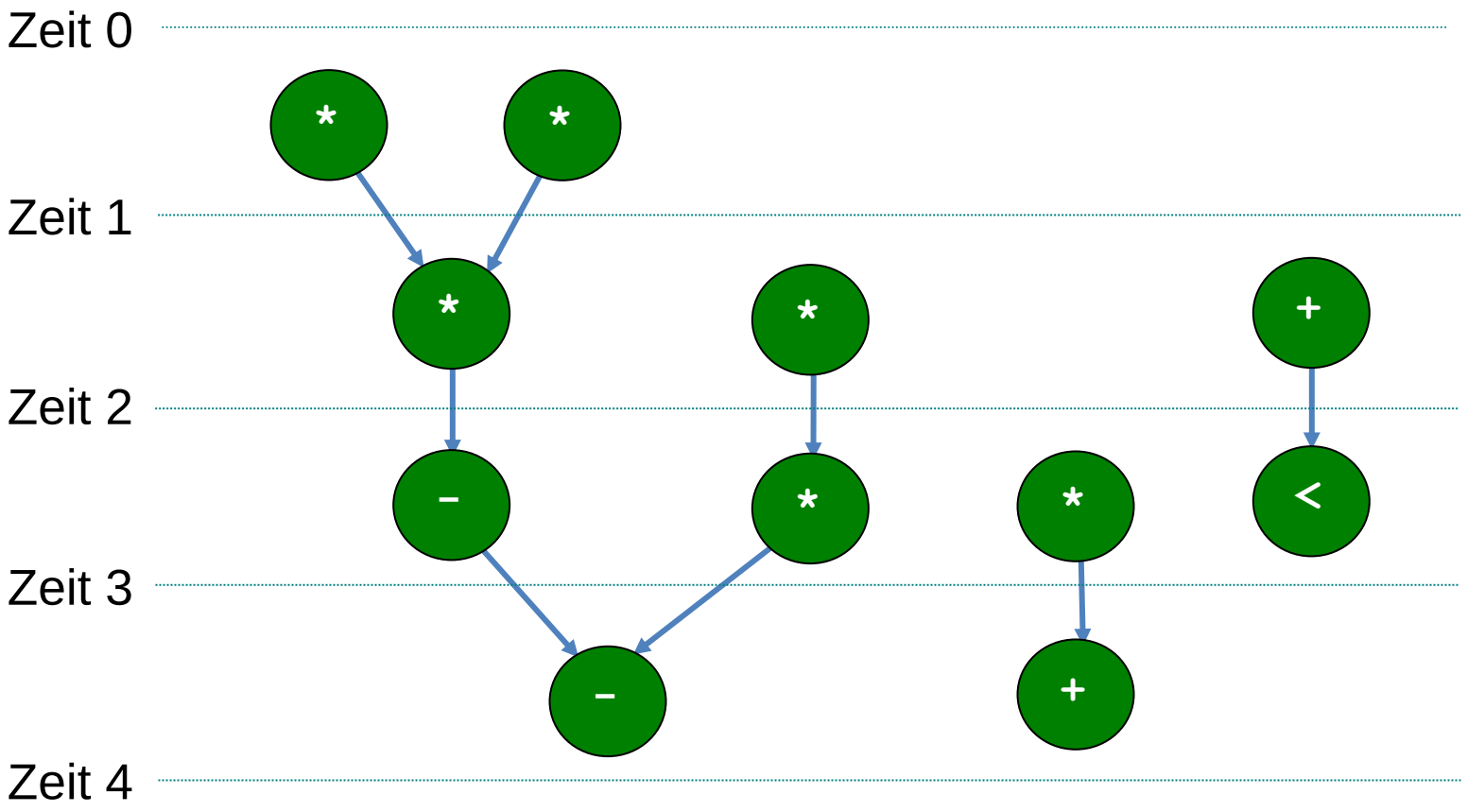
Erweitertes ALAP

- 2 Multiplizierer, 2 ALUs (+, -, <)
- Vorherige Lösung:



Erweitertes ALAP

- 2 Multiplizierer, 2 ALUs (+, -, <)



Listscheduling (1)

- Kandidaten:
  $K_{t,k} = \{v_i \in V : \beta(v_i) = r_k$ und für alle $j : (v_j, v_i) \in E : t \geq T(v_j)+d_j\}$

- Operationen, die zum Zeitpunkt t noch laufen:
  $G_{t,k} = \{v_i \in V : \beta(v_i) = r_k$ und $T(v_i)+d_i \geq t \geq T(v_i)\}$

- Prioritäten unter den Operationen:

