Algorithmen im Chip-Entwurf 5

Längenmaße
und
Platzierung: VPR

Andreas Koch
FG Eingebettete Systeme
und ihre Anwendungen
TU Darmstadt

Überblick

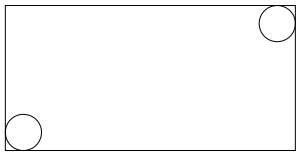
- Längenmaße: Halber Umfang
- Arten von Platzierungsproblemen: MPGA/FPGA
- Konkreter FPGA-Placer: VPR
- Zusammenfassung

Verdrahtungsfläche

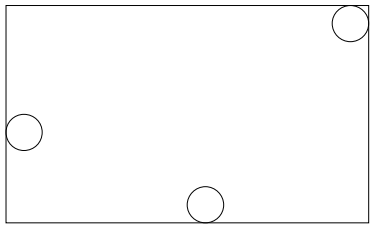
- Mögliches Platzierungs-Qualitätskriterium
  - Gesamtfläche für Verdrahtung
    - Nur bei ASIC
    - Bei FPGA: Feste Anzahl von Leitungen, Länge reicht
- Aber: Vollständiges Routing zu komplex
  - NP-vollständig
- Abschätzen der Länge durch Metrik
  - Einzeln pro Netz
  - Aufsummieren der Teillängen
  - Multiplizieren mit angenommener
    - Leitungsbreite plus
    - Leitungsabstand

Längenmetriken 1

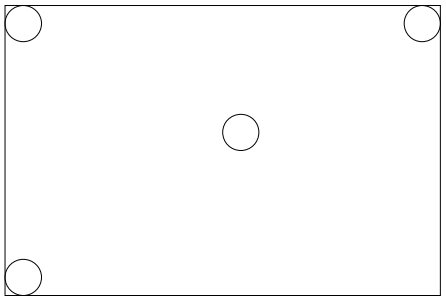
- Halber Umfang (half perimeter)
  - Rechteck um alle Terminals des Netzes



