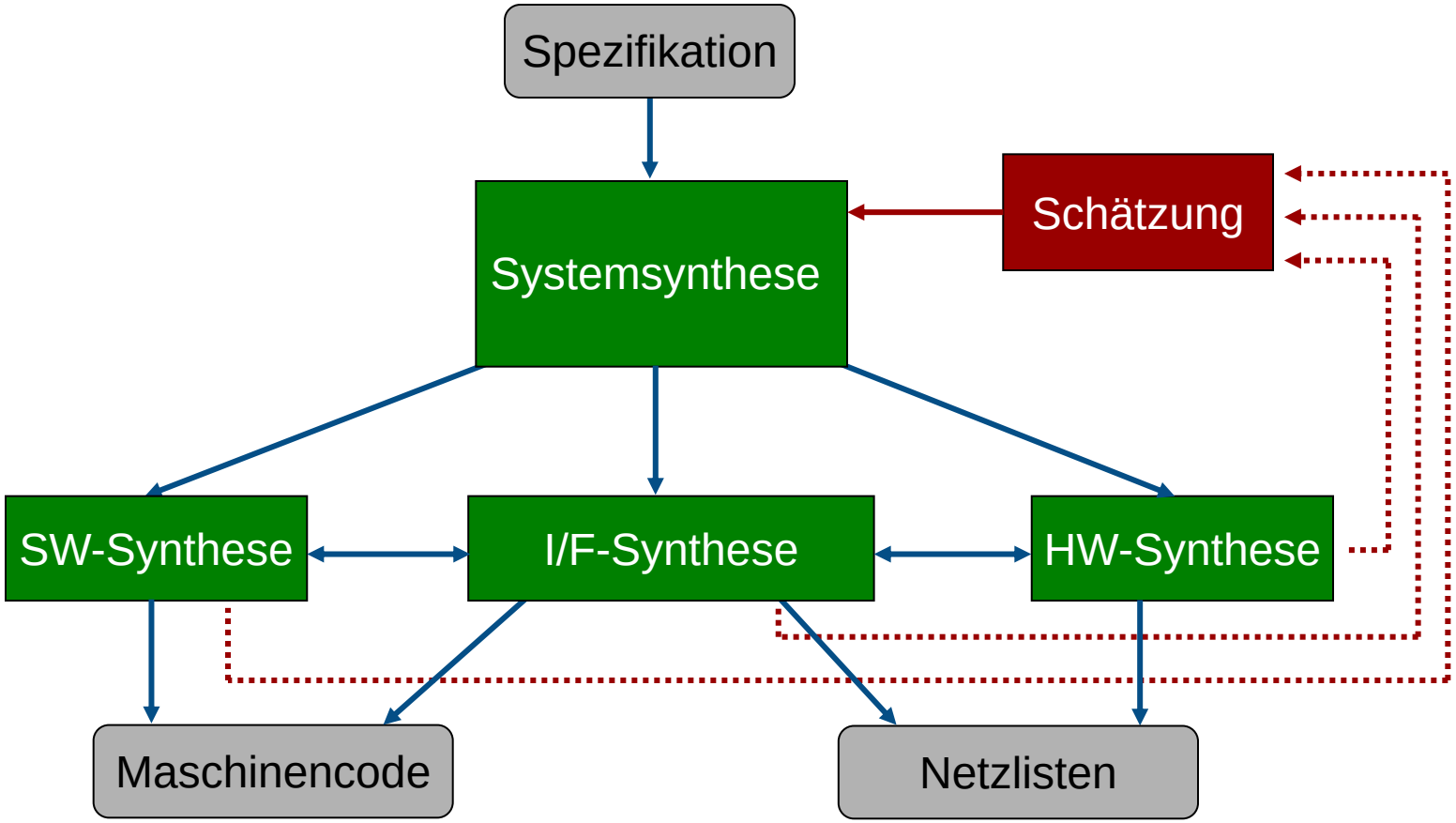
Einführung in die Technische Informatik

Schätzung der Entwurfsqualität

Robert Wille

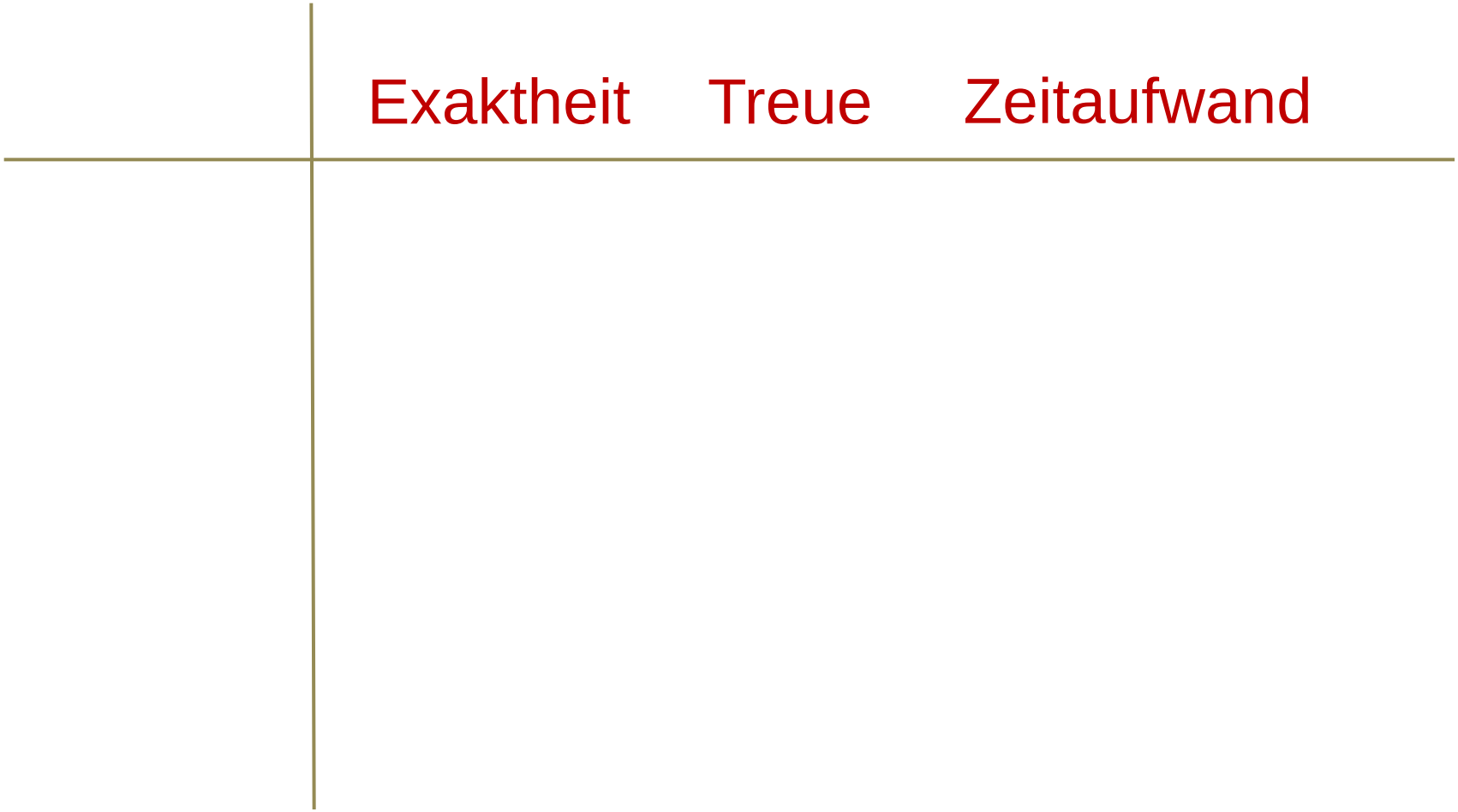
Systementwurf



Überblick

- Parameter von Schätzverfahren
- Qualitätsmaße
- Abschätzung von Hardware
- Abschätzung von Software