  $p : V \rightarrow Z^+_0$

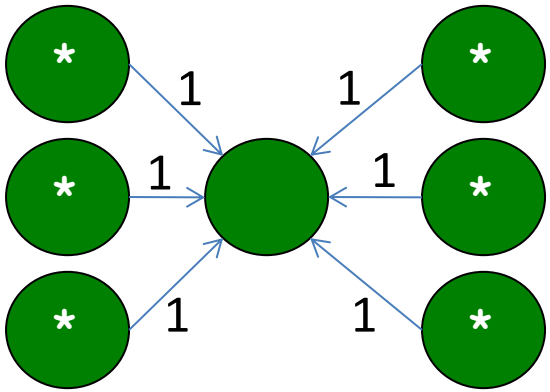
- Prioritäten können zum Beispiel aus Mobilität, Länge der Datenpfade etc. abgeleitet werden

Listscheduling (2)

LIST($G(V,E)$, $G_R(V_R,E_R)$, α, p) {
    $t := 0$; REPEAT {
        FOR $k := 1$ TO $|V_T|$ {
            Bestimme Kandidatenmenge $K_{t,k}$;
            Bestimme Menge nicht beendeter
                Operationen $G_{t,k}$;
            Wähle eine Menge maximaler Priorität $S_t \subseteq K_{t,k}$:
                $|S_t| + |G_{t,k}| \leq \alpha(r_k)$
            FOREACH ($v_i \in S_t$) $T(v_i) := t$
        }
        $t := t+1$;
    } UNTIL (alle Knoten v_i geplant)
    return T;
}

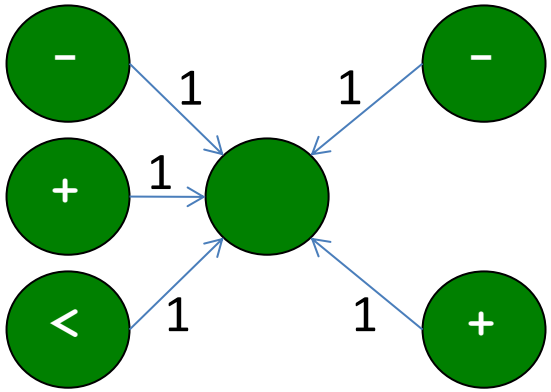
Listscheduling (3)

- Ressourcen: 1 Multiplizierer, 1 ALU (+, –, <)