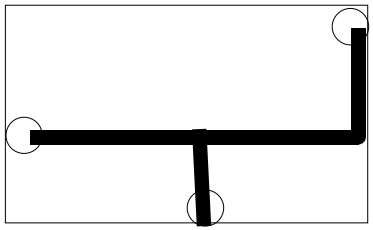
exakt

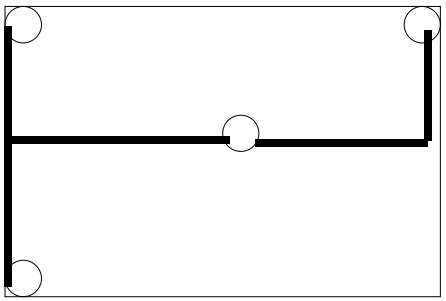


exakt



untere Grenze

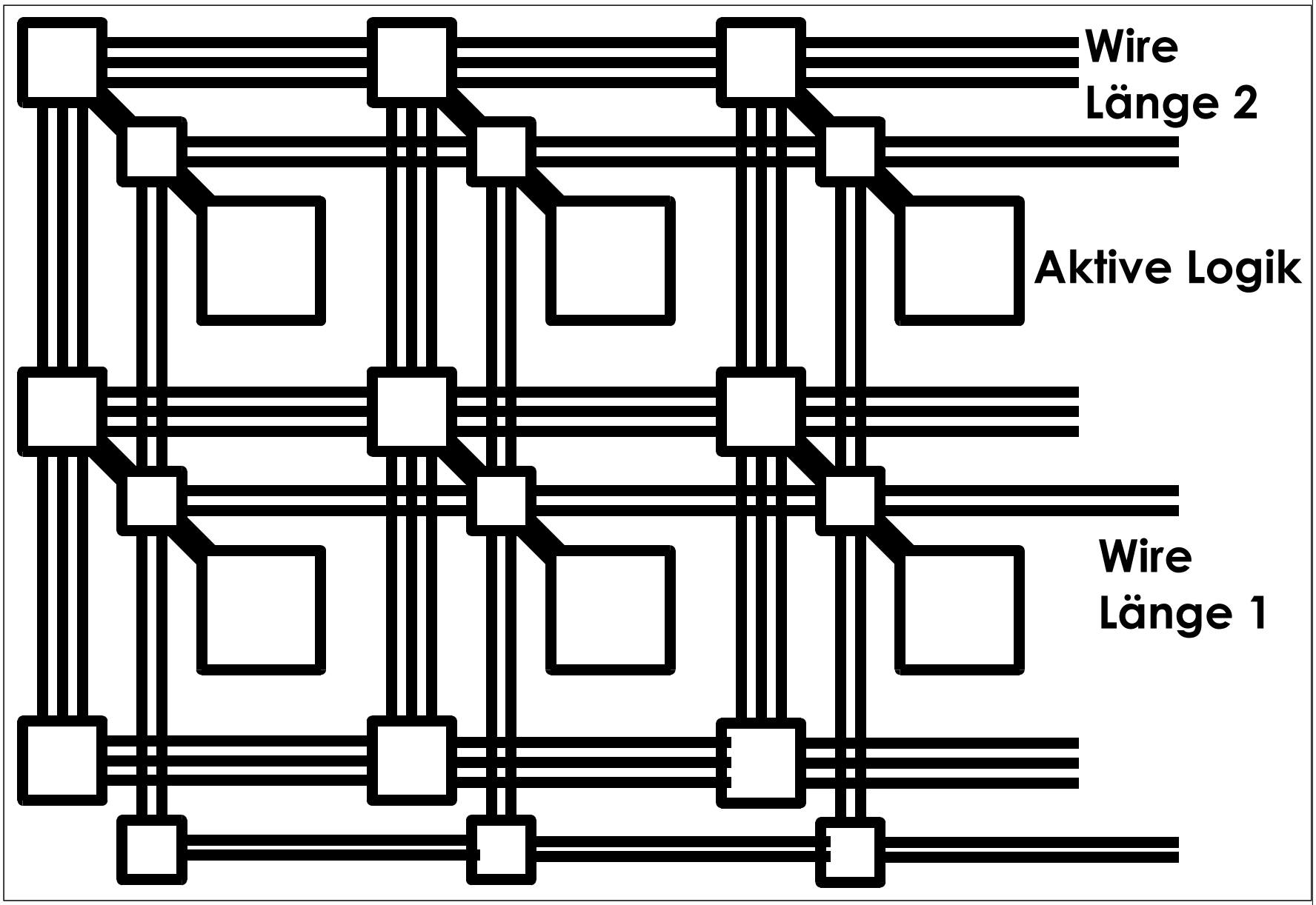




MPGA/FPGA 1

- Mask-Programmable Gate Array
  - Modebezeichnung: Structured ASIC
- Field-Programmable Gate Array
- Feste Anordnung von
  - Logik
  - Verdrahtung
- Anpassung auf Anwendung
  - MPGA: Beim Hersteller (Metalllagen)
  - FPGA: Beim Anwender (Programmierung)

MPGA/FPGA 2



MPGA/FPGA 3

- Sehr ähnlich zu UPP
- Aber: Segmentierte Verbindungen
  - Mehrere Verdrahtungslängen
- Verzögerung abhängig von
  - Anzahl durchlaufener Switch Boxes
  - Last (Fan-Out)
- Feste Verdrahtungskapazität
- ➢ Nicht jede Platzierung verdrahtbar
- Verdrahtbarkeit in Kostenfunktion

VPR

- Versatile Place and Route
  - Betz und Marquardt, U Toronto
  - Ab hier Auszüge aus Paper auf Web-Seite

- Platzierer
  - Simulated Annealing-basiert
    - Adaptive Annealing Schedule
  - Optimiert gleichzeitig
    - Leitungslänge
    - Verzögerung

Züge

- Paarweises Austauschen von Blöcken
  - N_{blocks} = Größe der Schaltung
- Aber nicht ganz wahllos
  - Beschränkung der Entfernung

Starttemperatur

- Wird automatisch bestimmt
  - Für aktuelle Schaltung passend
- Idee:
  - Anfangs fast alle Züge akzeptieren
  - Wie hoch muss die Starttemperatur sein?
- Vorgehen
  - N_{blocks} paarweise austauschen
  - Beobachte Änderung der Kostenfunktion x
    - Standardabweichung

$$s_x = \sqrt{\frac{1}{n-1}\left(\left(\sum_i x_i^2\right) - n\bar{x}^2\right)}$$