Parameter von Schätzverfahren



Exaktheit

- Definition: Sei E(D) eine abgeschätzte und M(D) die exakte (gemessene) Metrik einer Implementierung D. Die Exaktheit A der Abschätzung ist gegeben durch:

$$A = 1 - \frac{|E(D) - M(D)|}{M(D)}$$

Treue

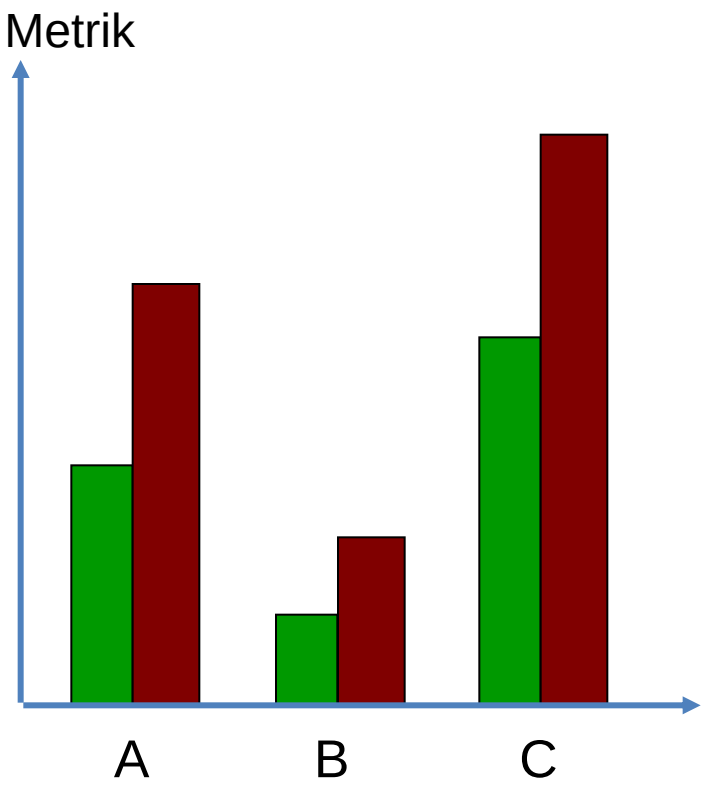
- Definition: Sei D = {D_1, D_2, …, D_n} eine Menge von Implementierungen. Die Treue F einer Schätzmethode ist die Prozentzahl der korrekten Abschätzungen

$$F = 100 \cdot \frac{2}{n(n-1)} \cdot \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mu_{i,j}$$

$$\mu_{i,j} = \begin{cases} 1 & if \quad (E(D_i) > E(D_j) \wedge M(D_i) > M(D_j)) \vee \\ & \quad (E(D_i) < E(D_j) \wedge M(D_i) < M(D_j)) \vee \\ & \quad (E(D_i) = E(D_j) \wedge M(D_i) = M(D_j)) \\ 0 & else \end{cases}$$