Multiplizierer

$\alpha(r_1)=1$

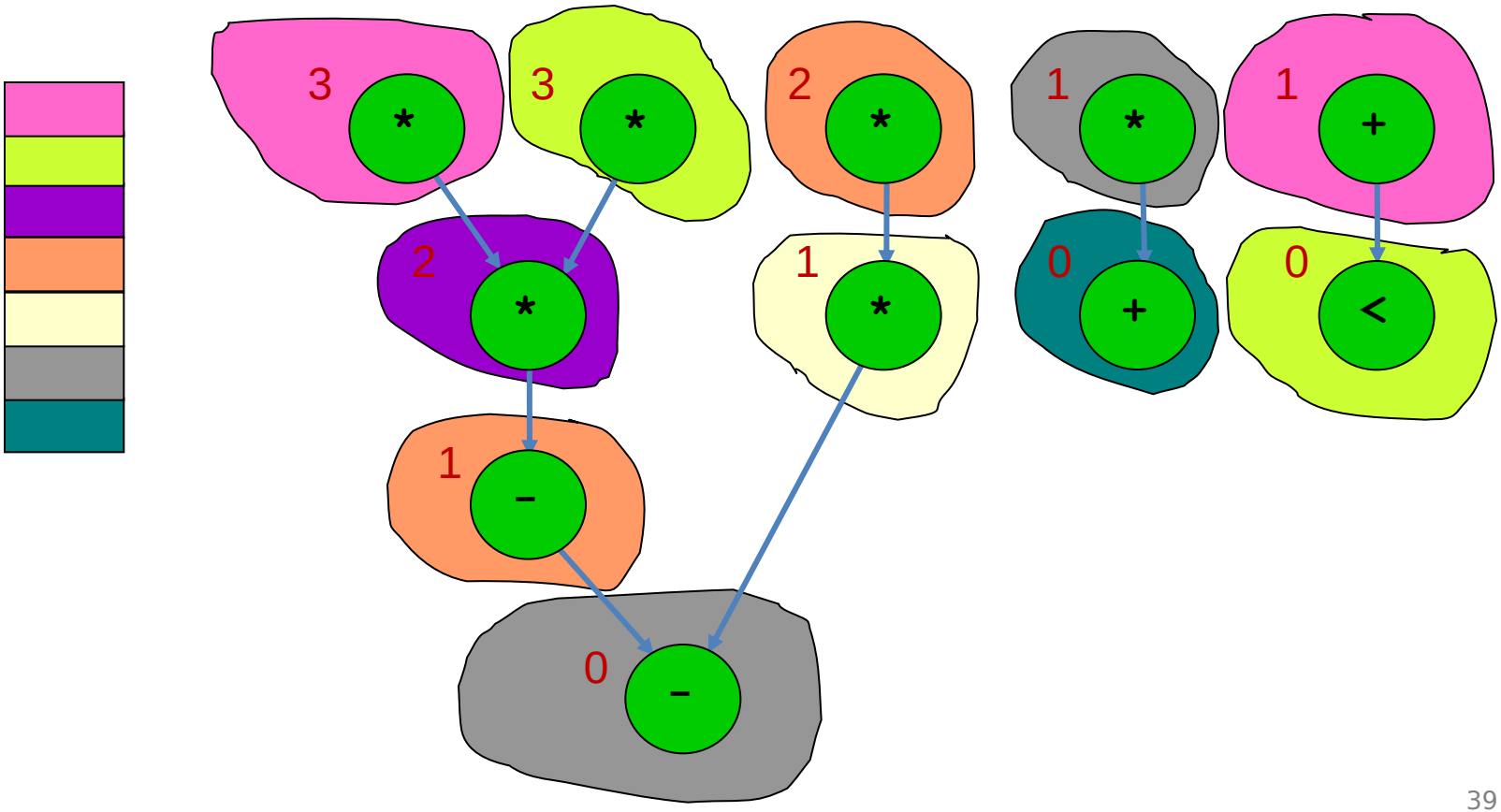


ALU

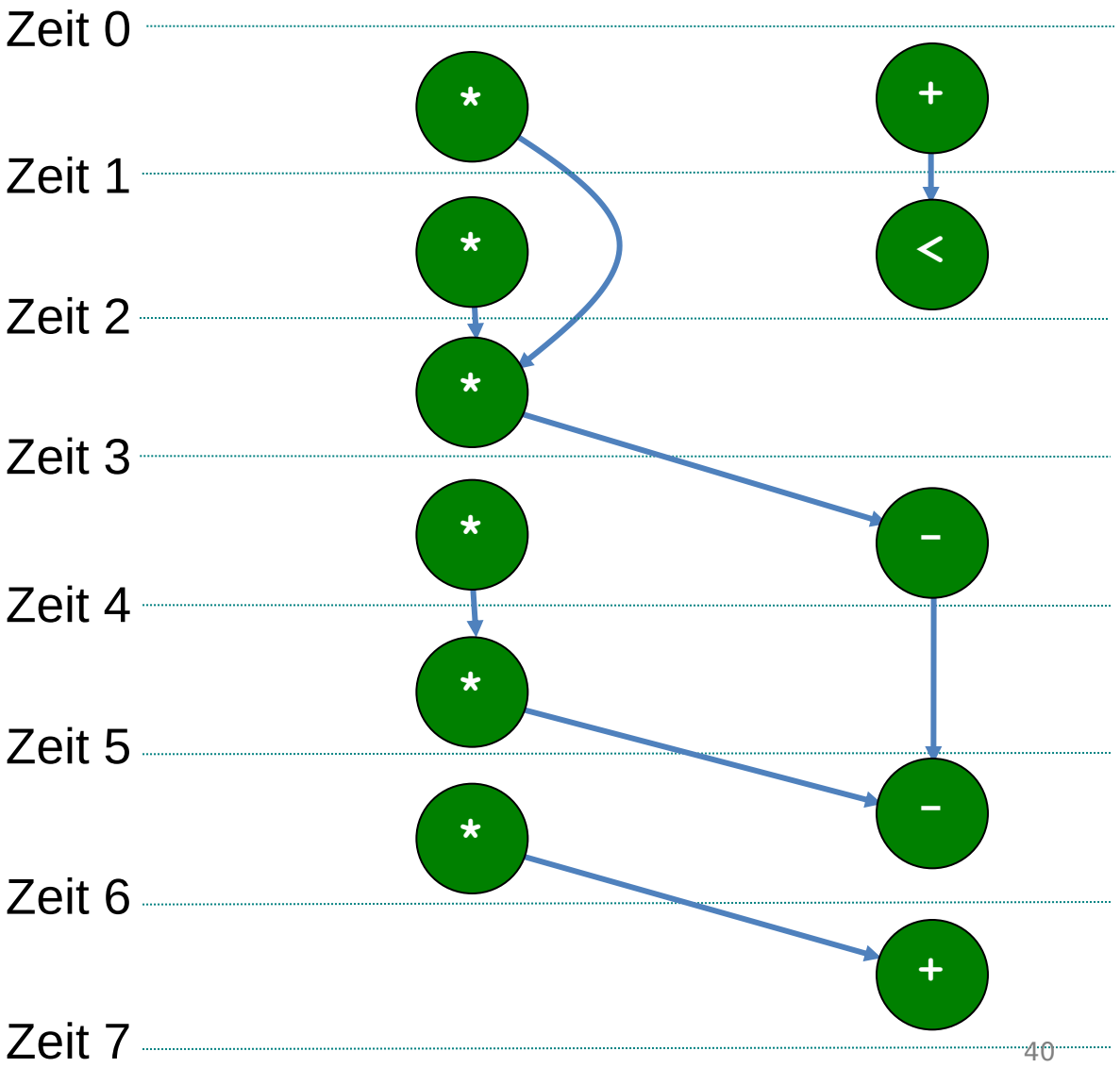
$\alpha(r_2)=1$

Listscheduling (4)