  - Starttemperatur = $20 \cdot s_x$

Thermal Equilibrium

- Anzahl von Schritten pro Temperaturstufe:

$$10\, N_{blocks}^{4/3}$$

- 10x schneller, aber nur ca. 10% schlechter:

$$N_{blocks}^{4/3}$$

Abkühlen 1

- Beobachtung
  - Anfangs: T hoch, fast alle Züge akzeptiert
    - Im wesentlichen zufälliges Bewegen
    - Keine echte Verbesserung der Kostenfunktion
  - Ende: T niedrig, kaum Züge akzeptiert
    - Fast keine Bewegung mehr
    - Wenig Veränderung in Kostenfunktion
- Idee
  - Meiste Optimierung passiert dazwischen
  - Bringe T schnell in den produktiven Bereich
  - Halte T lange im produktiven Bereich
- Vorgehen
  - Steuere T anhand der Akzeptanzrate

Abkühlen 2

- $T_{new} = \alpha T_{old}$

α	Acceptance Rate R_a
0.50	$R_a > 0.96$
0.90	$0.80 < R_a \leq 0.96$
0.95	$0.15 < R_a \leq 0.80$
0.80	$R_a \leq 0.15$

Abkühlen 3

- Vorahnung
  - Gute Fortschritte bei $R_a \approx 0{,}5$
- Am effizientesten $R_a = 0{,}44$
  - Beste Fortschritte
- Idee
  - R_a möglichst auf diesem Wert halten, aber wie?
  - *Nicht* temperaturbasiert (kühle nur ab!)
  - Sondern: Auswirkungen der Züge beeinflussen
  - Beobachtung
    - Weite Züge: Grosse Änderung der Kostenfunktion
    - Kurze Züge: Kleine Änderung der Kostenfunktion
- Vorgehen
  - Variiere Zugweite R_{limit}, um $R_a \approx 0.44$ zu halten

Abkühlen 4

- R_{limit} klein
  - Kleine Zugreichweite
  - Kleine Änderungen der Kostenfunktion
  - Kleine Verschlechterungen
    - Werden eher angenommen
  - R_a steigt
- R_{limit} gross
  - Grosse Zugreichweite
  - Grosse Änderungen der Kostenfunktion
  - Große Verschlechterungen
    - Werden eher abgelehnt
  - R_a sinkt

Abkühlen 5

- Anfangs: R_{limit} = ganzer Chip L_{Chip}
- Bei jedem Abkühlschritt:

$$R_{limit}^{new} = R_{limit}^{old}\left(1 + R_a^{old} - 0.44\right), 1 \leq R_{limit}^{new} \leq L_{Chip}$$

  - Zuviel akzeptiert: R_{limit} grösser machen
  - Zuwenig akzeptiert: R_{limit} kleiner machen

Abbruchbedingung

- Wann Abkühlung beenden?
- Idee
  - Erkennung von Stillstand
- Vorgehen
  - Jeder Zug beeinflusst mindestens ein Netz
  - Bestimme die durchschnittlichen Kosten pro Netz
  - Wenn T kleiner als Bruchteil davon ...
    - Nur noch kleine Chance, dass Zug akzeptiert wird
    - $T < 0.005$ Cost/#Nets
  - Auch einfachere Realisierungen möglich
    - Letzte *k* Züge ohne akzeptierten Zug
    - Letzte *k* Züge ohne Verbesserung von BSF
    - ...

Kostenfunktion 1. Teil

- Gleichzeitig optimieren
  - Zeitverhalten
  - Verdrahtungslänge
- Verdrahtungslänge
  - Bestimmt als korrigierter halber Netzumfang

$$c_w = \sum_{n \in N} \mathrm{q}\left(n_{pincount}\right)\left[\mathrm{bb}_x(n) + \mathrm{bb}_y(n)\right]$$

q(*i*) = 1 für *i*=1..3, =2.79 für i=50 (Cheng 1994)

  - Web-Seite: Paper, Datei mit Korrekturfaktoren q(*i*)