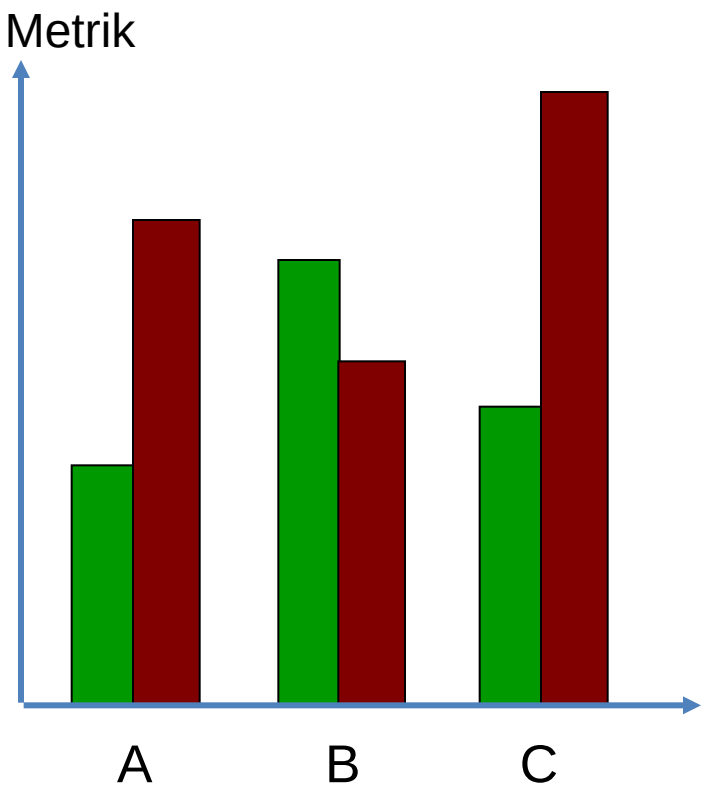
Treue - Beispiel

Treue = 100 %

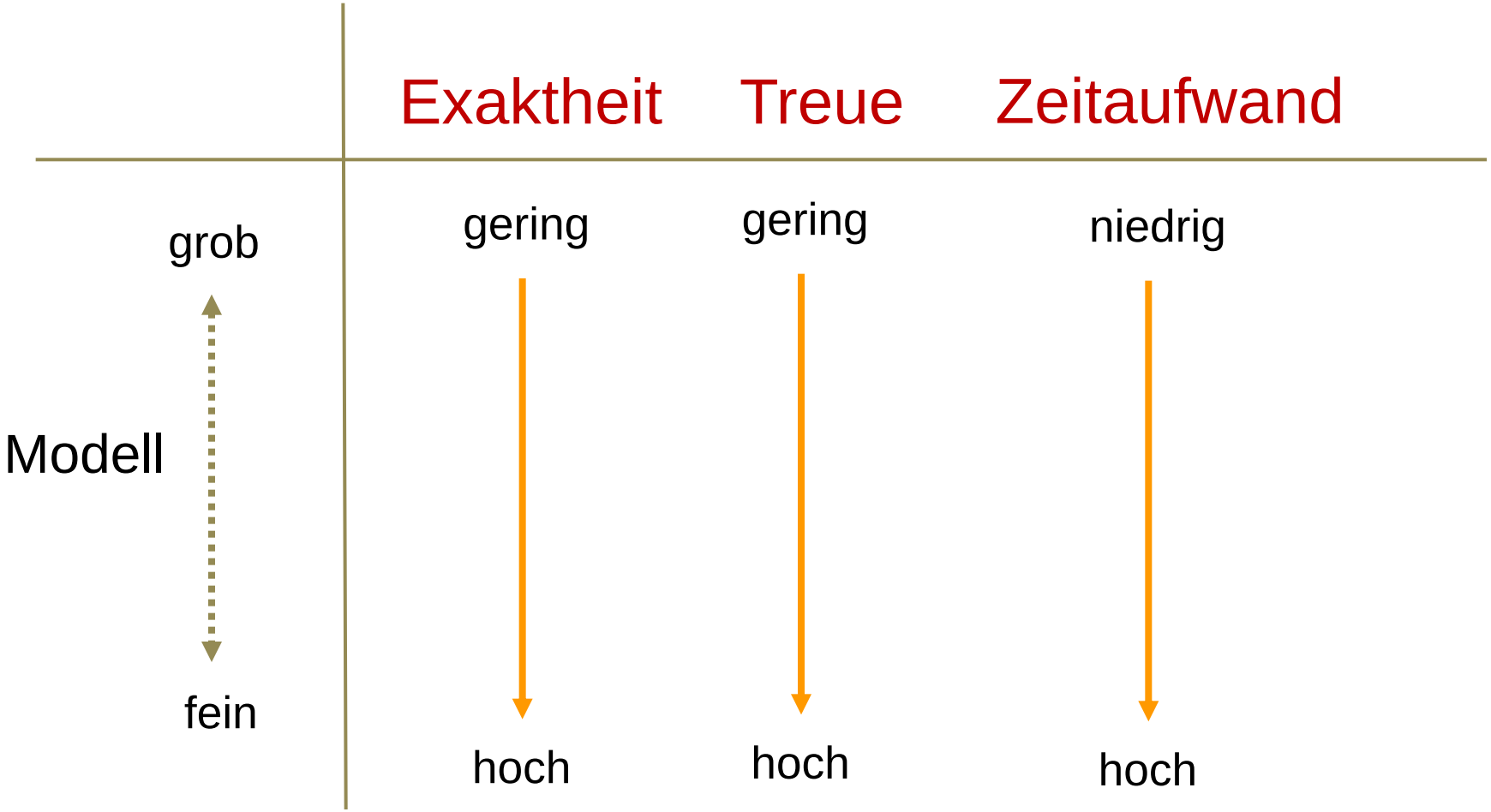


Implementierung

Treue = 33.3 %



Parameter von Schätzverfahren



Qualitätsmaße

- Performance
  - Hardware
    - Taktperiode, Latenz, Ausführungszeit, Datenrate
  - Software
    - Ausführungszeit
  - Kommunikation
    - Maximale, mittlere Bitrate
- Kosten
- weitere Maße
  - Leistungsaufnahme
  - Testbarkeit
  - etc.

HW Performance

- Taktperiode *T*
  - Hängt mit der Technologie, der Ausführungszeit, und den Ressourcen zusammen
- Latenz *L*
  - Anzahl der Taktschritte
- Ausführungszeit

$$T_{ex} = T * L$$

- Datenrate (Durchsatz) *R*
  - Pipelining mit *P* gleich langen Stufen:

$$R = P/T_{ex}$$

SW Performance

- Ausführungszeit
  - Profiling: Compilation und möglichst viele Testläufe
    → statistische Aussagen
  - Schätzung auf Basis des Quell- / Zwischen- / Zielcodes
  - Wichtig bei Echtzeitsystemen mit harten Zeitschranken:
    Worst-Case Execution Time (WCET)
- Speicherbedarf
  - Programmspeicher: Compilation, Schätzung
  - Datenspeicher (nur für statische Speicherallokation):
    Compilation, Schätzung

Kostenmaße

- Hardware
  - Maße proportional zu Halbleiterfläche:
    - mm^2, l^2 (l = feature size der Technologie)
    - Anzahl von Transistoren, Anzahl von Gattern
    - Anzahl von CLBs (FPGAs)
  - Anzahl von Pins

- Software
  - Größe von Programm- und Datenspeicher

HW Performance

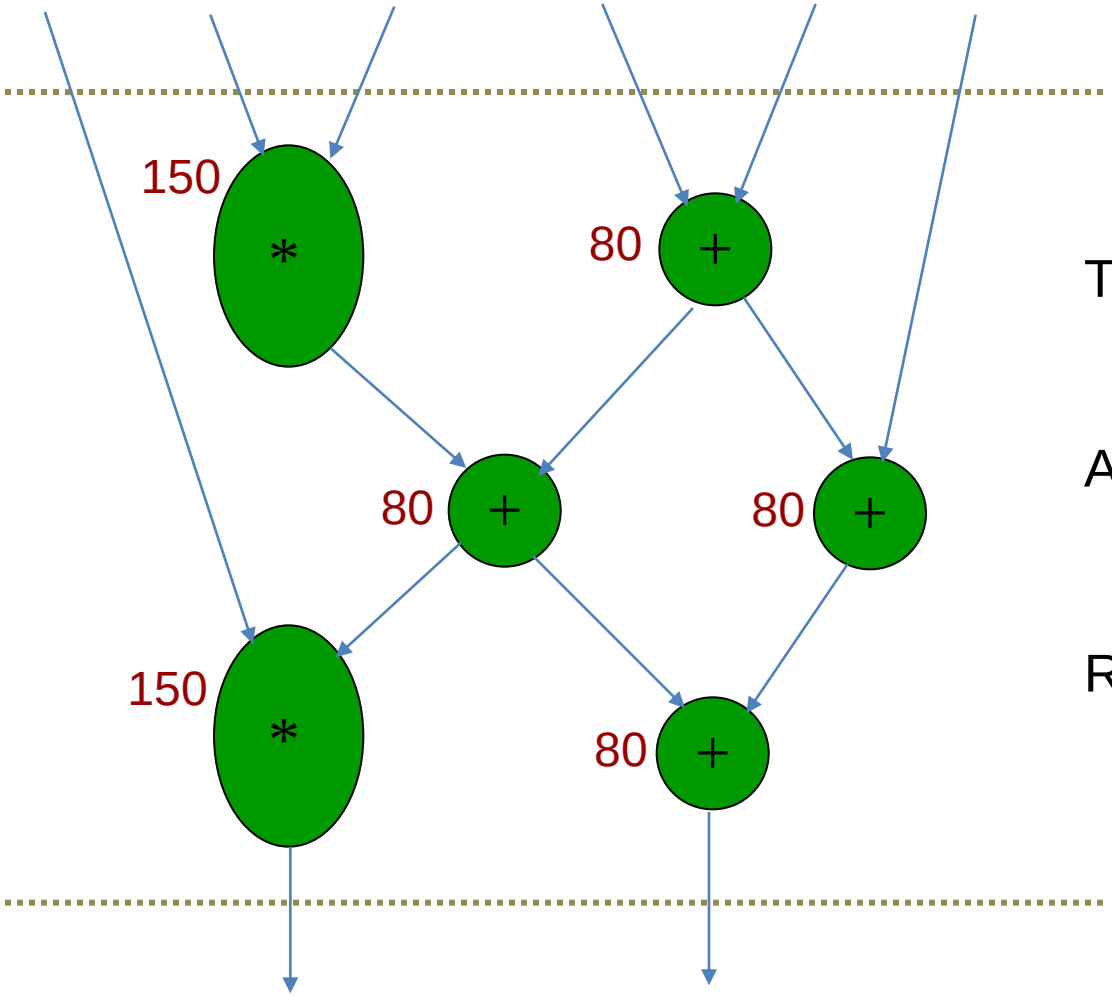
- Taktperiode *T*
  - Hängt mit der Technologie, der Ausführungszeit, und den Ressourcen zusammen
- Latenz *L*
  - Anzahl der Taktschritte
- Ausführungszeit

