technische universität
dortmund

fakultät für informatik
informatik 12

Synthese Eingebetteter Systeme

Sommersemester 2011

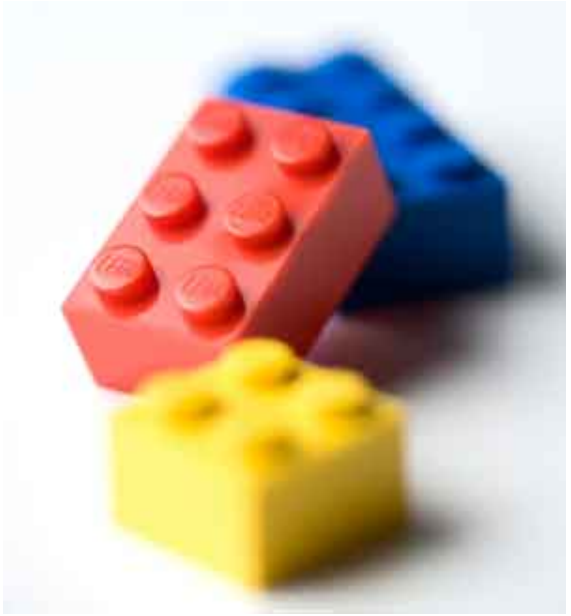
18 – Abbildung von Anwendungen: Daedalus und Simulated Annealing

Michael Engel
Informatik 12
TU Dortmund

2011/07/03

Daedalus und weitere Systeme

- Daedalus
- Simulated Annealing
- Beispiele



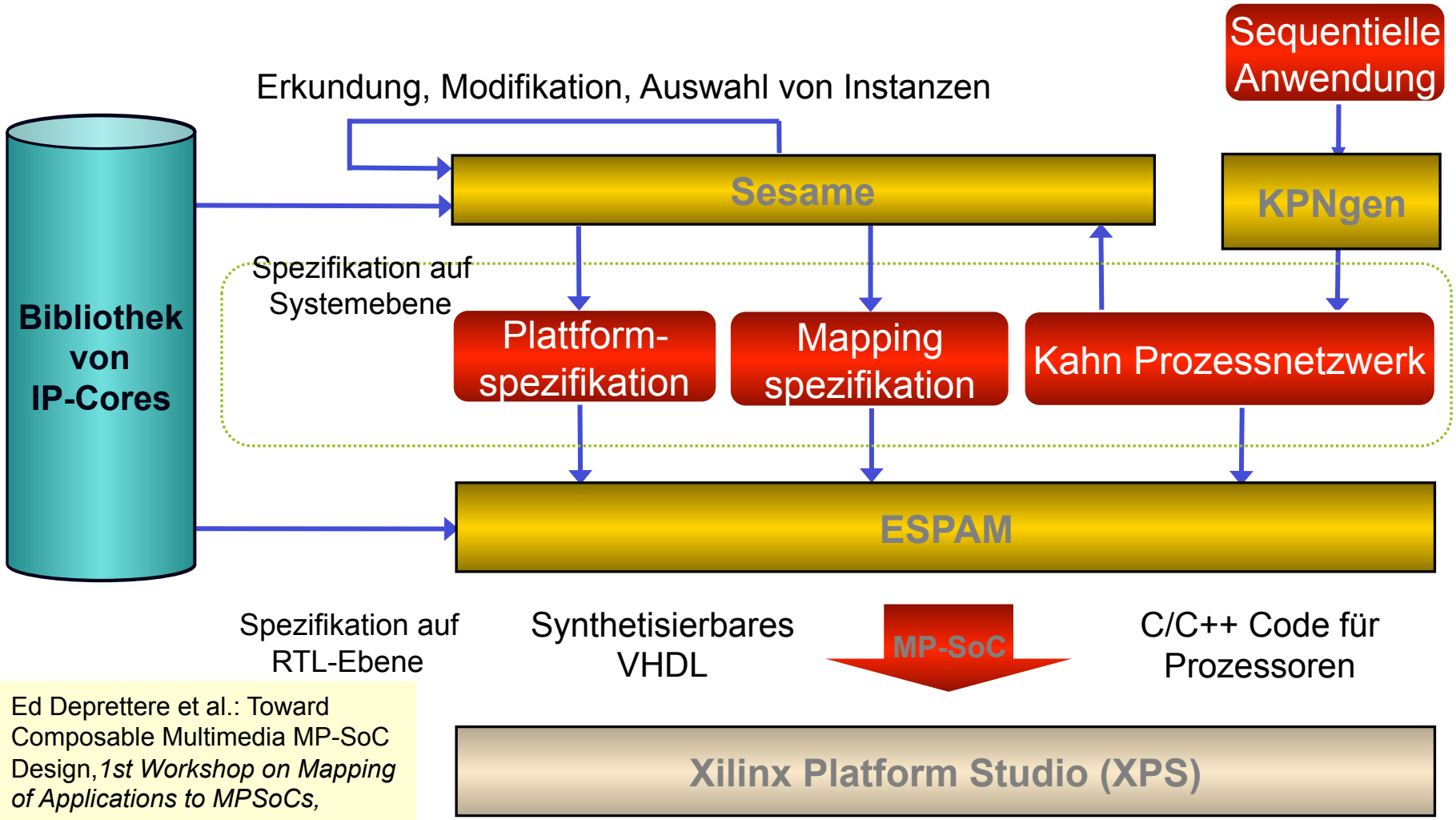
Überblick: Eine einfache Klassifikation

Architektur vorgegeben/ Automatische Parallelisierung	Vorgegebene Architektur	Architektur soll entworfen werden
Von vorgegebenem Modell ausgehend	Abbildung auf CELL, HOPES, ETHAM	COOL codesign tool; EXPO/SPEA2
Automatische Parallelisierung	Franke, O'Boyle et al., Mnemee MAPS	Daedalus

Daedalus

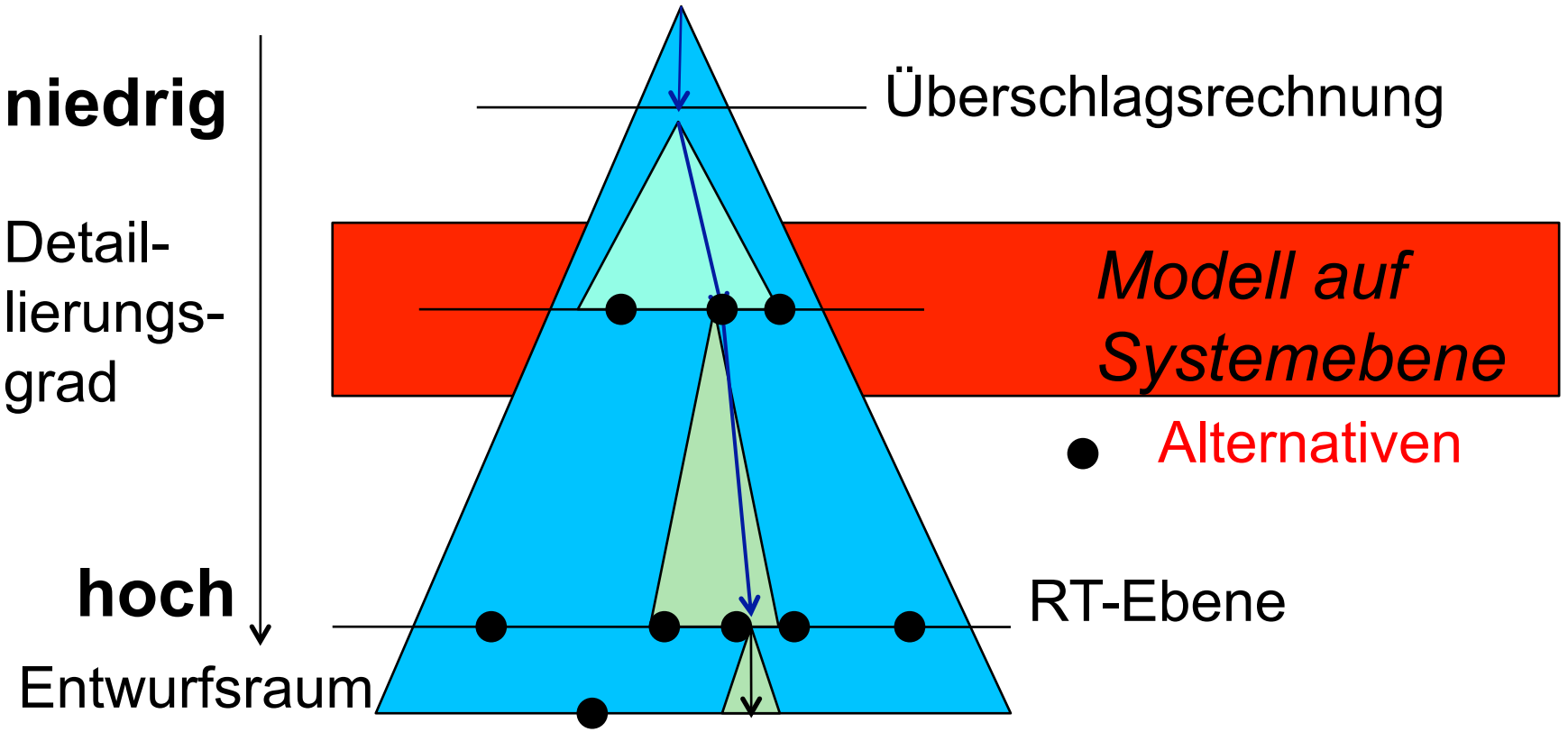
- Daedalus ist ein Framework für den Entwurf eingebetteter MPSoCs
- Komponenten:
  - *KPNgen:* Automatische Parallelisierung von streaming media-Anwendungen
  - *Sesame:* Modellierung auf Systemebene und Simulation für Entwurfsraumerkundung
  - *ESPAM:* Synthese auf Systemebene, “plug-and-play”

Entwurfsfluss in Daedalus



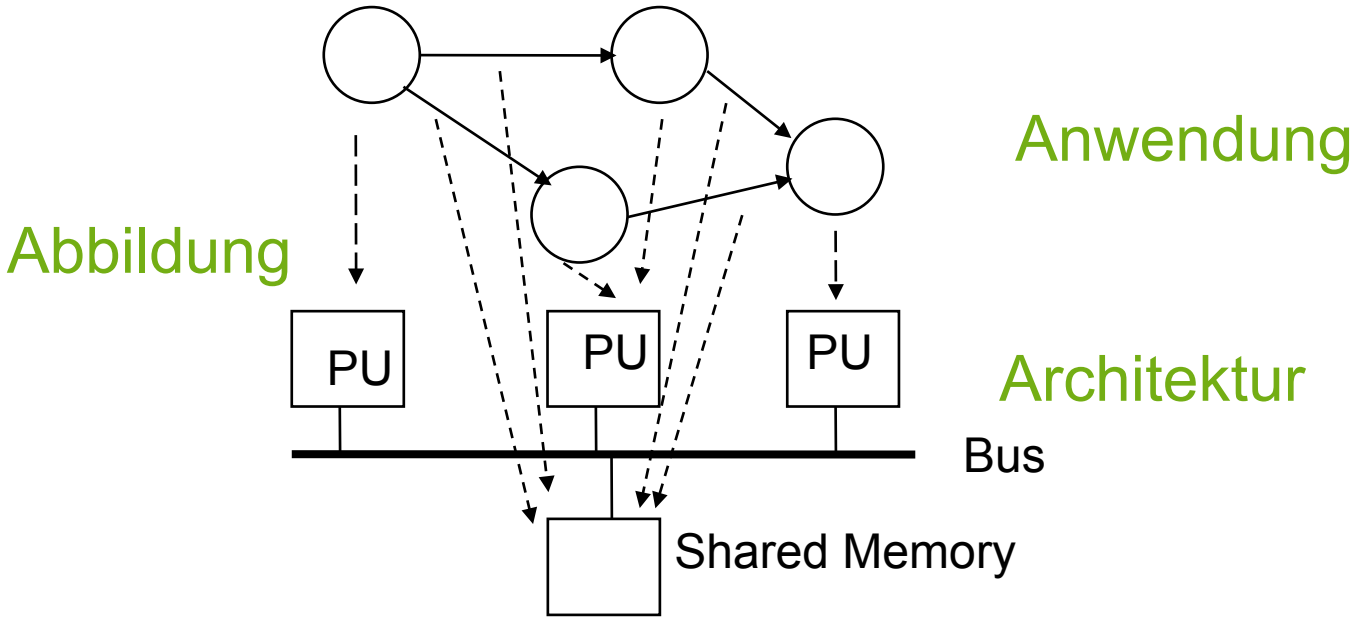
- Ed Deprettere et al.: Toward Composable Multimedia MP-SoC Design, *1st Workshop on Mapping of Applications to MPSoCs, Rheinfels Castle*, 2008