Inkrementelle Berechnung 1

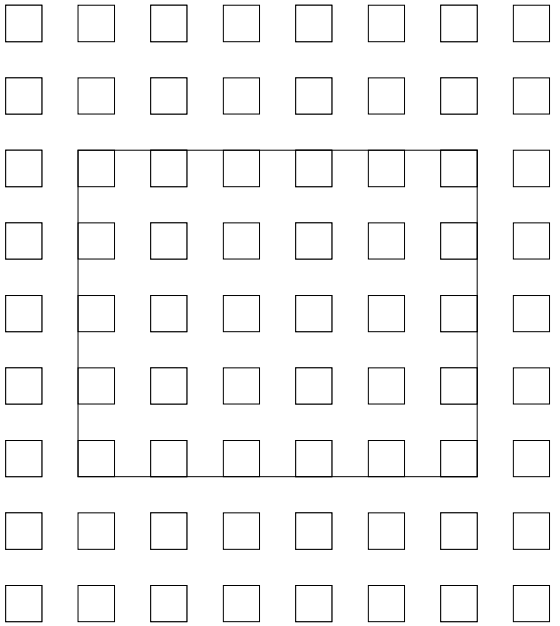
- Berechnung des Netzumfangs
  - Simpel: O(*k*), *k* Anzahl der Pins
  - Problem: k = 100 ... 1000 realistisch
  - Nach jedem Zug neu berechnen
- Besser:
  - Nach Möglichkeit nur bewegte Pins neu berechnen
    - Ein Pin ist nur in einem Netz
    - Ein Block hat aber mehrere Pins
- Vorgehen
  - Je Netz umspannendes Rechteck speichern
    - $(x_{min}, x_{max}, y_{min}, y_{max})$
      - Position der Seiten
    - $(N_{xmin}, N_{xmax}, N_{ymin}, N_{ymax})$
      - Anzahl Pins direkt auf den Seiten

Inkrementelle Berechnung 2

Betrachtet nur linke Seite (xmin)

- **Bewege Terminal von x_{old} nach x_{new}**
- **Netz an Terminal: n**

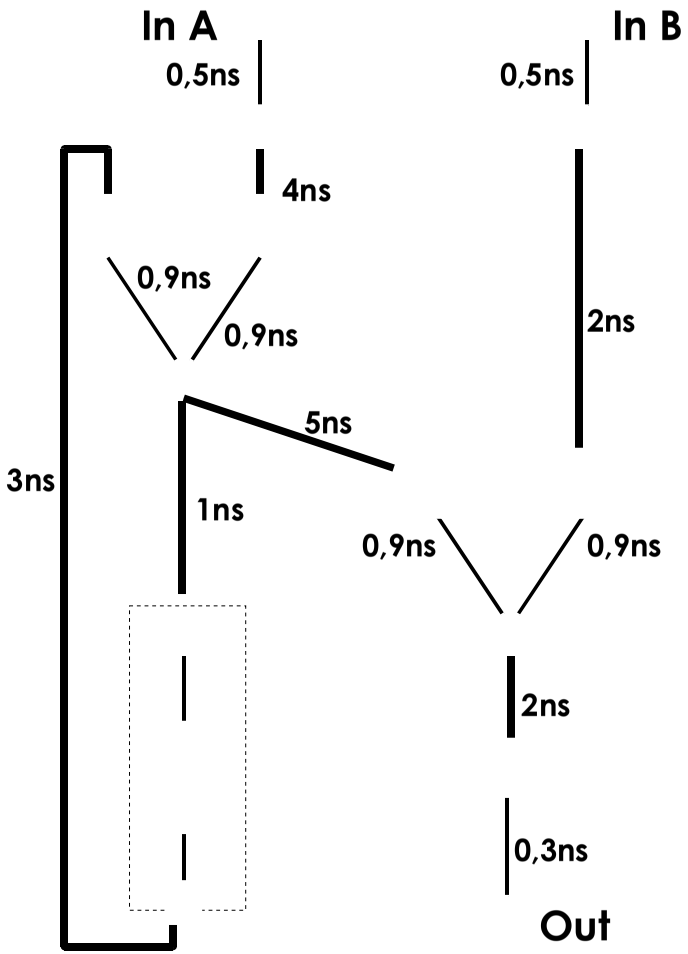
```
If (xnew != xold) {  // horiz. bewegt
        if (xnew < n.xmin) {
                n.xmin = xnew;
                n.Nxmin = 1;
        } else if (xnew == n.xmin) {
                n.Nxmin++;
        } else if (xold == n.xmin) {
                if (n.Nxmin > 1) {
                        n.Nxmin--;
                } else {
                        BruteForce(n);
                }
        }
}
```