$$T_{ex} = T * L$$

- Datenrate (Durchsatz) *R*
  - Pipelining mit *P* gleich langen Stufen:

$$R = P/T_{ex}$$

Beispiel (1)

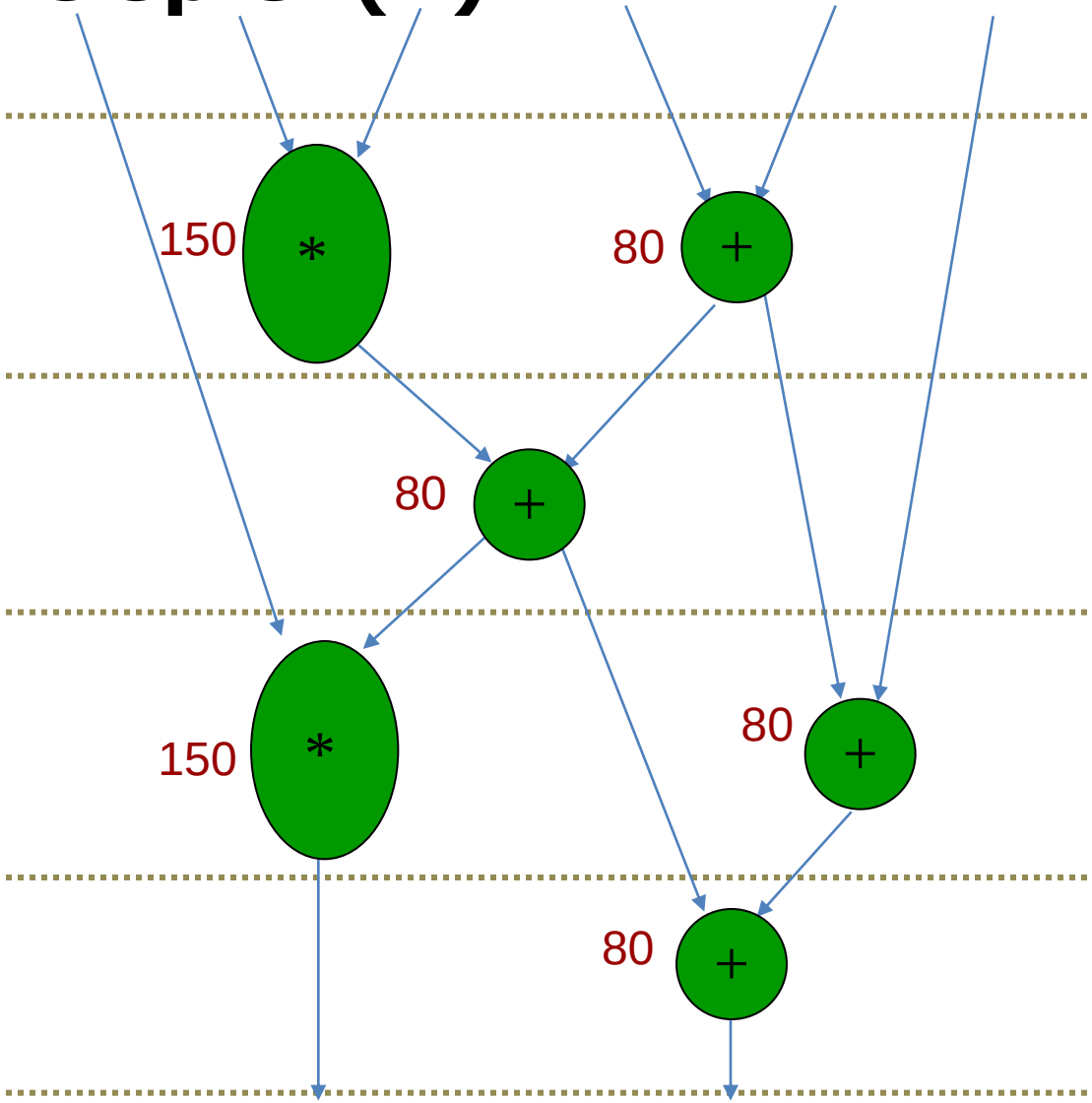


Taktperiode T = 380 ns

Ausführungszeit T_{ex} = 380 ns

Ressourcen: 2 *, 4 +

Beispiel (2)