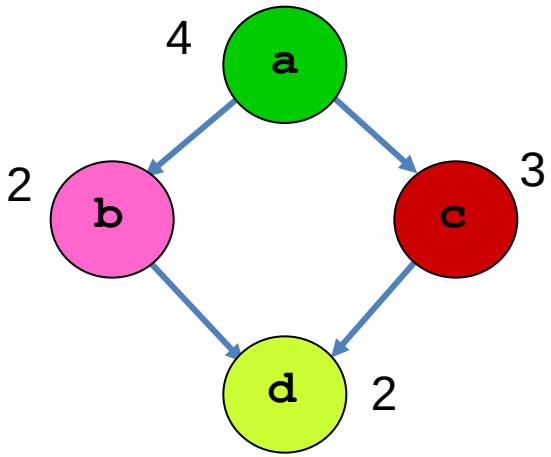
- Kriterium: Anzahl der Nachfolgerknoten
- Ressourcen: 1 Multiplizierer, 1 ALU (+, -, <)



Listscheduling (5)

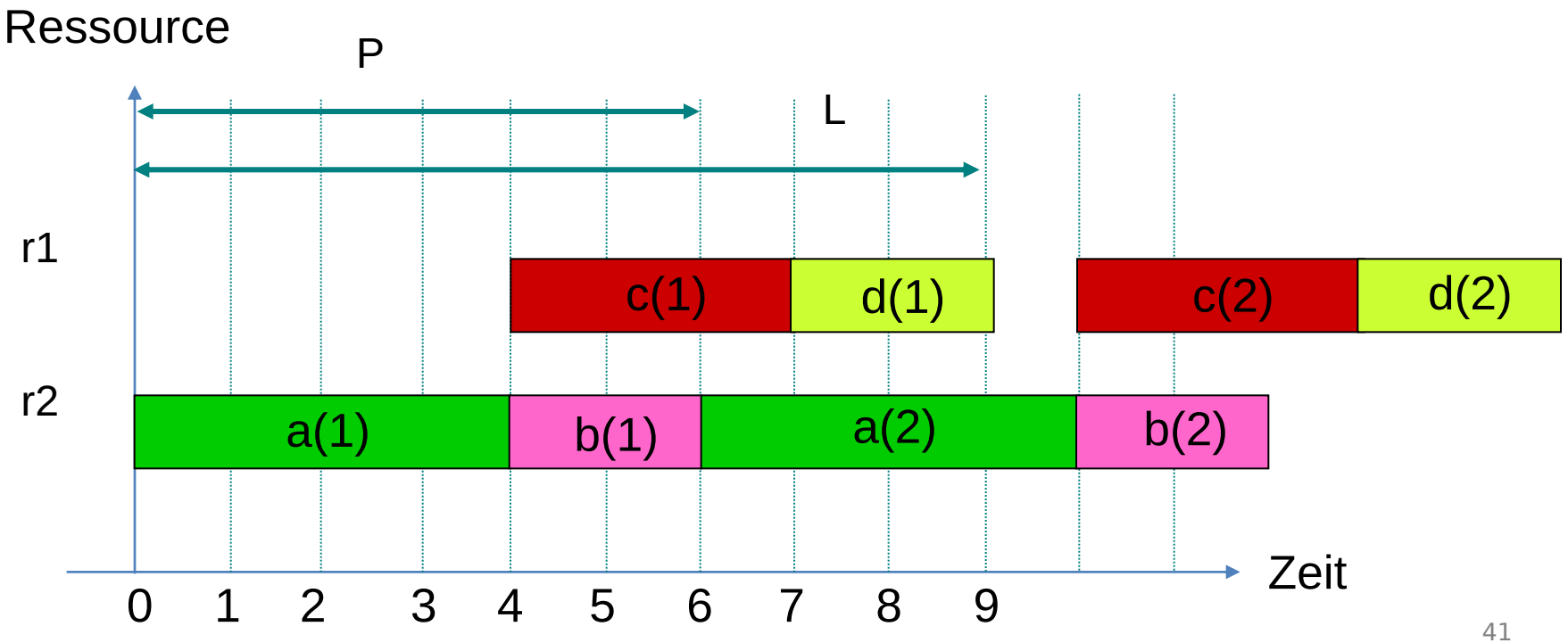


Periodische Ablaufpläne



Periode: P = 6

Latenz: L = 9



Modelle für die Systemsynthese