(1,1)

xmin=2	Nxmin=1
xmax=7	Nxmax=2
ymin=3	Nymin=3
ymax=7	Nymax=1

Kosten 2. Teil: Zeitverhalten 1



- Betrachte
  - Platzierungs-abhängiges Zeitverhalten
- Punkt-zu-Punkt Verbind.
- Von
  - Netzquelle *u*
- Zu
  - Jeder Netzsenke *v*
- Sicht: *Two-Terminal-Nets*
  - $E_{\text{NetTiming}} \subset E_{\text{Timing}}$
- Zeitverhalten
  - Bestimmt aus Slacks
  - *Nicht* auf Pfaden (langsam)

Zeitverhalten 2

- „Wichtigkeit“ einer Verbindung
  - Punkt-zu-Punkt zwischen Terminals *u* und *v*

$$\text{Criticality}(u,v) = 1 - \frac{\text{slack}(u,v)}{D_{max}}$$

  - (u,v) auf kritischem Pfad
    - slack(u,v) = 0 ⇔ Criticality(u,v) = 1
  - (u,v) absolut unkritisch
    - slack(u,v) = D_{max} ⇔ Criticality(u,v) = 0

- Timing Cost: Delay(*u*,*v*) ist Schätzung!
  - Noch kein „echtes“ Routing

$$c_t = \sum_{(u,v) \in E_{NetTiming}} \text{Delay}(u,v)\,\text{Criticality}(u,v)^{CriticalityExponent}$$

Zeitverhalten 3

- Criticality Exponent
  - Gewichtet kritischere Verbindungen höher
    - Wenige kritische Verbindungen dominieren c_t
  - Untergewichtet unkritischere Verbindungen
    - Fallen fast ganz aus c_t Berechnung heraus
- Idee
  - Gegen Ende auf kritische Netze konzentrieren
- Vorgehen:
  - Steigern von ce_{start}=1 auf ce_{final}=8 (experimentell)

$$\mathrm{CritExp} = \left(1 - \frac{R_{limit}^{now} - 1}{R_{limit}^{start} - 1}\right) \cdot \left(ce_{final} - ce_{start}\right) + ce_{start}$$

Zeitverhalten 4

- slack() ist platzierungsabhängig
  - Unkritische Netz können kritisch werden
    - Zu lange Leitungslängen
  - Kritische Netze können unkritisch werden
    - Sehr kurze Leitungslängen
- Slack-Werte müssen aktualisiert werden
  - Timing-Analyse: T_a, T_r
- Wie oft?
  - Nach jedem Zug? Nach N Zügen?
  - N-mal pro Temperaturstufe?
  - Alle N Temperaturstufen?
- Bewährt:
  - 1x pro Temperaturstufe

Gesamtkostenfunktion

- Selbstnormierend

$$\Delta c_w = c_w(g) - c_w(f)$$

$$\Delta c_t = c_t(g) - c_t(f)$$

$$\Delta c = \lambda \frac{\Delta c_t}{c_t^{old}} + (1-\lambda) \frac{\Delta c_w}{c_w^{old}}$$