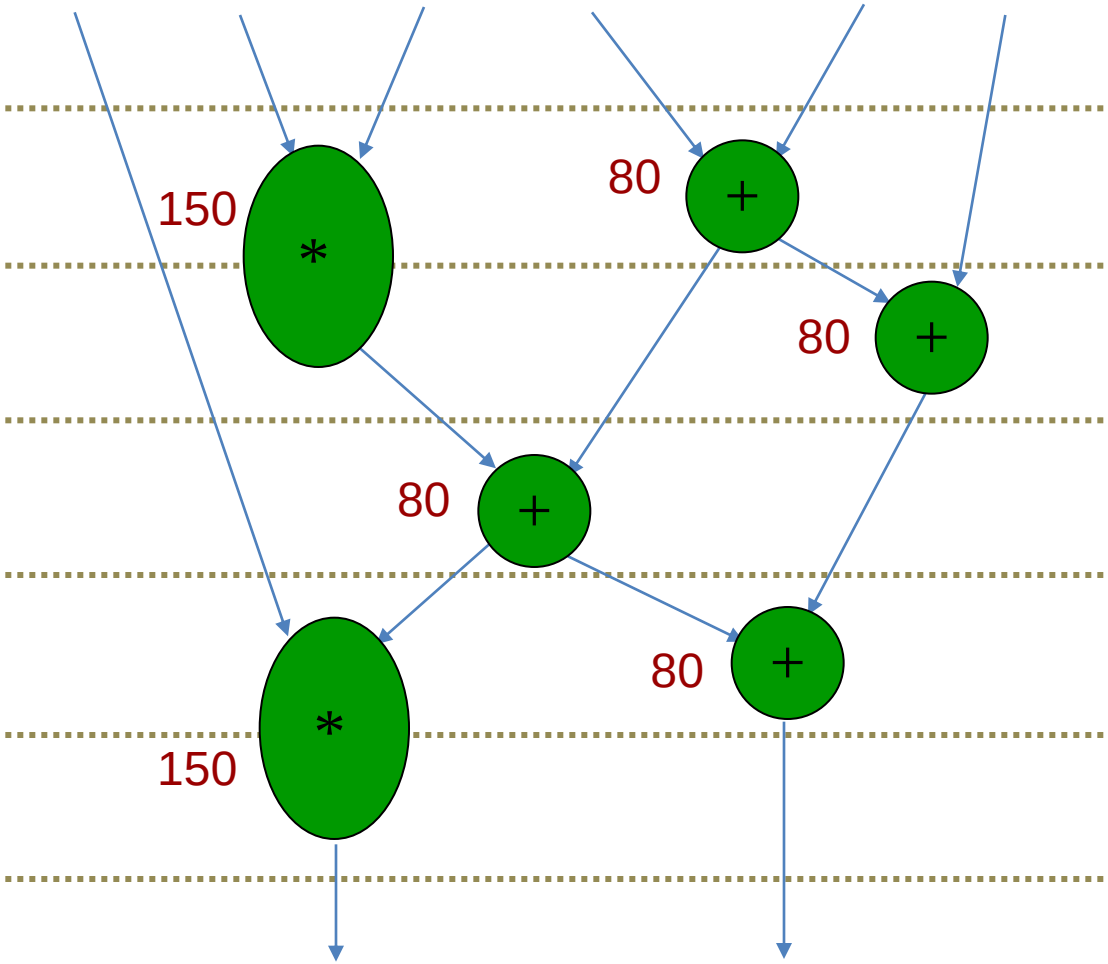


Taktperiode T = 150 ns

Ausführungszeit T_{ex} = 600 ns

Ressourcen: 1 *, 1 +

Beispiel (3)

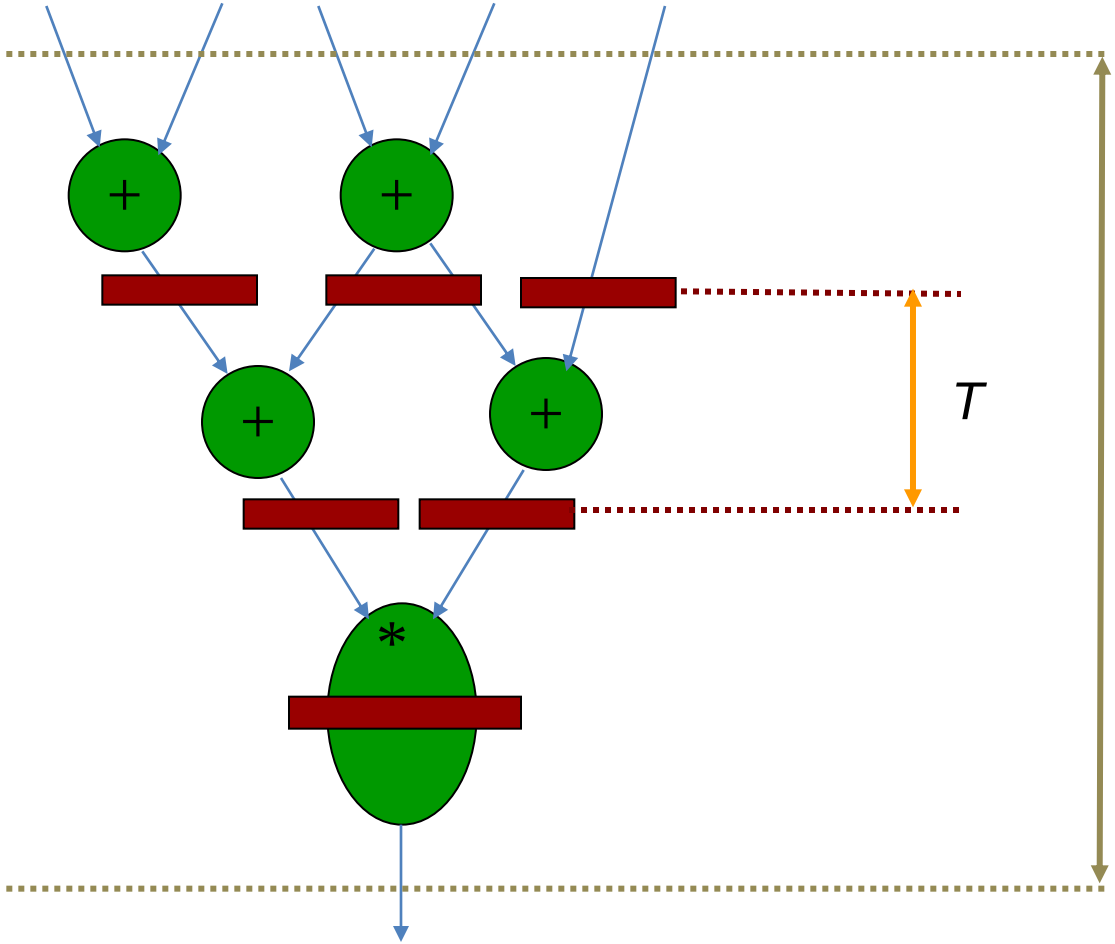


Taktperiode T = 80 ns

Ausführungszeit T_{ex} = 400 ns

Ressourcen: 1 *, 1 +

Pipelining



$R = 4/T_{ex}$

Taktperiode

- Funktionale Einheiten v_k mit Verzögerungszeit $del(v_k)$
- Methode der maximalen Operatorverzögerung