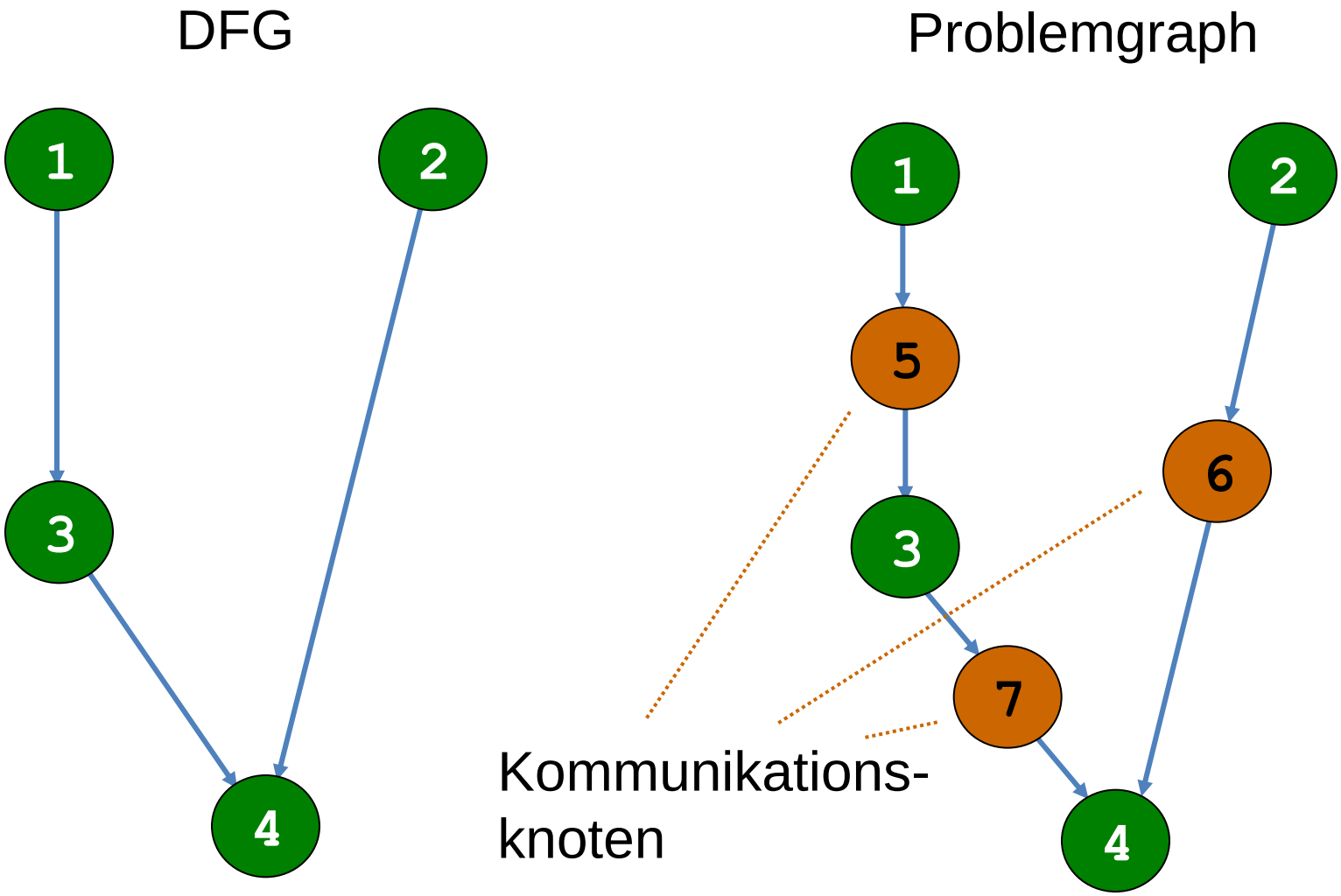
- Allokation + Bindung = Partitionierung

- Problemgraph
  - Knoten: funktionale Objekte und Kommunikationsobjekte
  - Kanten: Abhängigkeiten

- Architekturgraph
  - Knoten: funktionale Ressourcen und Kommunikationsressourcen
  - Kanten: gerichtete Kommunikationsmöglichkeiten