Entwurfsraum auf Systemebene



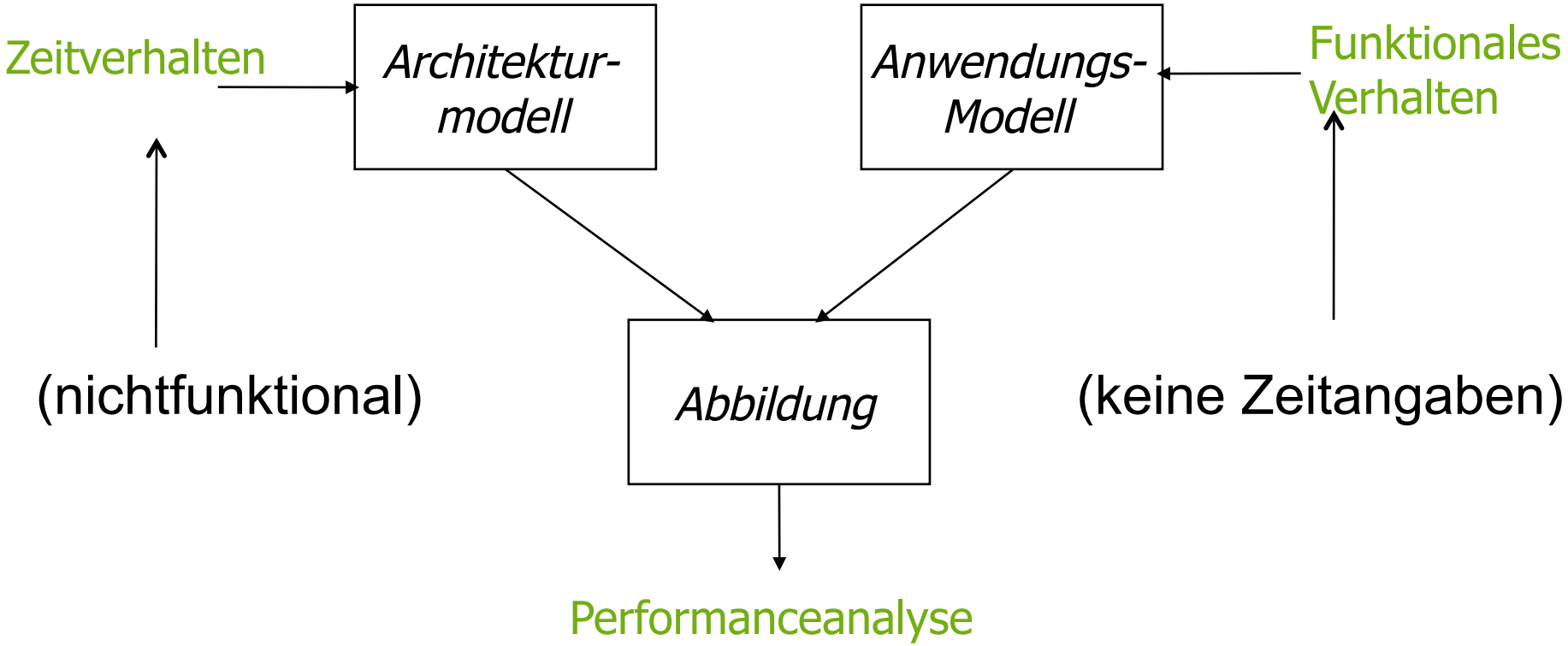
Systemebene

- Eingebettetes System modelliert als Tripel:
- {Anwendungsmodell, Architekturmodell, Abbildungsspezifikation}



Modelle auf Systemebene

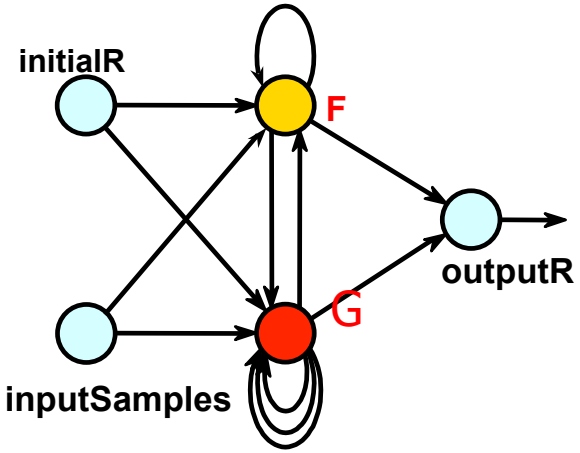
Y-Chart: Trennung der Belange



Modelle auf Systemebene

Anwendungsmodelle in Daedalus:

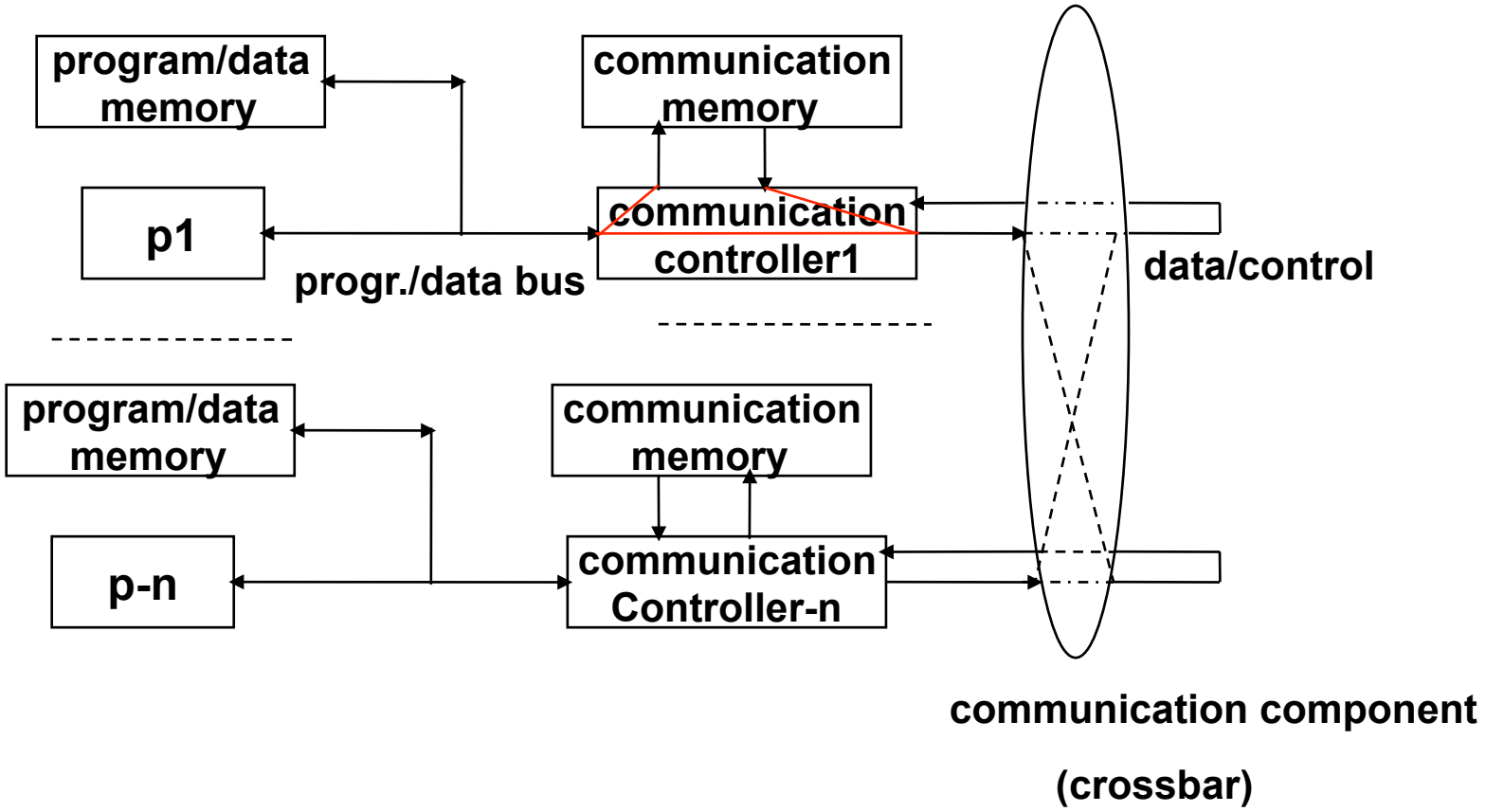
- Prozessnetzwerke



Architekturmodelle in Daedalus:

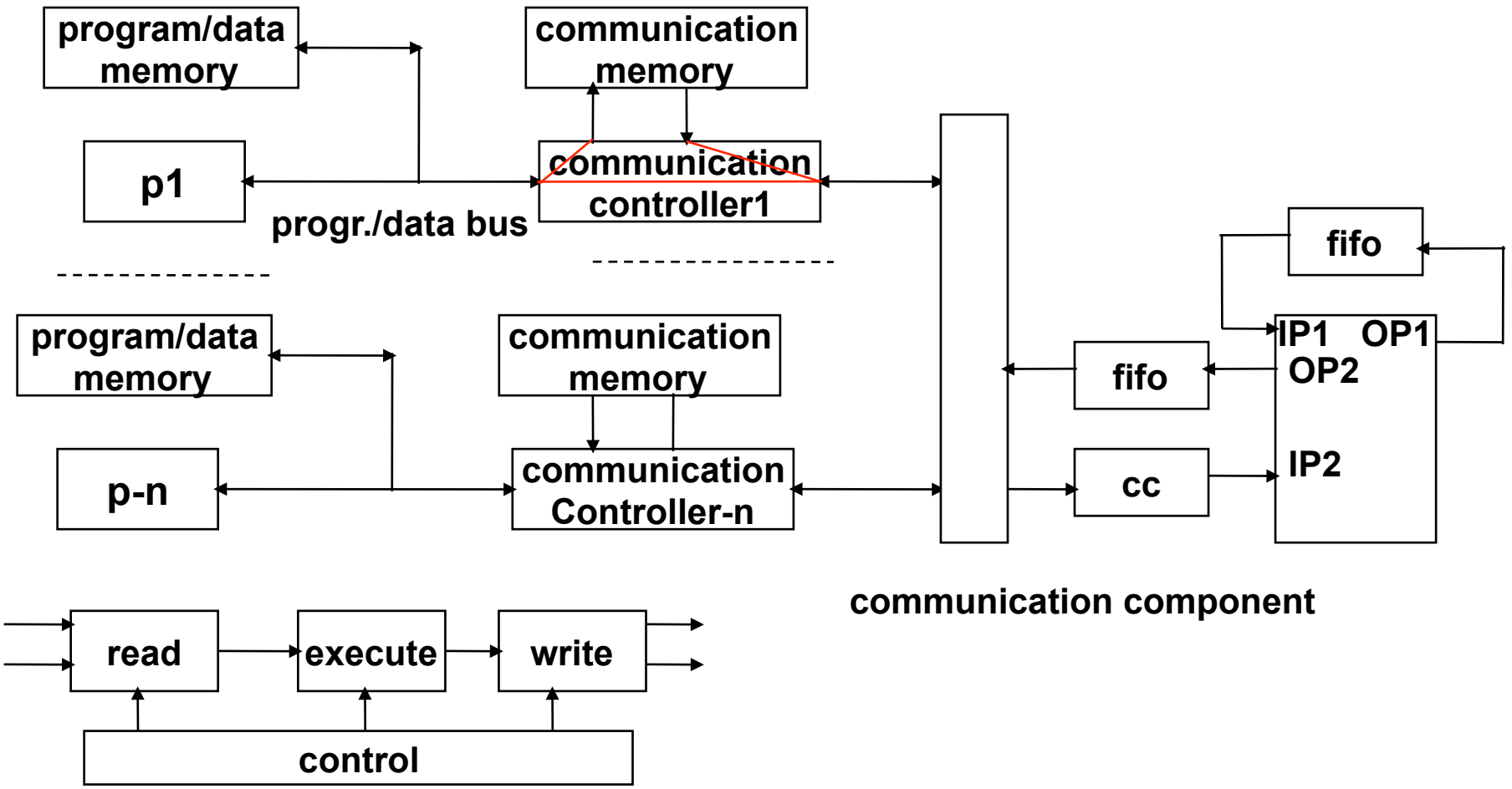
- MPSoCs, die *ausreichend ähnlich* zu den Prozessnetzwerk-Modellen der Anwendungen sind

Typische Architekturen (1)



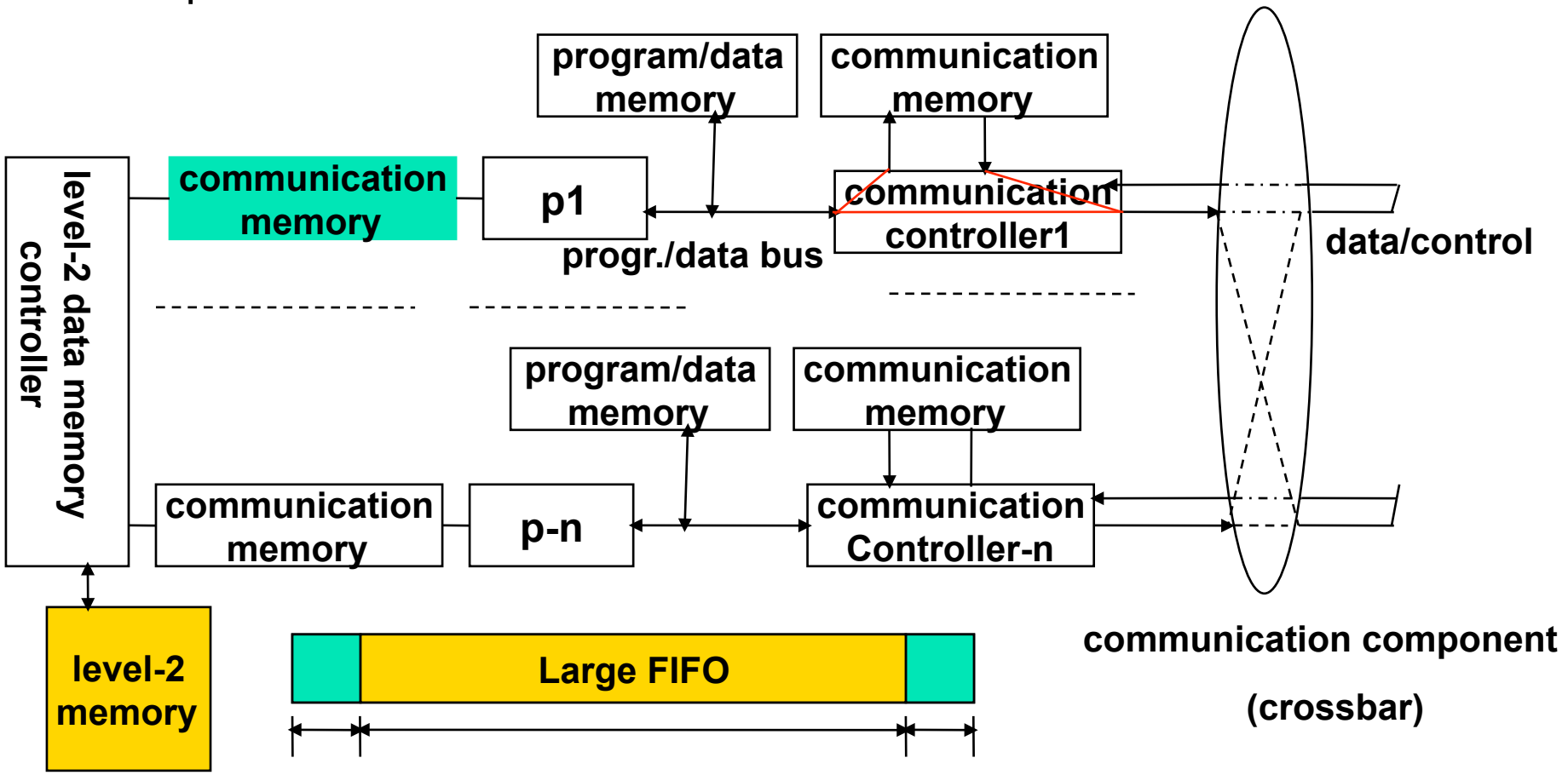
p-x: Mikroprozessor oder spezialisiertes Lese-/Schreibmodul

Typische Architekturen (2)



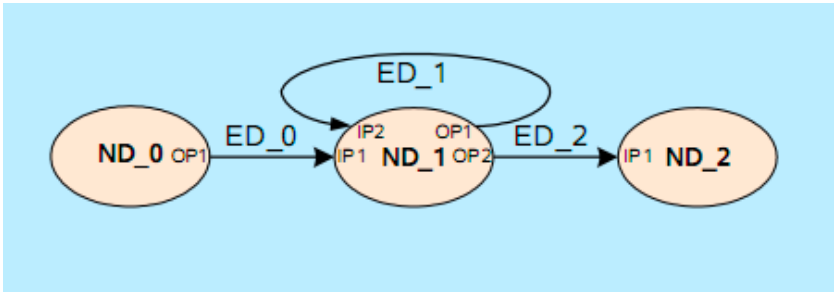
Typische Architekturen (3)

Speicherhierarchien sind realisierbar

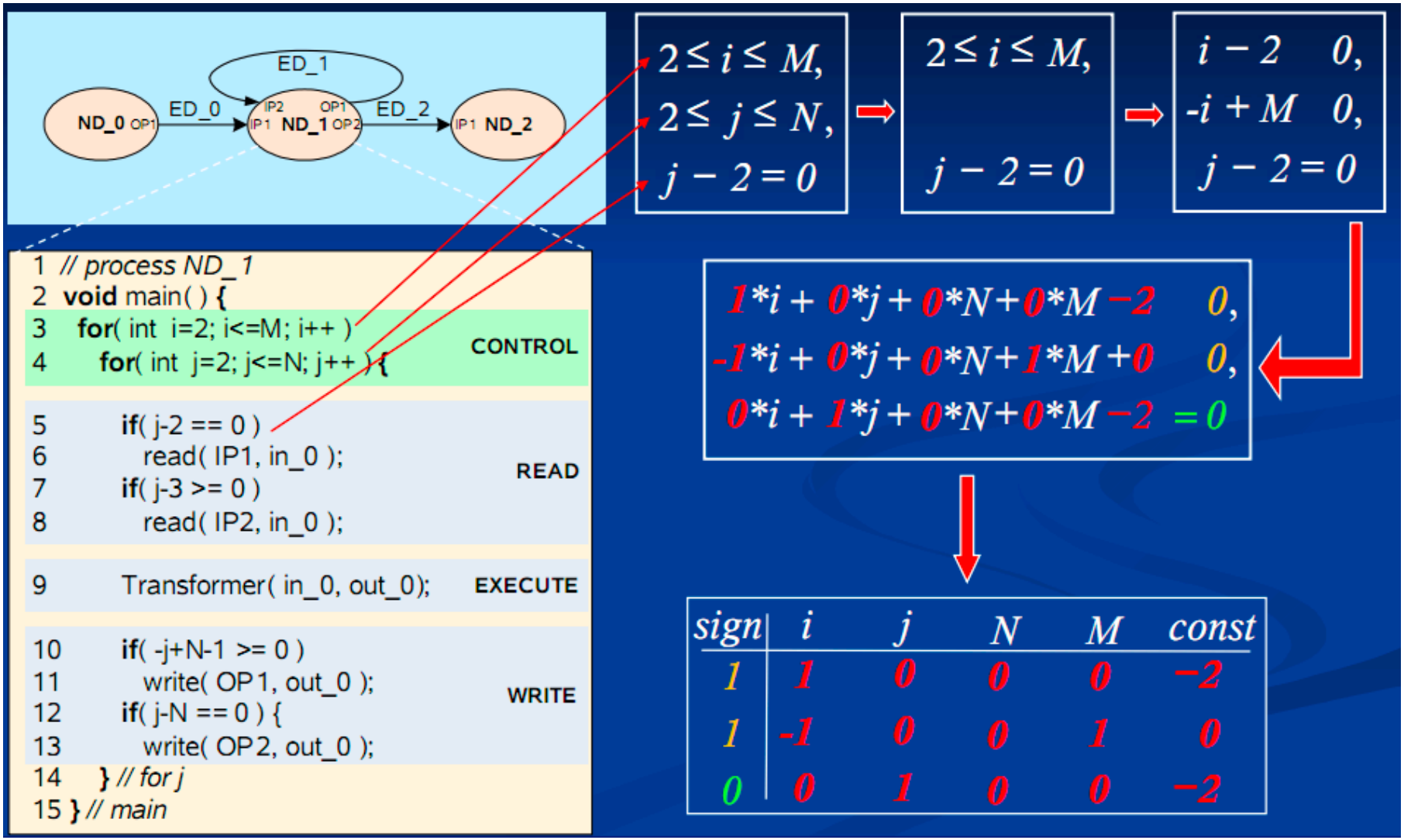


Daedalus-Netzwerktopologie: XML

```
<sadg>

  <adg name="simple" levelUpNode="">
    <parameter name="N" lb="450" ub="1000" value="450"/>
    <parameter name="M" lb="275" ub="1000" value="275"/>

    <node name="ND_0" levelUpNode="">
      <outport name="OP1" node="ND_0" edge="ED_0">
      </outport>
    </node>

    <node name="ND_1" levelUpNode="">
      <inport name="IP1" node="ND_1" edge="ED_0">
      </inport>
      <inport name="IP2" node="ND_1" edge="ED_1">
      </inport>
      <outport name="OP1" node="ND_1" edge="ED_1">
      </outport>
      <outport name="OP2" node="ND_1" edge="ED_2">
      </outport>
    </node>

    <node name="ND_2" levelUpNode="">
      <inport name="IP1" node="ND_2" edge="ED_2">
      </inport>
    </node>
    <edge name="ED_0" fromPort="OP1" fromNode="ND_0" toPort="IP1" toNode="ND_1" size="1">
    </edge>

    <edge name="ED_1" fromPort="OP1" fromNode="ND_1" toPort="IP2" toNode="ND_1" size="1">
    </edge>

    <edge name="ED_2" fromPort="OP2" fromNode="ND_1" toPort="IP1" toNode="ND_2" size="1">
    </edge>

  </adg>

</sadg>
```