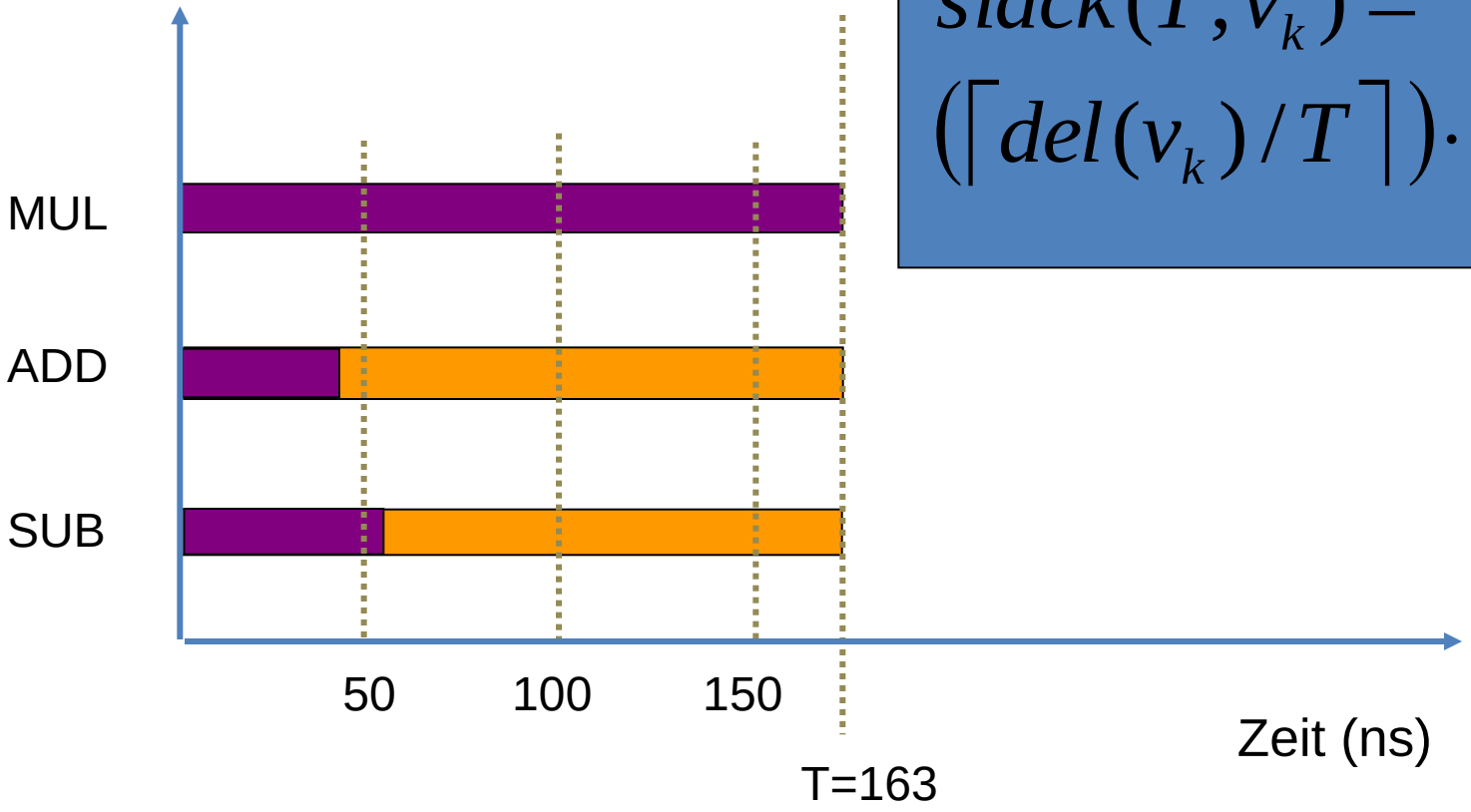
$$T = \max_k \left(del(v_k) \right)$$

Taktschlupf

belegte Einheit

Schlupf

$$slack(T, v_k) = \left(\lceil del(v_k)/T \rceil\right) \cdot T - del(v_k)$$



Taktschlupfminimierung

- Mit $occ(v_k)$, der Anzahl der Operationen vom Typ k, und $|V_T|$, der Anzahl unterschiedlicher Operationstypen, ist der mittlere Schlupf:

$$avgslack(T) = \frac{\sum_{k=1}^{|V_T|} (occ(v_k) \cdot slack(T, v_k))}{\sum_{k=1}^{|V_T|} occ(v_k)}$$

- Taktauslastung

$$util(T) = 1 - \frac{avgslack(T)}{T}$$

Taktperiode

- Funktionale Einheiten v_k mit Verzögerungszeit $del(v_k)$
- Methode der maximalen Operatorverzögerung

$$T = \max_k \left(del(v_k) \right)$$

- Methode der Minimierung des Taktschlupfs
  - Suche im Intervall T_{min} … T_{max} nach der Taktperiode T mit maximaler Taktauslastung
- ILP-Suche

SW Performance (1)

- Ausführungszeit *T*

$$T = I_c \times CPI \times t = (I_c \times CPI) / f$$

  - I_c … instruction count eines Programmes
  - *CPI* … cycles per instruction (Mittelwert)
  - *t* … Taktperiode, *f* … Taktfrequenz

- Bsp.:

  $I_c = 2000$, $CPI = 0.4$, $f = 400$ MHz

SW Performance (2)

- MIPS-Rate (million instructions per second)

$$MIPS = I_c / (T \times 10^6) = f / (CPI \times 10^6)$$

Bsp.:

- Prozessor mit einer Taktfrequenz von 500 MHz
- 3 Instruktionsklassen A, B, C mit $CPI_A = 1$, $CPI_B = 2$, $CPI_C = 3$
- 2 verschiedene Compiler erzeugen für das gleiche Programm folgende Instruktionen (x 10^9):

	A	B	C
Compiler 1	5	1	1
Compiler 2	10	1	1

SW Performance (3)

- Clockzyklen
  - Compiler1: $(5 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 10 \times 10^9$
  - Compiler2: $(10 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 15 \times 10^9$

- Ausführungszeiten
  - Compiler1: $(10 \times 10^9) / (500 \times 10^6) = 20$ sec
  - Compiler2: $(15 \times 10^9) / (500 \times 10^6) = 30$ sec

- MIPS-Raten
  - Compiler1: $((5 + 1 + 1) \times 10^9) / (20 \times 10^6) = 350$ MIPS
  - Compiler2: $((10 + 1 + 1) \times 10^9) / (30 \times 10^6) = 400$ MIPS

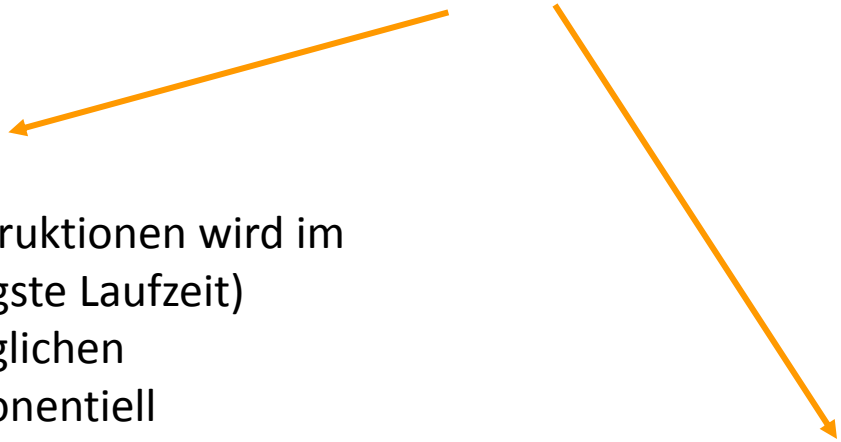
Compiler2 erzeugt höhere MIPS-Rate, aber Compiler1 schnelleren Code!