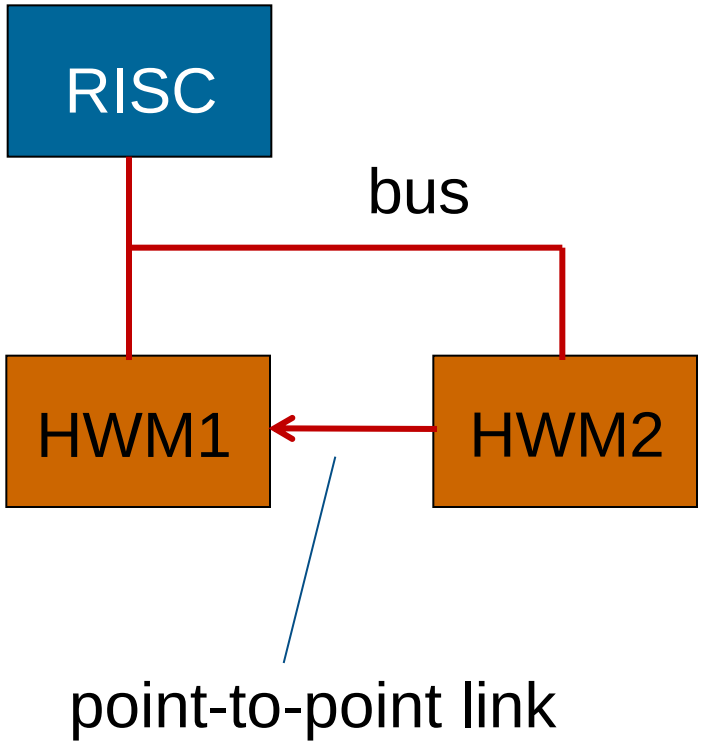
- Spezifikationsgraph
  - Problemgraph + Architekturgraph + Abbildungsmöglichkeiten