Daedalus-Prozesskontroll-Code

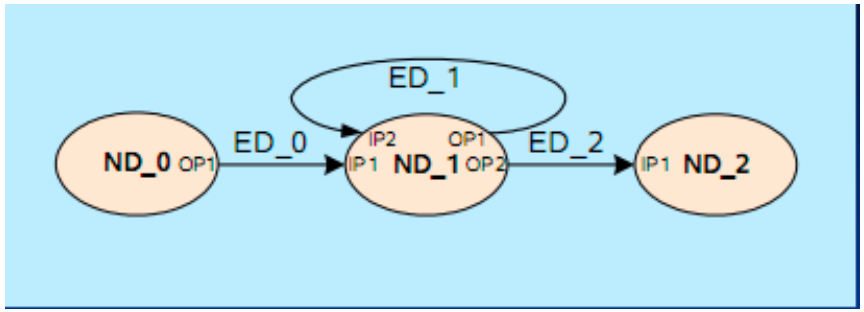


Daedalus-Prozesskontroll-Code (2)

```
<sadg>

  <adg name="simple" levelUpNode="">
    <parameter name="N" lb="450" ub="1000" value="450"/>
    <parameter name="M" lb="275" ub="1000" value="275"/>

    <node name="ND_1" levelUpNode="">

      <inport name="IP1" node="ND_1" edge="ED_0">
        <bindvariable name="in_0" dataType="int"/>
        <domain type="LBS">
          <linearbound index="i, j" staticControl="" dynamicControl="" parameter="N, M">
            <constraint matrix="[0,  0, 1, 0, 0, -2;
                                 1,  1, 0, 0, 0, -2;
                                 1, -1, 0, 0, 1,  0]"/>
          </linearbound>
        </domain>
      </inport>

      <inport name="IP2" node="ND_1" edge="ED_1">
      </inport>

      <outport name="OP2" node="ND_1" edge="ED_1">
      </outport>
      <outport name="OP2" node="ND_1" edge="ED_2">
      </outport>

      <function name="Transformer">
        <inargument name="in_0" dataType="int"/>
        <outargument name="out_0" dataType="int"/>
      </function>
      <domain type="LBS">
        <linearbound index="i, j" staticControl="" dynamicControl="" parameter="N, M">
          <constraint matrix="[1,  1,  0, 0, 0, -2;
                               1, -1,  0, 0, 1,  0;
                               1,  0,  1, 0, 0, -2;
                               1,  0, -1, 1, 0,  0]"/>
        </linearbound>
      </domain>
    </node>
  </adg>

</sadg>
```

sign	i	j	N	M	const
1	1	0	0	0	−2
1	-1	0	0	1	0
0	0	1	0	0	−2

Schedule-Spezifikation



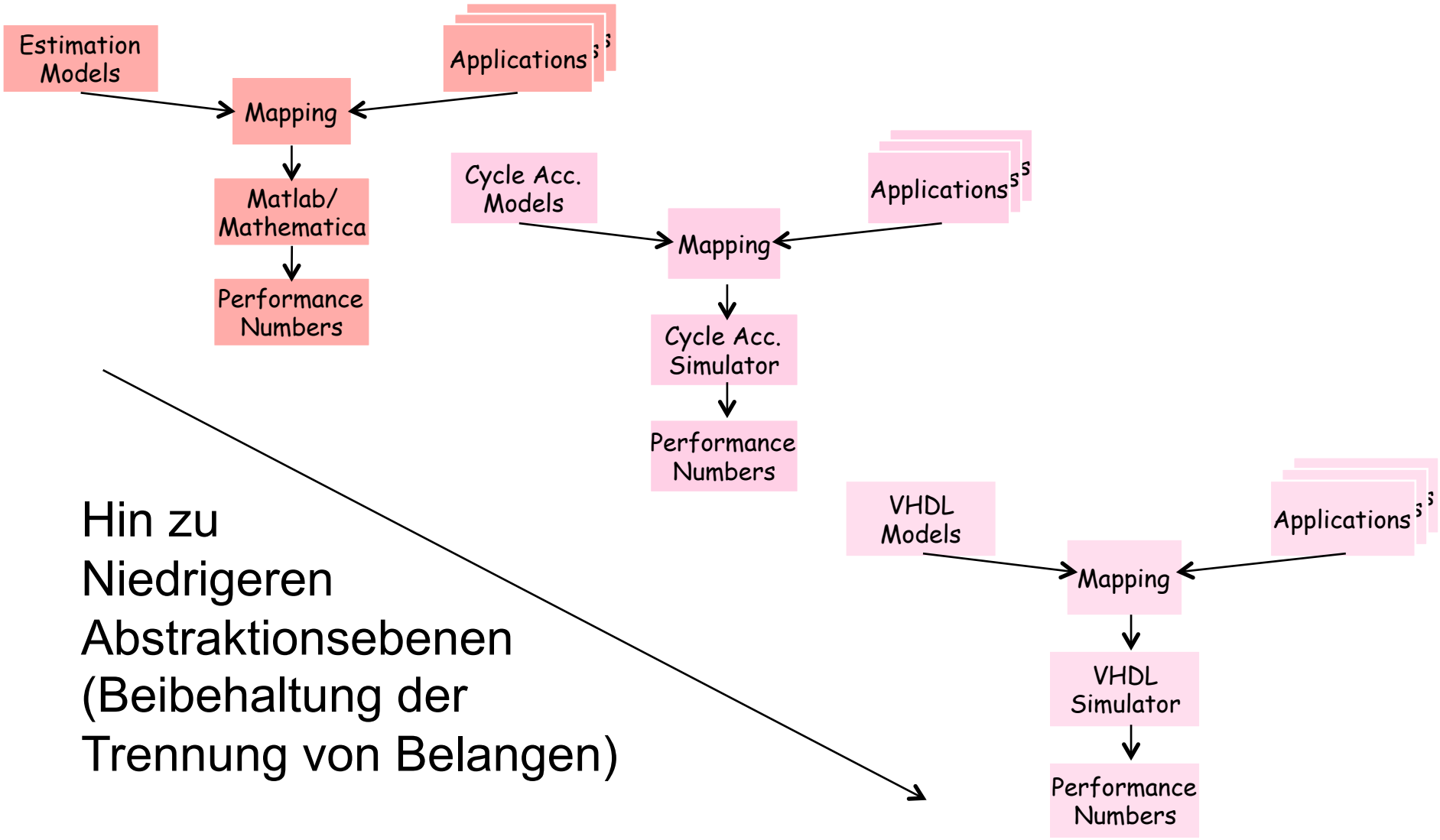

```
<?xml version="1.0"?>
<sadg>

  <adg name="simple" levelUpNode="">
      ...
  </adg>

  <ast>

    <for iterator="c0" LB="2" UB="1*M+1" stride="1">
      <for iterator="c1" LB="2" UB="1*N+1" stride="1">

        <stmt node="ND_0"/>

        <if LHS="1*c0" RHS="1*M" sign="-1">

            <if LHS="1*c1" RHS="1*N" sign="-1">
               <stmt node="ND_1"/>
            </if>

            <if LHS="1*c1" RHS="1*N" sign="0">
               <stmt node="ND_2"/>
            </if>

        </if>

      </for>
    </for>

  </ast>

</sadg>
```

- Schedule garantiert deadlockfreie Ausführungsreihenfolge
  - Entweder mit absolut maximalen FIFO-Größen, die deadlockfreie Ausführung garantieren
  - Oder mit der minimalen FIFO-Größe, die maximale Performance garantiert
- Schedule als abstrakter Syntax-baum in XML – <ast>-Tag
- Der <ast> kann in ein *control program* umgewandelt werden, das den Schedule implementiert

Übereinstimmung von Modellen

- Wesentlich für die Effizienz des Systems ist die Übereinstimmung von Anwendung und Architektur
- Das Prozessnetzwerkmodell kann eine ungünstige Abbildung für davon abweichende Architekturen aufweisen (z.B. den CELL)

Hierarchische Y-Charts



Performance-Evaluierung

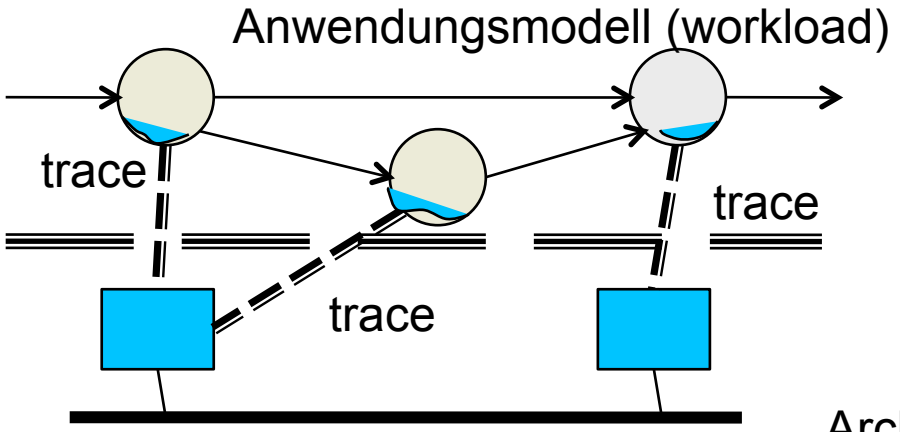
Ziel: Erzeugung von Performancewerten

- Analytisch durchführbar (Abschätzung)
  - Kann bei Einschränkung des Entwurfsraums helfen
- Meist *simulationsbasiert*

Evaluierung durch Cosimulation

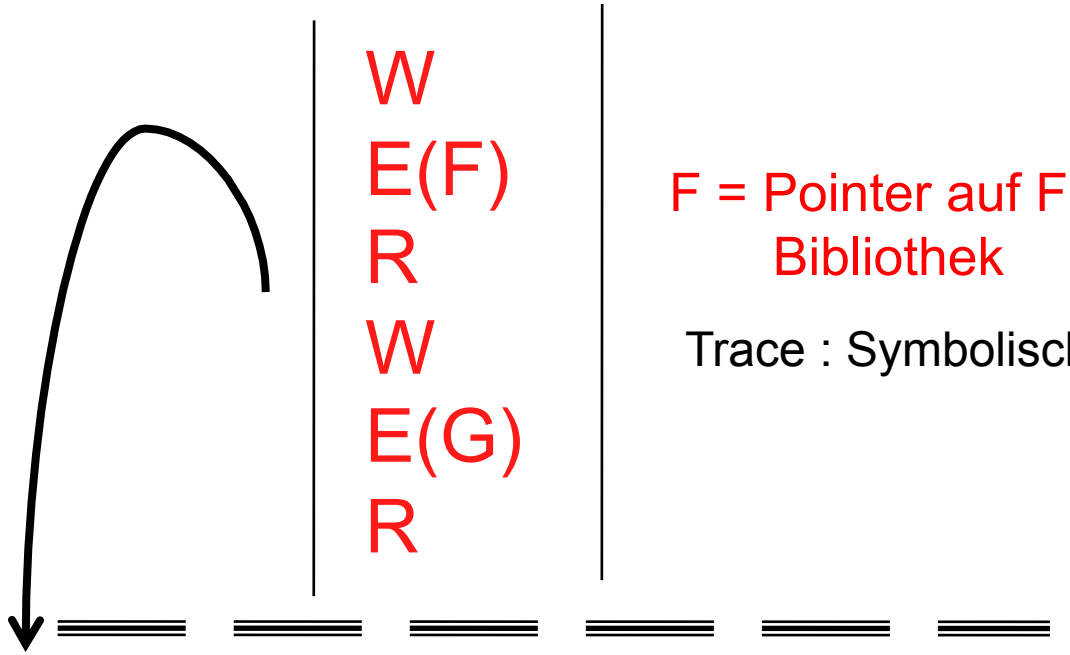
Grundprinzip:

- Anwendungs- (oder funktionaler) Simulator **steuert** Architektur- (oder Plattform-)Simulator
- Modelle interagieren über **Traces** (Ablaufprotokolle) **von Aktionen**
- Traces können **on-line** oder **off-line** erzeugt werden



Trace: lineare Abfolge abstrakter Instruktionen:
Read, Execute, Write

Trace-getriebene Simulation



F = Pointer auf Funktion oder Code in Bibliothek

Trace : Symbolische Instruktionen

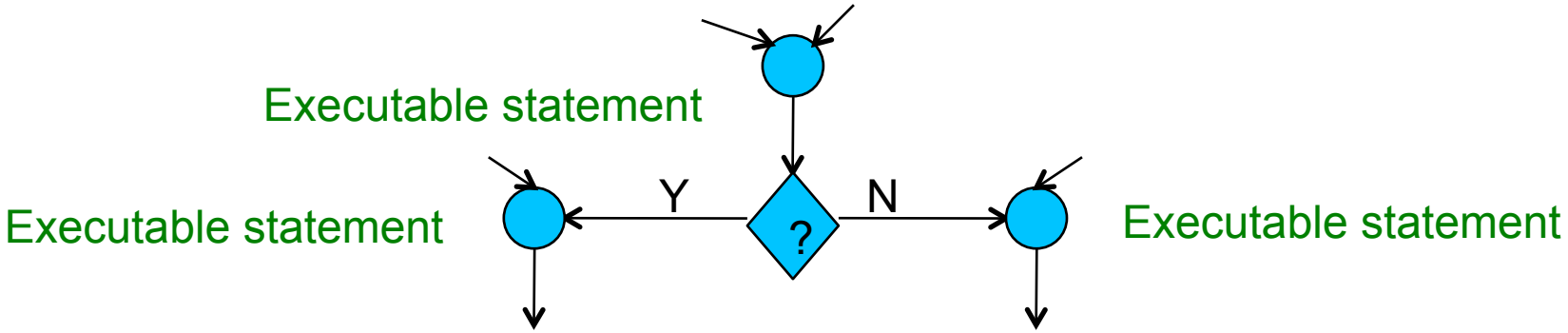
$t_w = f_w$(check for room, store data, signal data)

$t_e = f_e$(execute(G)) *G ist Pointer auf Prozessor in Bibliothek*

$t_r = f_r$(check for data, load data, signal room)

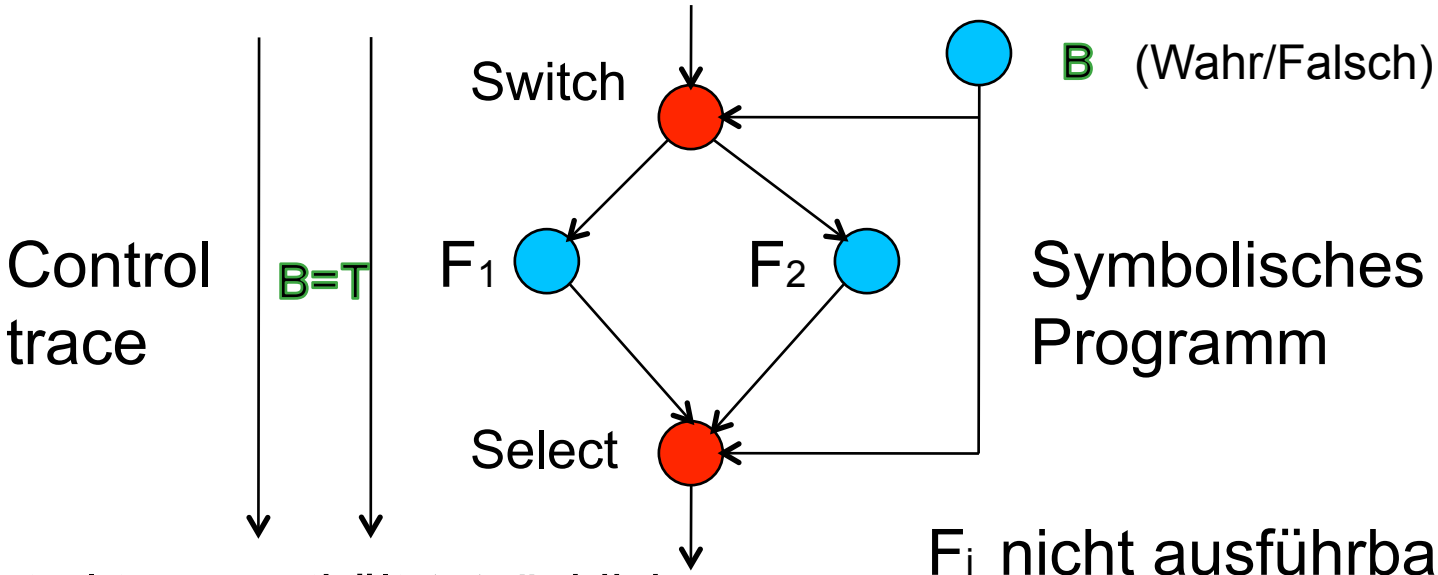
Symbolische Programme

- Trace-getriebene Co-Simulation ist eingeschränkt, da Traces nur *Instruktionen* enthalten, die den Datenfluss, nicht aber den Kontrollfluss modellieren
- Kontrollflüsse können nicht modelliert werden
- Klassische Modellierung auf niedriger Ebene (high-level Synthese) basiert auf CDFG.

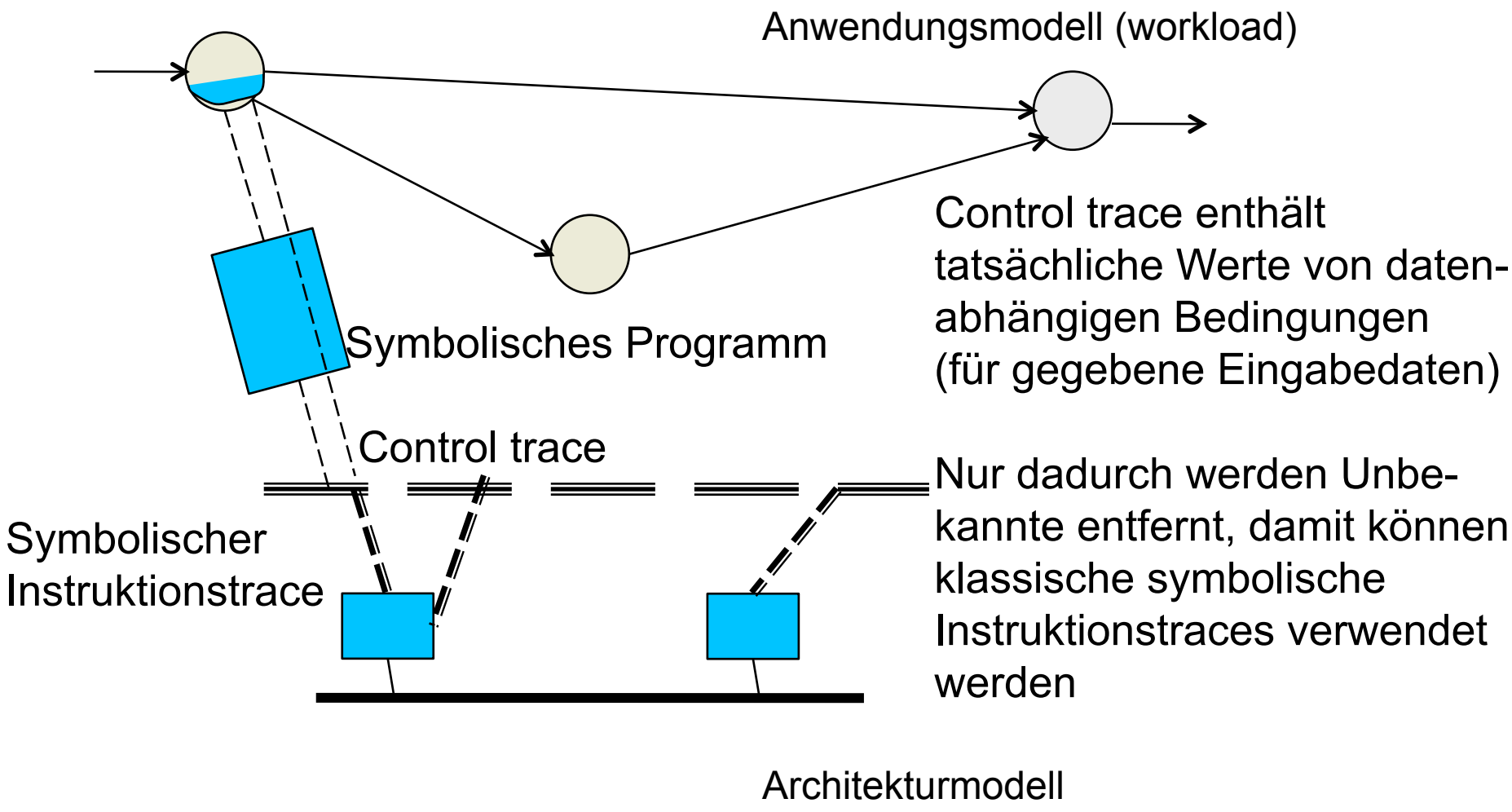


Symbolisches Programm

- Ähnlich zu CDFG, aber abstrakter
  - Knoten sind kompakter dargestellt und nicht ausführbar.