SW Performance (4)

- MFLOPS (million floating-point operations per second)
  - berücksichtigt Parallelоperationen
  - setzt optimale Belegung voraus
- MACS (million multiply & accumulates per second)
  - wichtig bei DSPs
- MOPS (million operations per second)
  - alle Operationen zusammengezählt:
    ALUs, Adressrechnungen, DMA
  - setzt optimale Belegung voraus

Worst-case execution time (WCET)

- Kann nicht durch Profiling bestimmt werden
- Schätzung mittels Programmanalysetechniken



Programmpfadanalyse

- Welche Reihenfolge von Instruktionen wird im Worst-case ausgeführt? (längste Laufzeit)
- Problem: die Anzahl der möglichen Programmpfade wächst exponentiell

Modellierung der Zielarchitektur

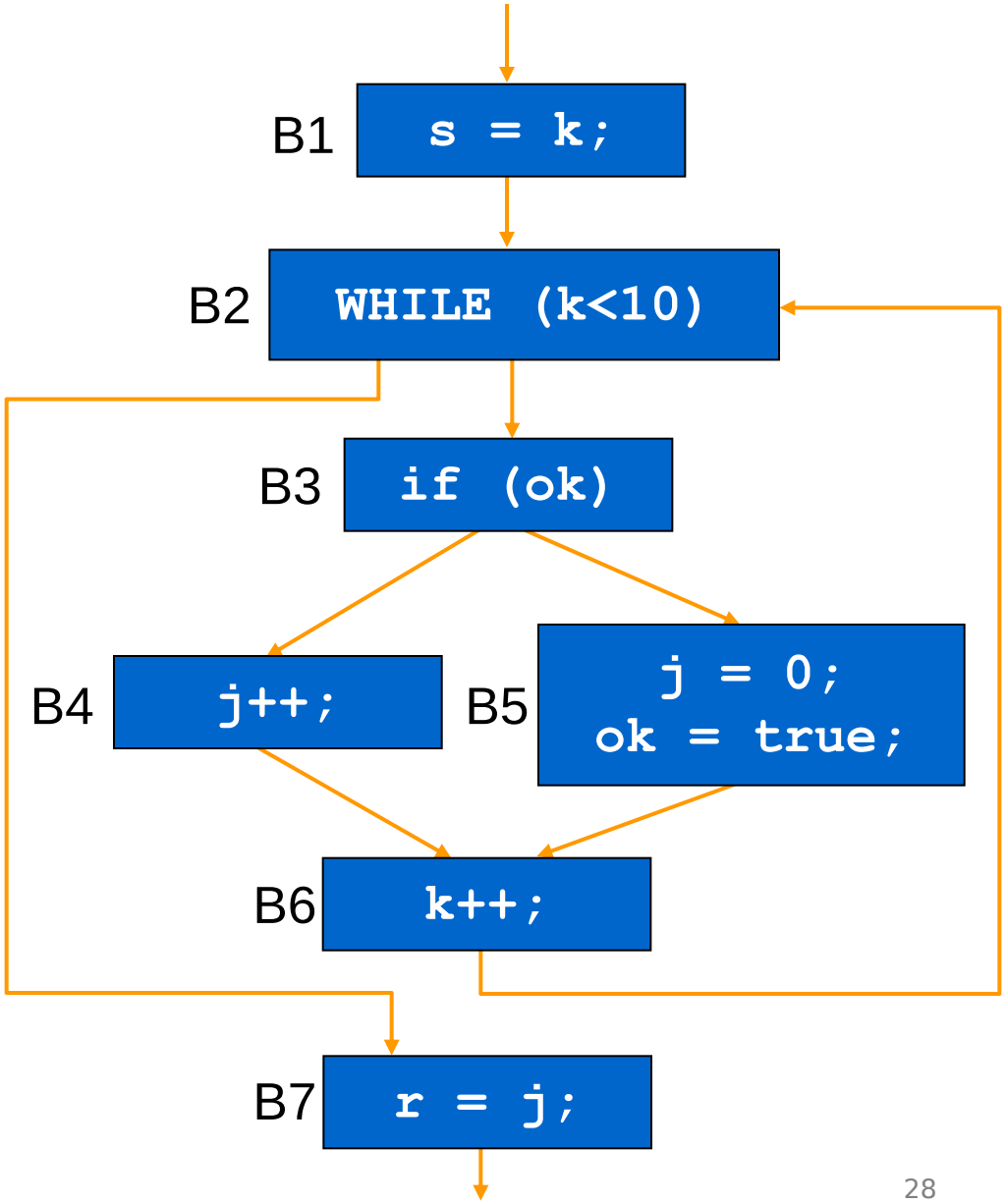
- Berechnung der geschätzten WCET für ein spezifisches Prozessormodell
- Probleme: Compileroptimierungen, dynamische Effekte durch Pipelining, Caches

Programmpfadanalyse

- Die geschätzte WCET
  - ist immer größer als die tatsächliche WCET, eine gute Schätzung approximiert die tatsächliche WCET aber möglichst nahe
- Prozessormodell
  - ein Prozessor (eine skalare Einheit)
  - keine Interrupts
  - kein Betriebssystem
- Programmiermodell
  - keine rekursiven Funktionsaufrufe (direkt und indirekt)
  - keine Pointeroperationen
  - Schleifen müssen beschränkt sein

Beispiel

```
/* k >= 0 */
s = k;
WHILE (k < 10) {
    IF (ok)
        j++;
    ELSE {
       j = 0;
       ok = true;
    }
    k ++;
}
r = j;
```