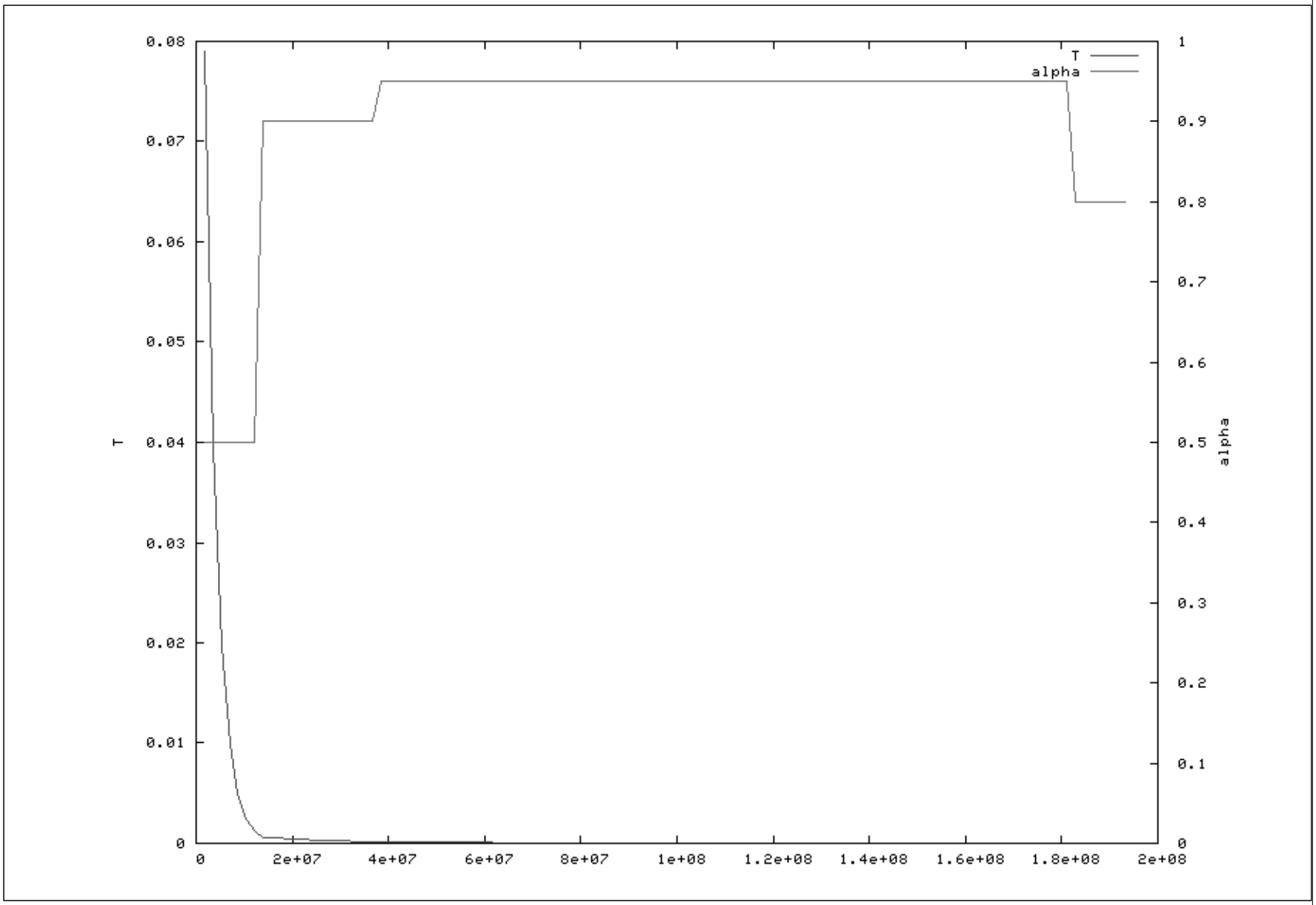
- λ gewichtet Zeit ./. Längenoptimierung
  - Aber $\lambda=1$ erzeugt <u>nicht</u> die schnellste Lösung
  - Netze wechselnd kritisch/unkritisch
    - Nicht erkannt, da Timing-Analyse nur 1x pro Temp.
  - Besser $\lambda=0.5$
    - Längenmaß wirkt als Dämpfer für Oszillation