Symbolische Programme



Control trace enthält tatsächliche Werte von daten-abhängigen Bedingungen (für gegebene Eingabedaten)

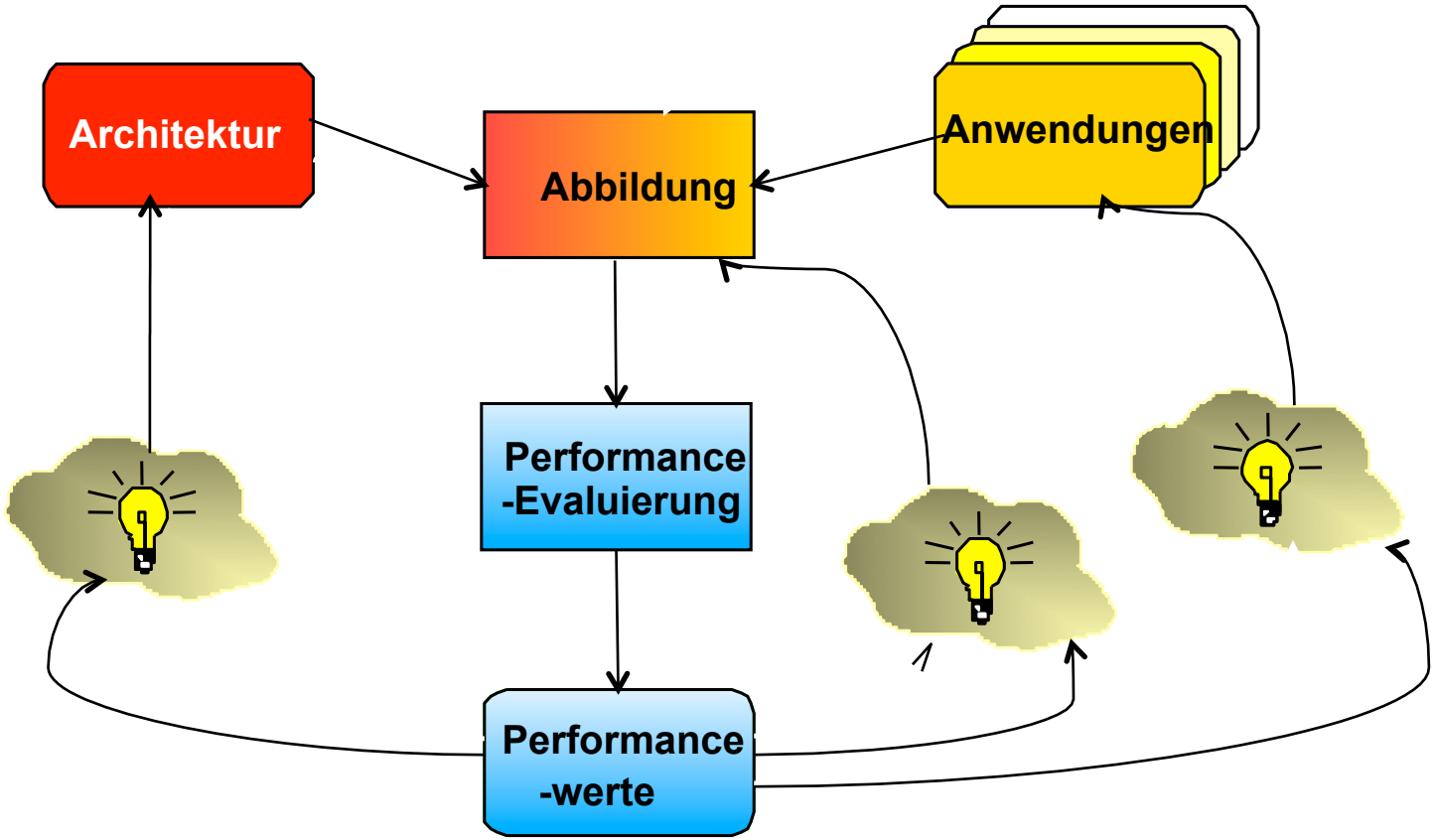
Nur dadurch werden Unbekannte entfernt, damit können klassische symbolische Instruktionstraces verwendet werden

Entwurfsraumerkundung

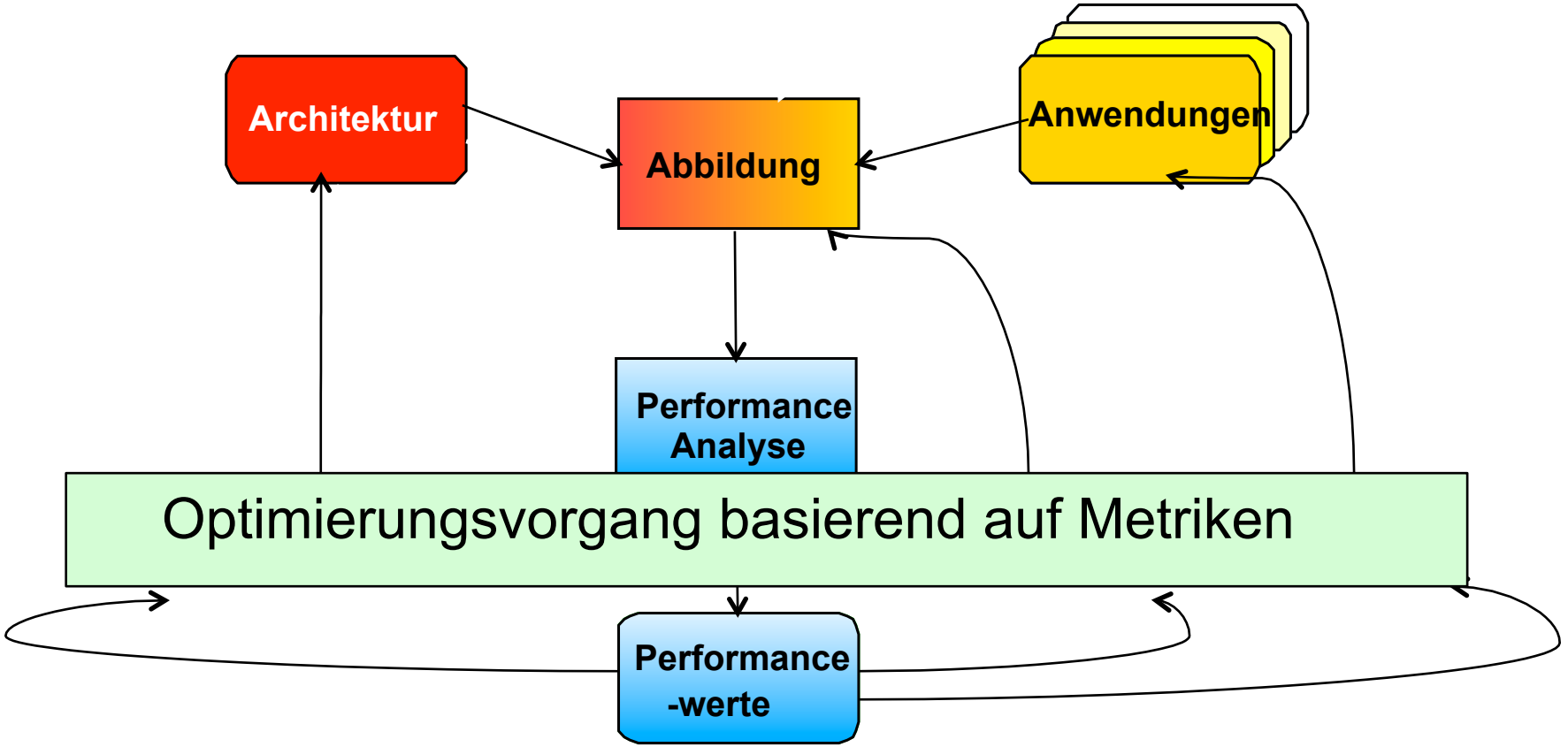
Wie behandelt man Performancewerte?

- Suche nach einer oder mehreren Architektur(en), die für die Anwendung “am Besten” geeignet sind
- Erzielung eines guten *trade-off* für gegenläufige Ziele.

Feedback und Verbesserung



Entwurfsraumerkundung



Ziele

Finde ein oder mehrere *Anwendungs-Architektur-*Modellpaare, die eine Menge an Anforderungen oder Beschränkungen erfüllen.

Beispiele:

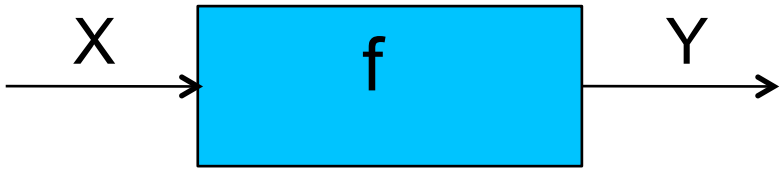
- Bestimmte Prozesse müssen auf bestimmten Prozessoren oder Prozessorarten laufen
- Teilmengen von Prozessen müssen auf den selben Prozessor abgebildet werden
- Bestimmte Kommunikation muss über vorgegebene Wege ablaufen
- Restriktionen bei gleichzeitiger Verwendung von Ressourcen
- Performance maximal, Kosten minimal

Optimierung

- Entwurfsraumerkundung ist ein *großes, schlecht gestelltes* Optimierungsproblem:
  - Eigenschaften des Optimierungsraumes sind nicht bekannt
- Deterministische Verfahren (z.B. branch and bound) sind stets NP-hart
  - Sie garantieren asymptotische Konvergenz
- Heuristische Zufallssuche-Methoden sind einfach zu implementieren und ‘schnell’
  - Sie stellen eine Konvergenz der Wahrscheinlichkeit sicher

Optimierungsverfahren

- Die meisten praktisch verwendeten Methoden sind Zufallssuchen-(Meta-)Heuristiken,
  - leicht mit der ‘black box’-Auswertung implementierbar
  - nicht *greedy* (Vermeidung lokaler Optima).



(f muss keine Funktion im mathematischen Sinn sein)

- Iterative Verbesserung in Hinblick auf Qualitätsmaße
- Wenige oder keine Annahmen über das zu optimierende Problem
  - Die Performance einer Methode hängt von der Art des Problems ab.

Optimierungsverfahren

- Basieren auf klassischen Methoden für die Optimierung von einzelnen Zielen
- Unterschiedliche Ziele werden gewichtet
  - Eine Lösung pro Iteration
- Probabilistische Suchverfahren
  - *simulated annealing* (simulierte Abkühlung)
- Populationsbasierte Suchverfahren **=> DOL**
  - Evolutionäre Algorithmen
  - Genetische Algorithmen
    - Eine Population pro Iteration

Simulated Annealing

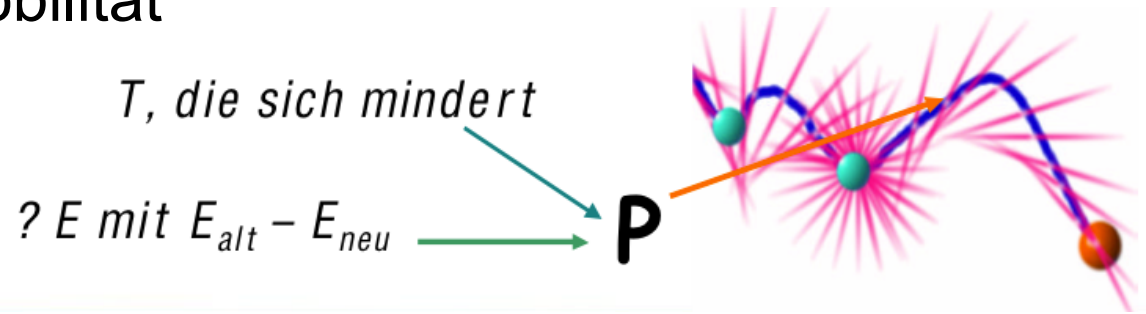
- Jeder Punkt s im Suchraum entspricht einem *Zustand* eines physikalischen Systems
- Die zu minimierende Funktion E(s) entspricht der *internen Energie* des Systems in diesem Zustand
- Das Optimierungsverfahren verschiebt den Systemzustand in einen Zustand *minimaler* Energie, ausgehend von einer initialen Speicherbelegung
- Übergang von Zustand s zu Zustand s' basiert auf einer Wahrscheinlichkeitsfunktion P(e, e', T)
  - e und e': Energien der beiden Zustände
  - T ist ein *Temperatur* genannter Parameter