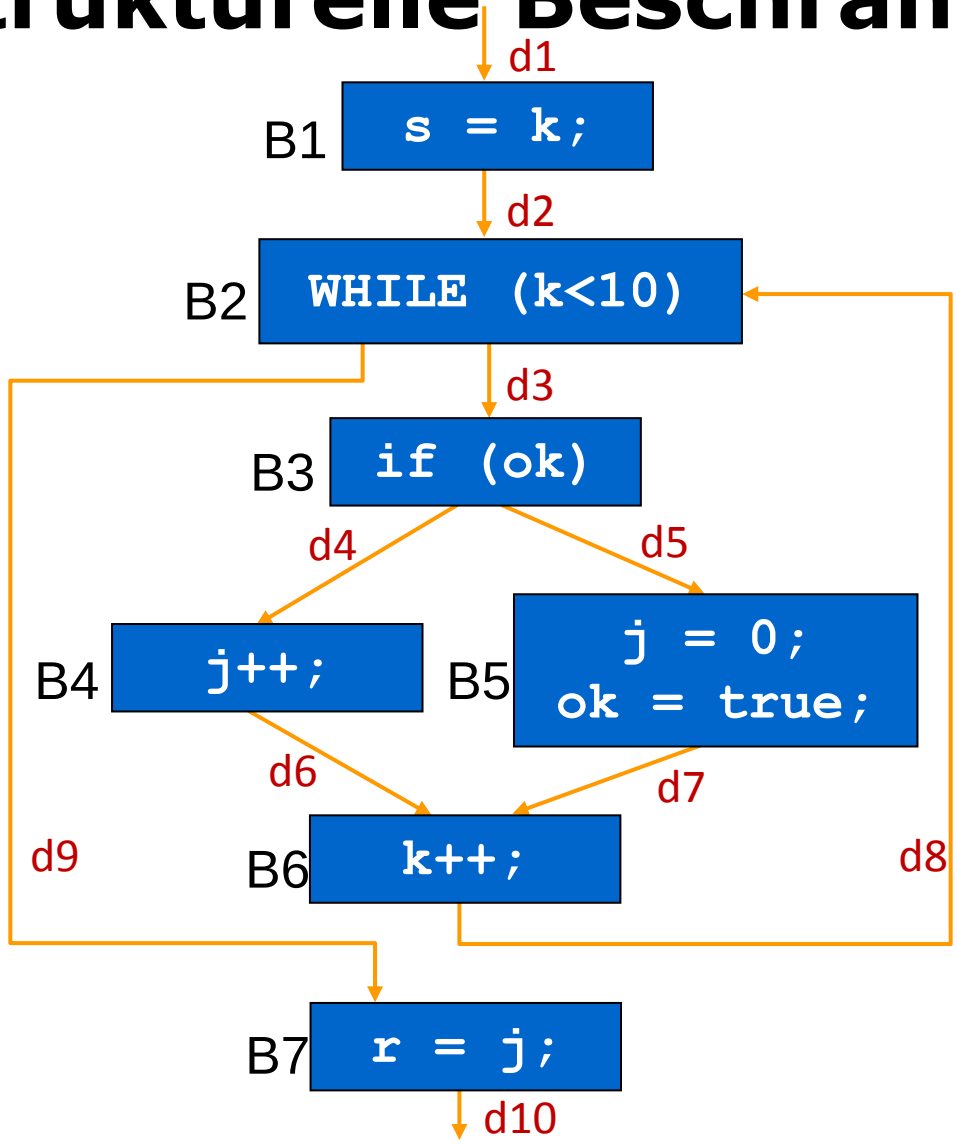


Berechnung der WCET

- Definition: Ein Programm besteht aus N Grundblöcken, wobei jeder Grundblock B_i eine Ausführungszeit c_i hat und maximal x_i mal ausgeführt wird. Dann ist die

$$WCET = \sum_{i=1}^{N} c_i \cdot x_i$$

Strukturelle Beschränkungen



Flussgleichungen:

$d1 = d2 = x_1$

$d2 + d8 = d3 + d9 = x_2$

$d3 = d4 + d5 = x_3$

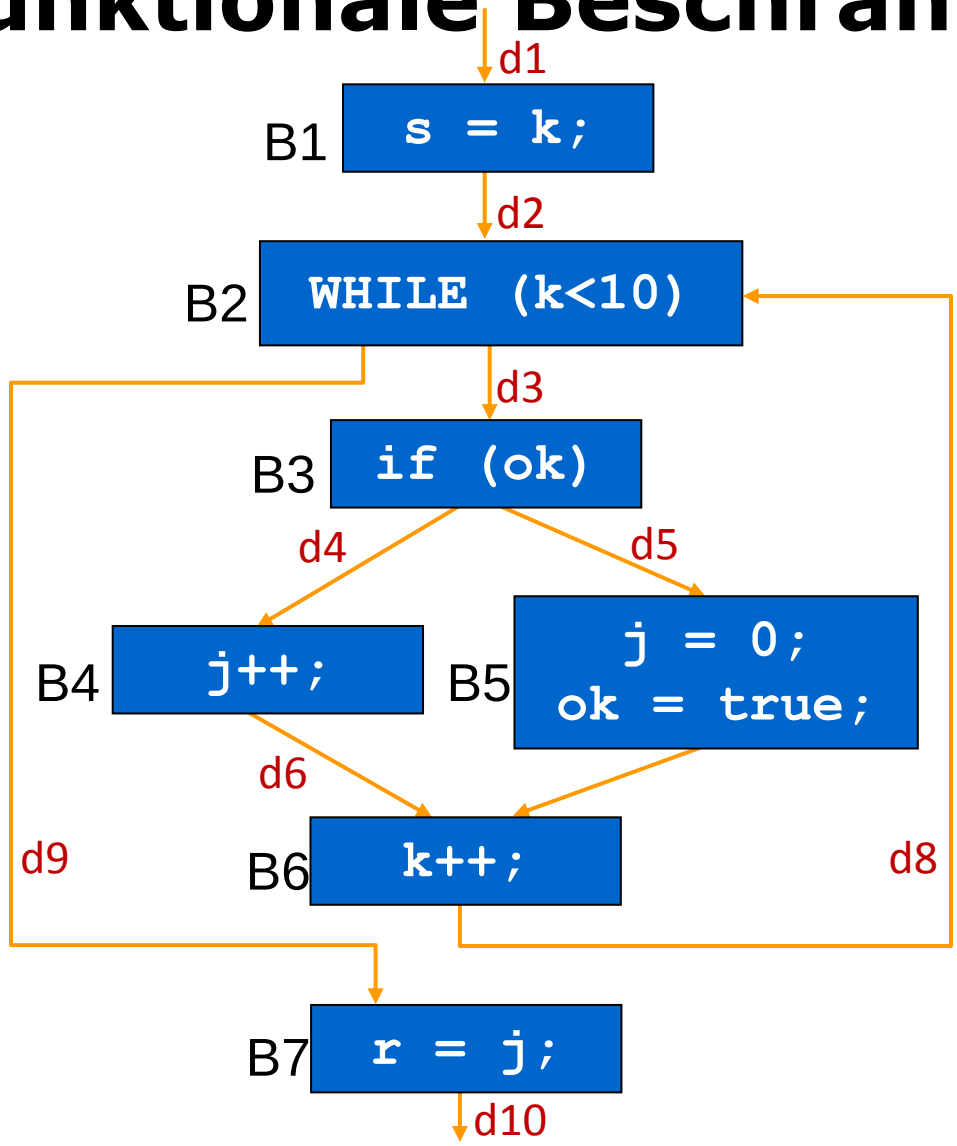
$d4 = d6 = x_4$

$d5 = d7 = x_5$

$d6 + d7 = d8 = x_6$

$d9 = d10 = x_7$

Funktionale Beschränkungen (1)



Schleife wird maximal 10 mal durchlaufen:

$$x_3 <= 10 \cdot x_1$$

B5 wird maximal einmal durchlaufen:

$$x_5 <= 1$$

Funktionale Beschränkungen (2)

- Werden durch den Programmierer definiert
  - durch Schranken für Schleifenzähler
  - durch Kenntnis des Programmkontextes

  oder durch formale Analyse abgeleitet

- Können komplex sein
  - Bsp.: “wird der ELSE-Zweig in der Schleife ausgeführt, wird die Schleife genau 5 mal durchlaufen”

$$(x_5 = 0) \;||\; (x_5 >= 1) \;\&\; (x_3 = 5 \cdot x_1)$$

WCET - ILP

- ILP mit strukturellen und funktionalen Beschränkungen

$$WCET = \max\left\{\sum_{i=1}^{N} c_i \cdot x_i \;\middle|\; d_1 = 1 \wedge \right.$$

$$\sum_{j \in in(B_i)} d_j = \sum_{k \in out(B_i)} d_k = x_i, i = 1...N \wedge$$

$$\left. \text{functional constraints} \right\}$$