Problemgraph

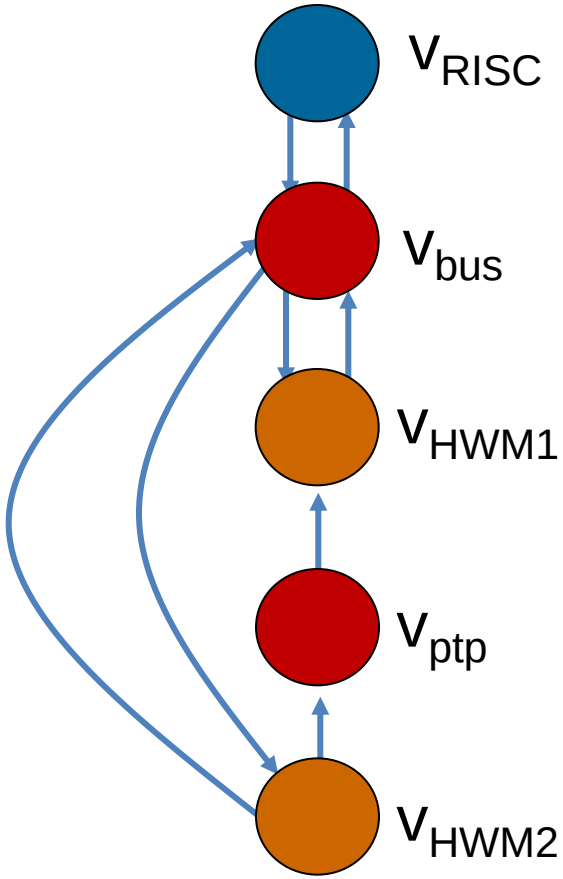


Architekturgraph

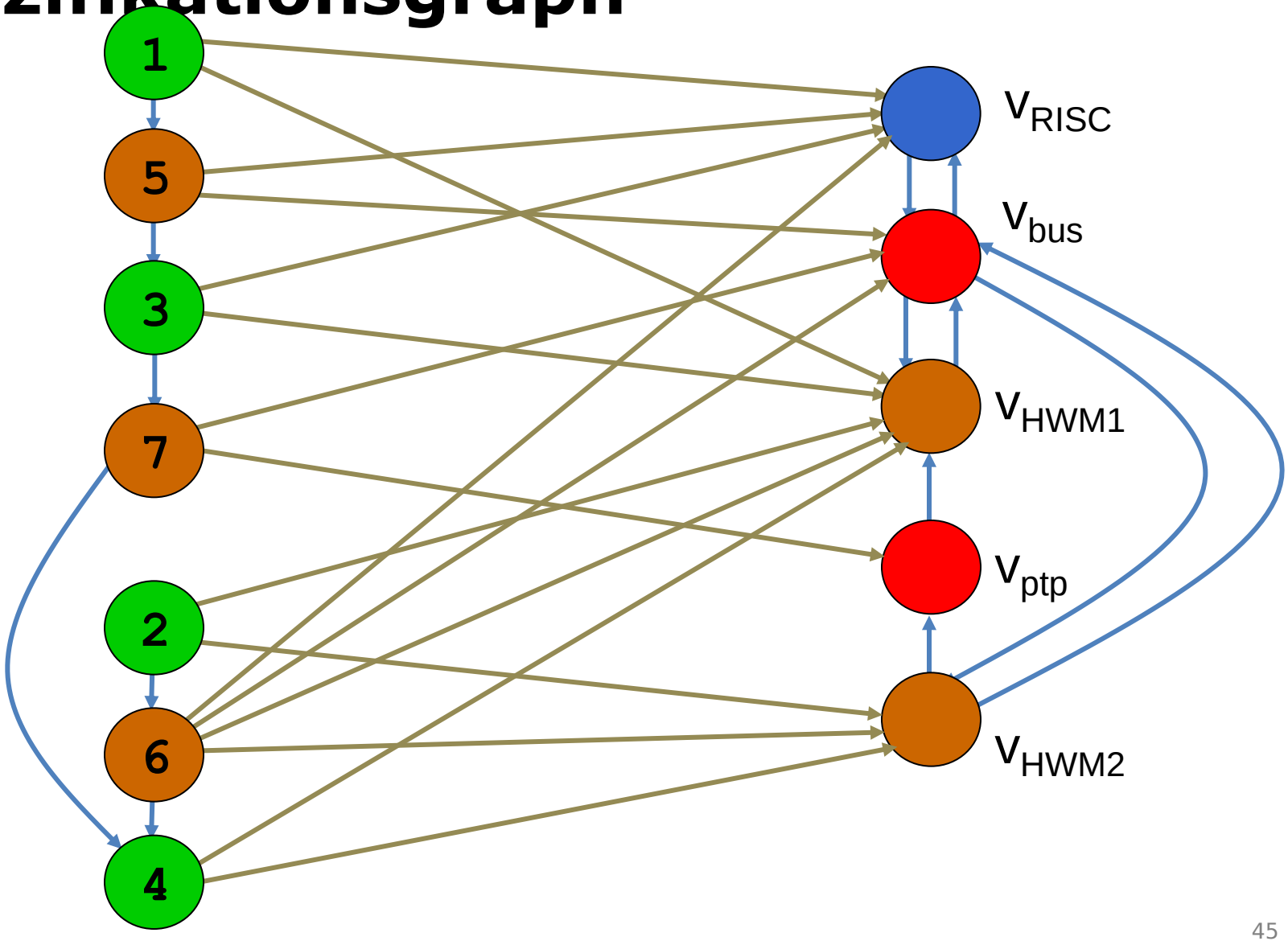
Architektur



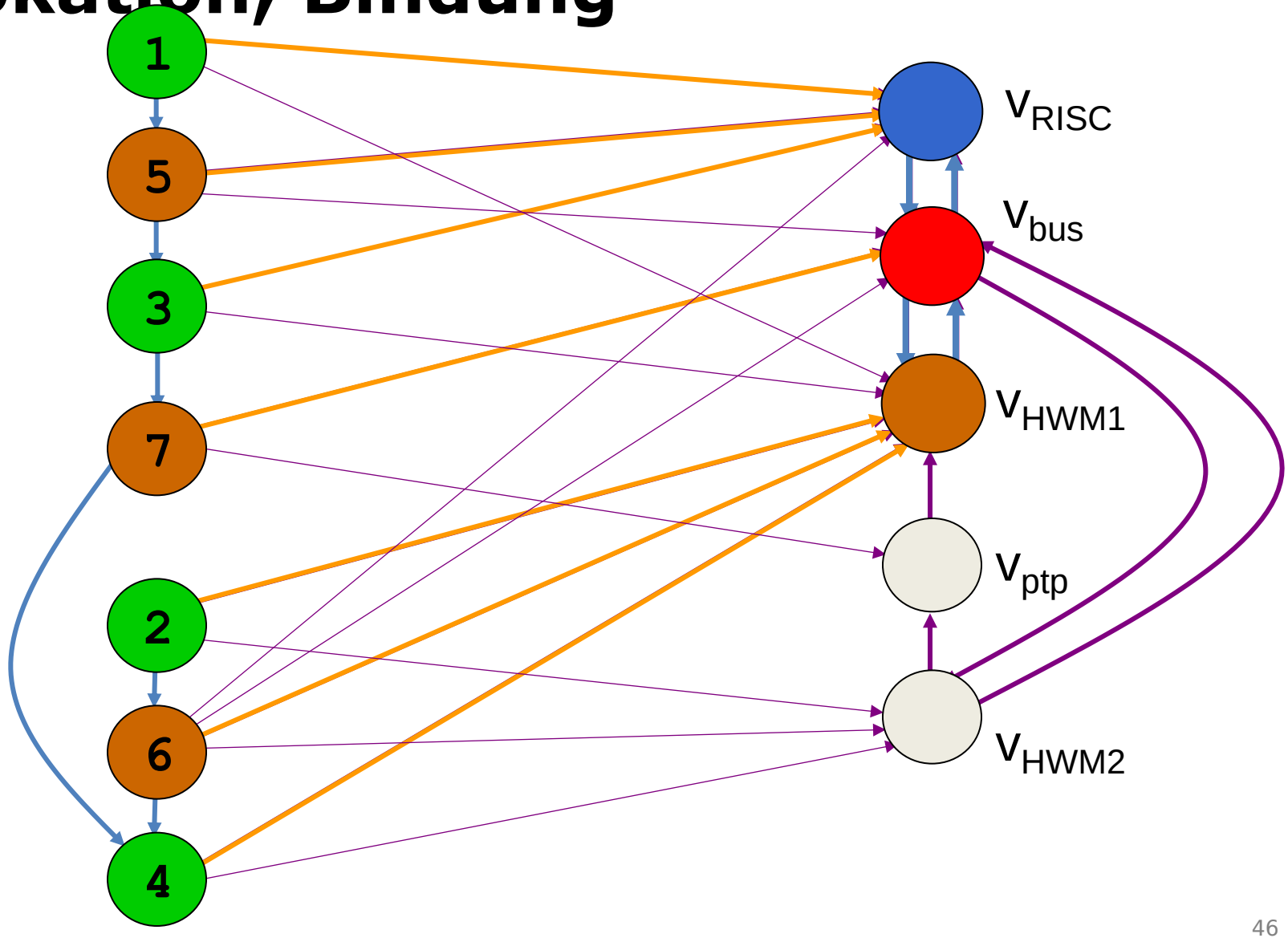
Architekturgraph



Spezifikationsgraph

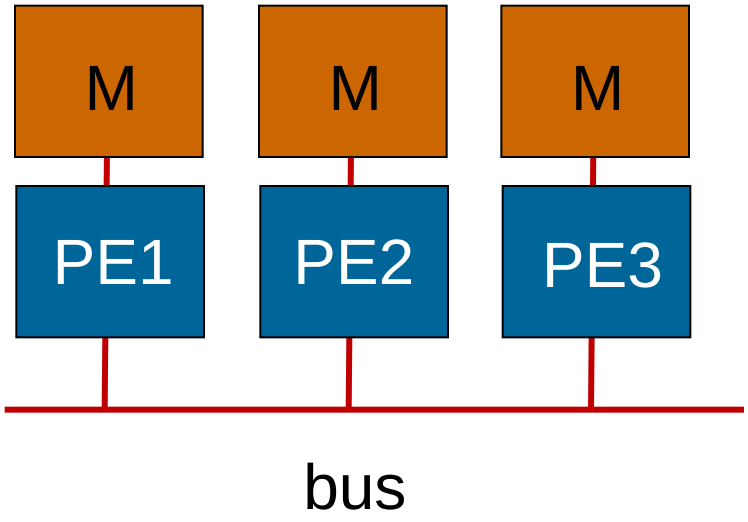


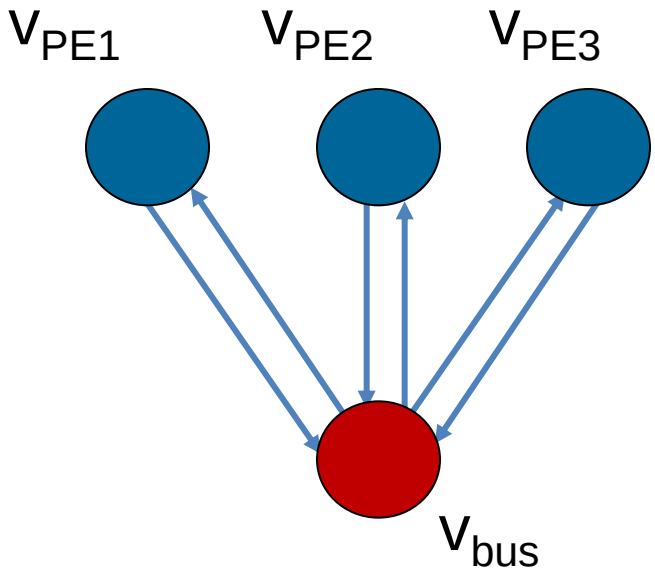
Allokation, Bindung



Bsp.: homogener Multiprozessor

- Allokation gegeben
- Bindung und Ablaufplan gesucht mit
  - Einhaltung von Deadlines oder
  - minimaler Latenz





Bsp.: Hw/Sw Partitionierung

- im einfachsten Fall nur zwei Blöcke: Sw und Hw (Bipartitionierung)