Simulated Annealing

- Untersuchung von Vielkörpersystemen, Fluiden, Gasen, Festkörper auf Mittelwerte, Abweichungen, Temp., Druck [Metropolis 1953]
  - abkühlende Flüssigkeiten streben beim Abkühlen nach einer minimalen Energiebilanz E (stabile, regelmäßige Struktur)
- Nur möglich bei langsamer Kühlung
  - Ist Kühlung zu schnell, ordnen sich Teilchen unregelmäßig an

Simulated Annealing

- Schlechte Energiebilanz (Auftreten von Nebenminima):
  - Entdeckung eines in der Natur auftretenden Optimierungsprozesses
- **Vermutung**:
  - System akzeptiert Verschlechterung während Abkühlens, um in besseres Optimum zu gelangen
- Um lokalem Minimum zu entgehen, wird auch ein schlechterer Zustand des Systems erlaubt durch eine Wahrscheinlichkeit (Verteilung), abh. von:
- 1. Temperatur (Mobilität der Teilchen)
- 2. Zustandsverschlechterung



Simulated Annealing

- Akzeptanz (Wahrscheinlichkeit) der Teilchen, gewisse kinetische Energien beim Ordnen aufnehmen zu können, folgt der Verteilung:

  $P(E) \sim e^{(-\Delta E/kT)}$

- k = Konstante
- T = Temperatur
- ΔE = Differenz zweier Energiebilanzen

Metropolis-Algorithmus

- E1: alter Zustand
- E2: neuer Zustand
- T: Temperatur
- p: Wahrscheinlichkeit der Akzeptanz

$E_2 < E_1$	$E_2 > E_1$
Verbesserung der Lösung, wird sofort akzeptiert $p = 1$	Verschlechterung, nur akzeptiert mit $p = e^{(-(E2-E1)/T)}$ und $p >= random[0;1]$

Simulated Annealing

- Nutzbarmachung des Metropolis-Algorithmus für allgemein gültige theoretische und praktische Probleme der Optimierung [Kirkpatrick 1983]

- **Idee:**
  Optimierung durch simulierten Abkühlprozesses

  - Teilchenposition = Entwurfsvariablen des Problems
  - Energiefunktion E = Zielfunktion f

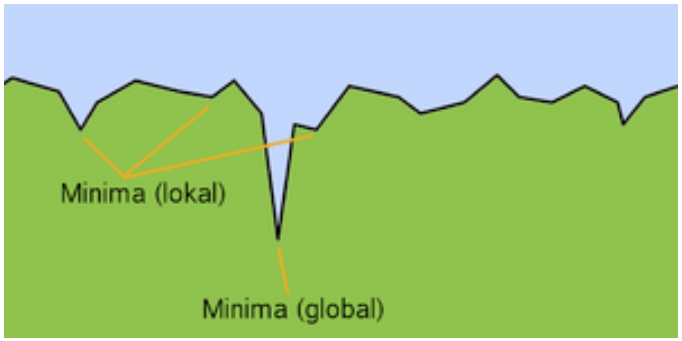
- Metaheuristisches Verfahren

Simulated Annealing

- Um Metropolis-Alg. für kombinatorische Probleme nutzbar zu machen, benötigt man:
  - Beschreibung aller möglichen Zustände, die das System einnehmen kann
  - Generator, der zufällige Änderung am System vornimmt
  - Eine qualitätsmessende Funktion *f*
  - Einen Kontrollparameter T

Thermodynamik	Optimierung
Systemzustände	zulässige Lösungen *i* aus S
Zustandsänderung	$i_h = j \in S_i$ (Nachbarschaft)
Temperatur	Kontrollparameter T_k

Simulated Annealing



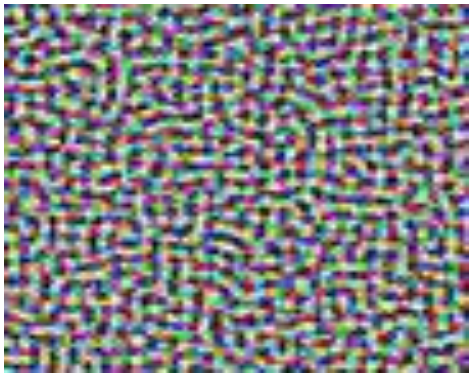
Eigenschaften von P(e, e', T):

- P muss für e' > e positiv sein, damit der Algorithmus lokale Optima wieder verlassen kann
  - Zustände, die schlechter als das globale Optimum, aber besser als die Nachbarwerte sind
- Wenn sich T Null annähert, muss P auch nach Null tendieren, wenn e' > e und positiv sein, wenn der Wert von e > e' sinkt
- Sobald T=0, werden die Algorithmen *greedy*
  - Werte nehmen nur noch ab

T sinkt dabei ständig

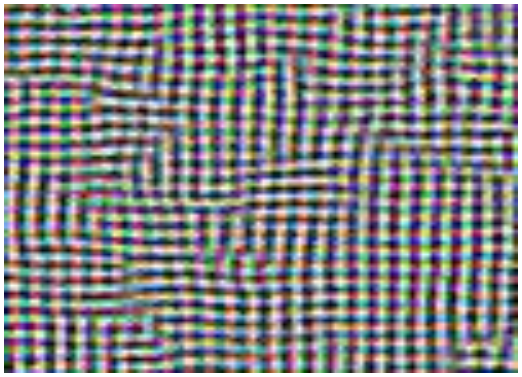
Simulated Annealing

Der Algorithmus konvergiert hin zu einem globalen Optimum für $t \rightarrow \infty$ (in Realiät ist t_{inf} ein Wert, der größer als die für vollständige Suche benötigte Zeit ist)



Schnelle Abkühlung

Langsame Abkühlung



Beispiel: Neuanordnung der Pixel eines Bildes zur Minimierung einer Potentialenergie. Bildpunkte mit ähnlichen Farben ziehen sich bei kurzem Abstand an und stoßen sich bei größeren Abständen ab. Elementaroperationen vertauschen zwei Pixel.

Simulated Annealing

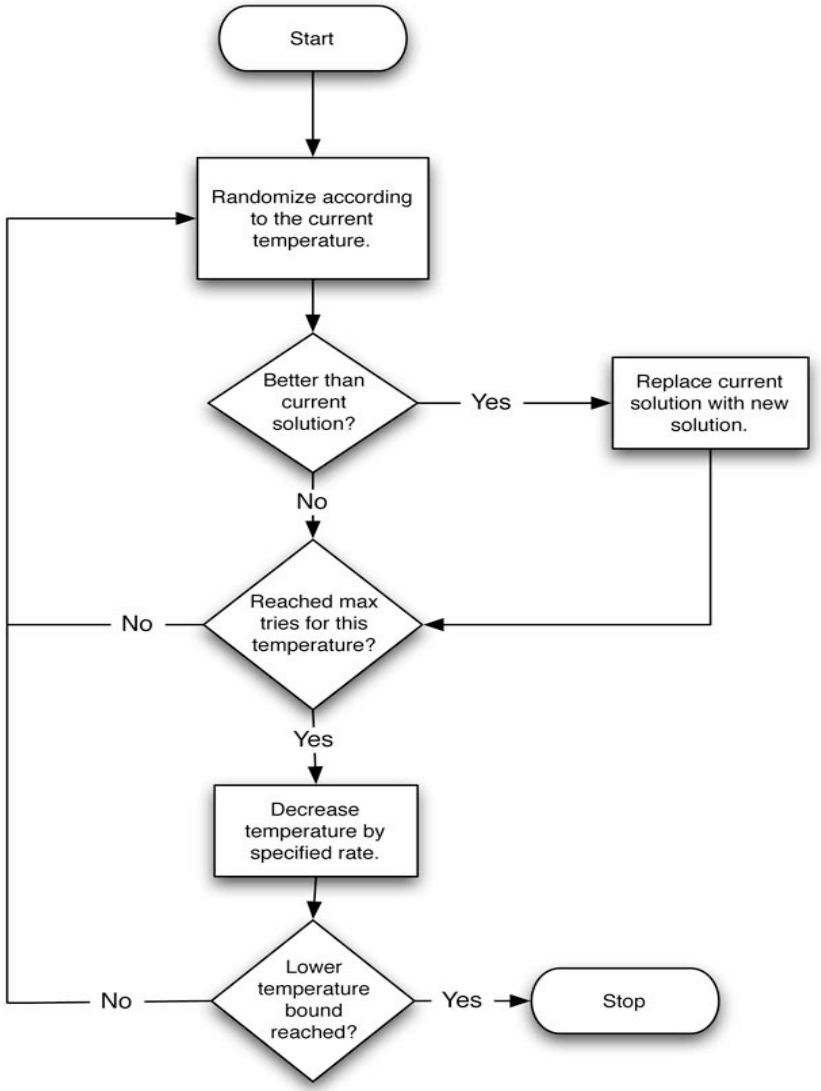
Pseudocode:

```
s ← s0; e ← E(s)                                    // Initial state, energy.
sbest ← s; ebest ← e                                // Initial "best" solution
k ← 0                                               // Energy evaluation count.
while k < kmax and e > emax                         // While time left & not good enough:
  snew ← neighbour(s)                               // Pick some neighbour.
  enew ← E(snew)                                    // Compute its energy.
  if P(e, enew, temp(k/kmax)) > random() then       // Should we move to it?
    s ← snew; e ← enew                              // Yes, change state.
  if enew < ebest then                              // Is this a new best?
    sbest ← snew; ebest ← enew                      // Save 'new neighbour' to 'best found'.
  k ← k + 1                                         // One more evaluation done
return sbest                                        // Return the best solution found.
```

Simulated Annealing

Pseudocode:

```
s ← s0; e ← E(s)
sbest ← s; ebest ← e
k ← 0
while k < kmax and e > emax
  snew ← neighbour(s)
  enew ← E(snew)
  if P(e, enew, temp(k/kmax)) > random() then
    s ← snew; e ← enew
  if enew < ebest then
    sbest ← snew; ebest ← enew
  k ← k + 1
return sbest
```



Simulated Annealing

- Der eigentliche Sim. Ann.-Algorithmus merkt sich die bisherige beste Lösung nicht
  - keine Verwendung der Variablen sbest and ebest
  - Kein zweites **if** innerhalb der Schleife
  - Liefert aktuellen Zustand s an Stelle von sbest
- Merken des besten Zustand ist Standardtechnik bei Optimierungen, die in jeder Metaheuristik verwendet werden kann
- Keine Analogie zu physikalischer Abkühlung
- Phys. System kann nur einen Zustand speichern

```
s ← s0; e ← E(s)
sbest ← s; ebest ← e
k ← 0
while k < kmax and e > emax
  snew ← neighbour(s)
  enew ← E(snew)
  if P(e, enew, temp(k/kmax)) > random() then
    s ← snew; e ← enew
  if enew < ebest then
    sbest ← snew; ebest ← enew
  k ← k + 1
return sbest
```

Parameterwahl bei Simulated Annealing (1)

- Anfangstemperatur T_{start}
  - muss groß genug sein, um Unabhängigkeit der sich am Ende ergebenden Lösung von der Startlösung zu gewährleisten
  - Verwende eine „Anheizphase“, in der T_{start} so bestimmt wird, dass eine gewünschte Akzeptanz-rate resultiert (i.d.R. zwischen 40% und 60%)
- Kühlplan d(T)
  - Meist geometrische Funktion
    - d(T) = fT mit f [0,8; 0,99]
  - Langsames Kühlschema
    - d(T) = T/(1+fT) mit f>0 und klein

Parameterwahl bei Simulated Annealing (2)

- Anzahl HT von Lösungsnachbarn
  - HT muss groß genug sein, um ausreichende Untersuchung der Nachbarschaft zu gewährleisten (sonst „Hängenbleiben“)
  - Hohes HT anfangs sinnlos, da lediglich zu anderen zufälligen Startlösungen übergegangen wird
  - Gegen Ende intensivere Untersuchung notwendig, um Optimum zu erreichen

- Folge: erhöhe HT additiv oder geometrisch mit Reduktion der Temperatur

Parameterwahl bei Simulated Annealing (3)

- Abbruchkriterien
  - werden häufig in Kombinationen verwendet:
  - Erreichen der Endtemperatur oder Überschreiten einer maximalen Anzahl von Iterationen
    - T_{End} unterschritten
    - k hat Schwellenwert erreicht
  - Während einer fixierten Anzahl von Iterationen oder Temperaturreduktionen wurde keine verbesserte Lösung ermittelt
    - z.B. 100000-mal ohne Verbesserung gelaufen
  - Akzeptanzrate fällt unter einen vorgegebenen Wert
    - Verteilung liefert nur noch Werte unter z.B. 3%

Simulated Annealing: Beispiel Traveling-Salesman-Problem

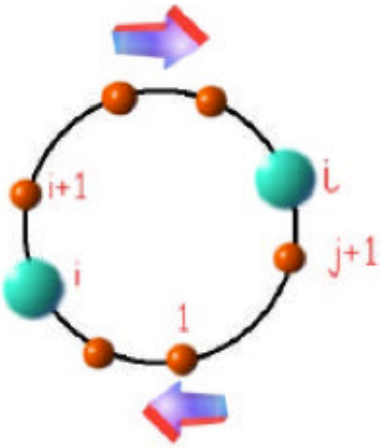
n Städte	alle Städte in Koordinatensystem eintragen und durchnummerieren
i_{start}	zufällige Permutation wählen
Nachbarschafts-bestimmung	durch „ 2-opt“
$f(i)$	aufsummieren d. euklid. Distanzen mit: $f(i) = S_{k=1} = \text{sqrt}((xk-xk+1)^2+(yk-yk+1)^2)$
d(T)	f*T mit f =0.995
HT	100n
d(HT)	f*HT mit f =1.25
Stoppkriterium	k = 1000x erfolglos iteriert oder T < 0,005

TSP und Simulated Annealing: 2-opt

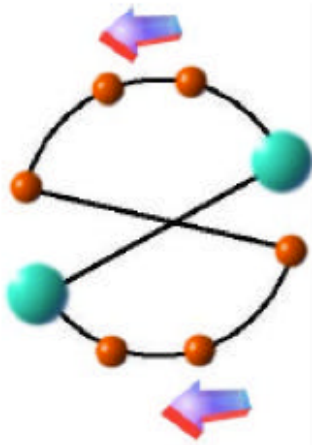
- 2-opt-Verfahren:
  - Zug zur Bestimmung eines j basierend auf i aus S_i

- Überführe (1,...,i,i+1, ..., j,j+1, ...,n)
  durch Umdrehen der Sequenz (i+1,...,j)
  in (1, ...,i, j,...,i+1, j+1,...,n):

(1,...,i,**i+1, ..., j**,j+1, ...,n)



(1, ...,i, **j,...,i+1,** j+1,...,n)



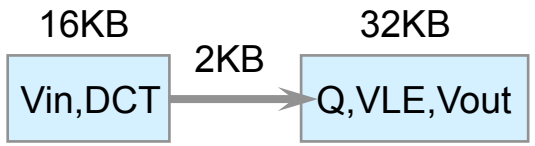
TSP und Simulated Annealing: Ablauf

1. Generierung einer neuen Rundreise j aus S_i durch 2-opt
   1. Iterator, der HT zählt, hinaufsetzen
2. Berechnung der neuen Gesamtlänge von j mit *f*
3. Vergleich:

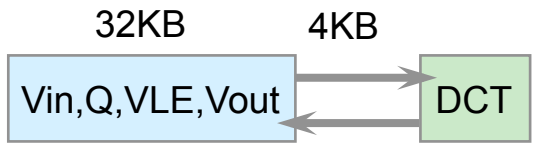
<table>
<tr><td colspan="2">j kürzer als i</td><td colspan="2">j länger als i
rand[0;1] > e^(-(f(i)- f(j))/T) ?</td></tr>
<tr><td colspan="2">i := j</td><td>ja
i := j</td><td>nein
-</td></tr>
<tr><td colspan="4">HT erreicht ???</td></tr>
<tr><td colspan="2">ja</td><td colspan="2">nein</td></tr>
<tr><td colspan="2">senke T, berechne HT</td><td colspan="2">gehe zu 1.</td></tr>
<tr><td colspan="2">Stoppkriterium erreicht ?</td><td colspan="2"></td></tr>
<tr><td>ja</td><td>nein</td><td colspan="2"></td></tr>
<tr><td>i = Lösung</td><td>gehe zu 1.</td><td colspan="2"></td></tr>
</table>

Beispiel: JPEG/JPEG200

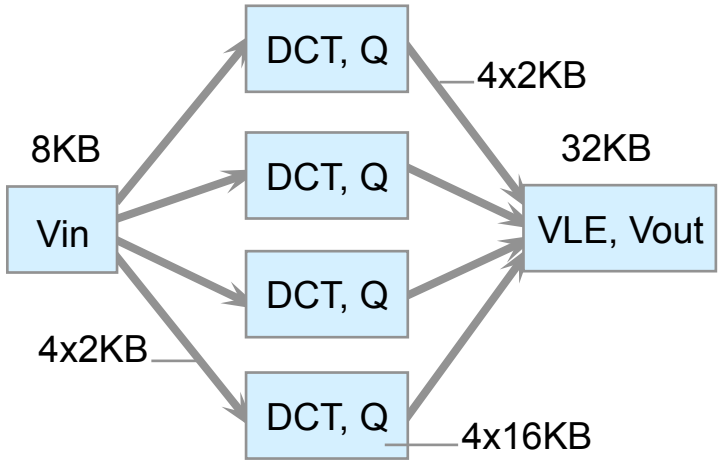
- **Architekturinstanzen für einen JPEG-Encoder:**



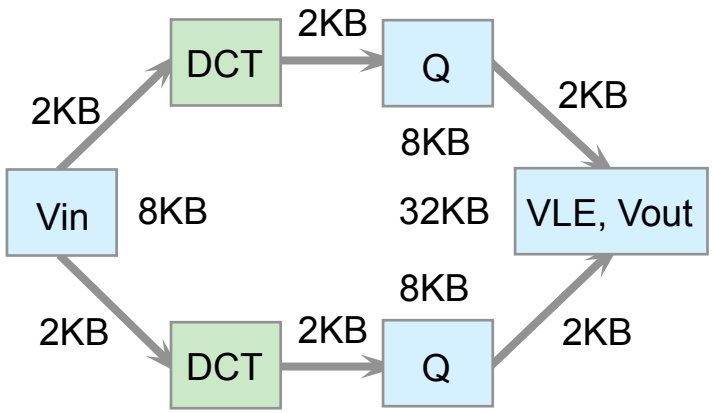
2 MicroBlaze processors (50KB)



1 MicroBlaze, 1HW DCT (36KB)

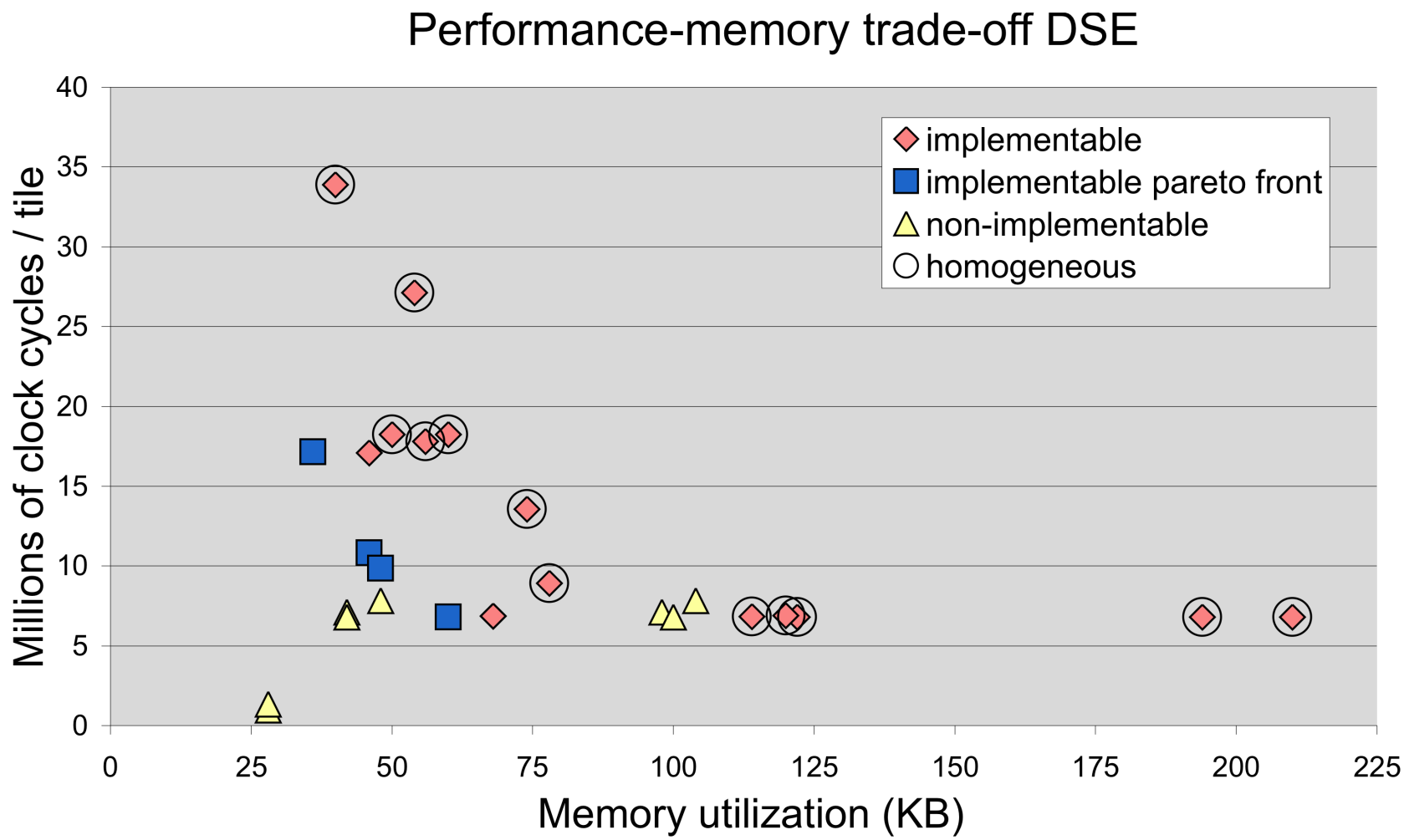


6 MicroBlaze processors (120KB)



4 MicroBlaze, 2HW DCT (68KB)

Ergebnisse der Entwurfsraumerkundung (Sesame): JPEG-Encoder



Weitere MPSoC-Entwurfsframeworks auf Systemebene

- System on Chip Environment der UC Irvine
- SystemCoDesigner der Universität of Erlangen
- Metropolis der UC Berkeley
- Koski der University of Tampere
- PeaCE/HOPES der Seoul National University
- …

Zusammenfassung

- Daedalus
- Simulated Annealing
- Beispiele