Gesamtalgorithmus

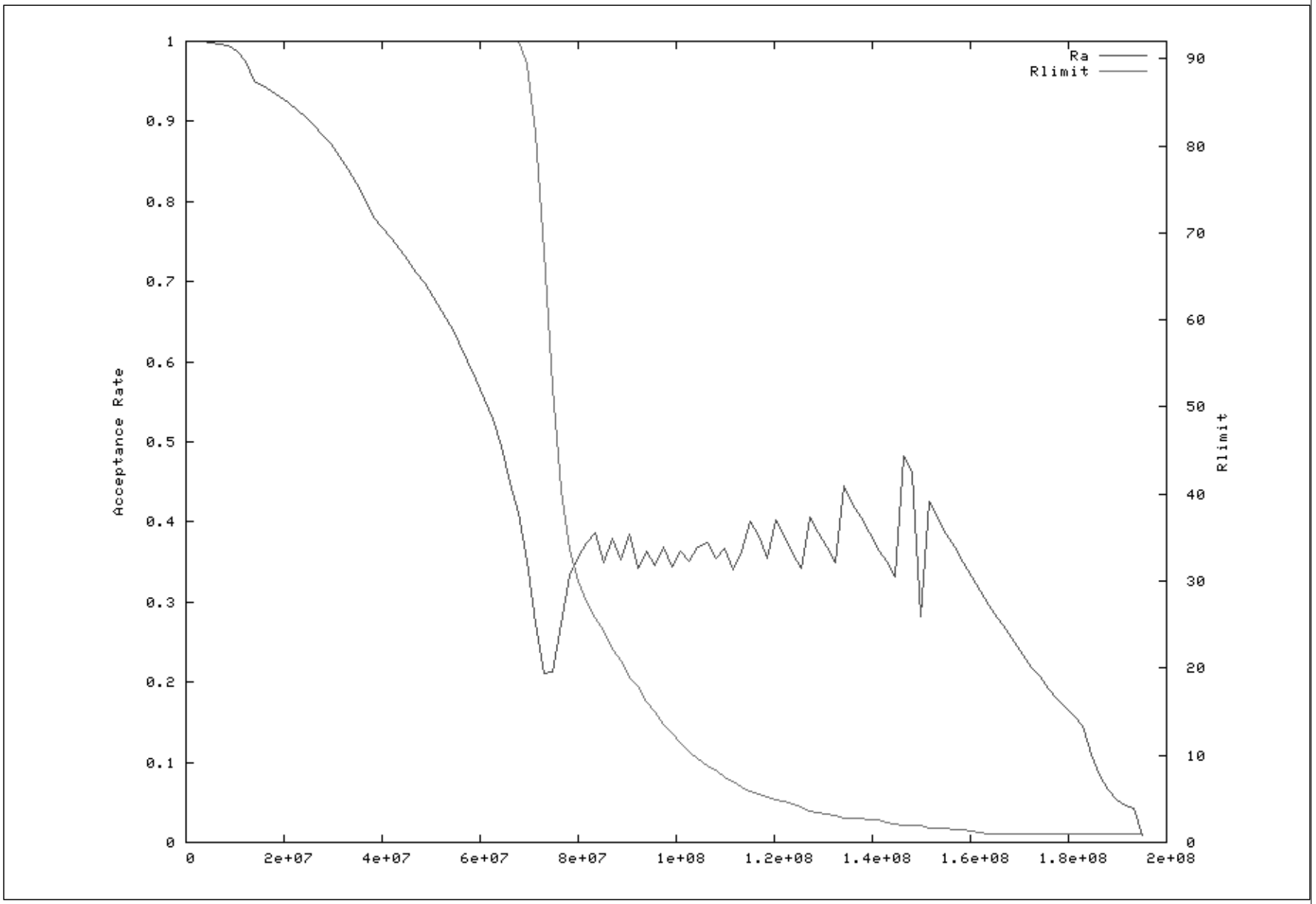
```
S = RandomPlacement();
T = InitialTemperature();
Rlimit = InitialRlimit();
CritExp = ComputeNewExponent(Rlimit);

while (!ExitCriterion()) {
    TimingAnalyze();                    // Bestimme Ta, Tr und slack()
    OldWiringCost = WiringCost(S);      // für Normalisierung der Kostenterme
    OldTimingCost = TimingCost(S);
    while (!InnerLoopCriterion()) {     // eine Temperaturstufe
        Snew = GenerateSwap(S, Rlimit);
        ΔtimingCost = TimingCost(Snew) – TimingCost(S);
        ΔwiringCost = WiringCost(Snew) – WiringCost(S);
        ΔC = λ (ΔtimingCost/OldTimingCost) + (1-λ) (ΔwiringCost/OldWiringCost);
        if (ΔC =< 0)
            S = Snew;
        else
            if (random(0,1) < exp(-ΔC/T))
                S = Snew
    }
    T = UpdateTemp();
    Rlimit = UpdateRlimit();
    CritExp = ComputeNewExponent(Rlimit);
}
```

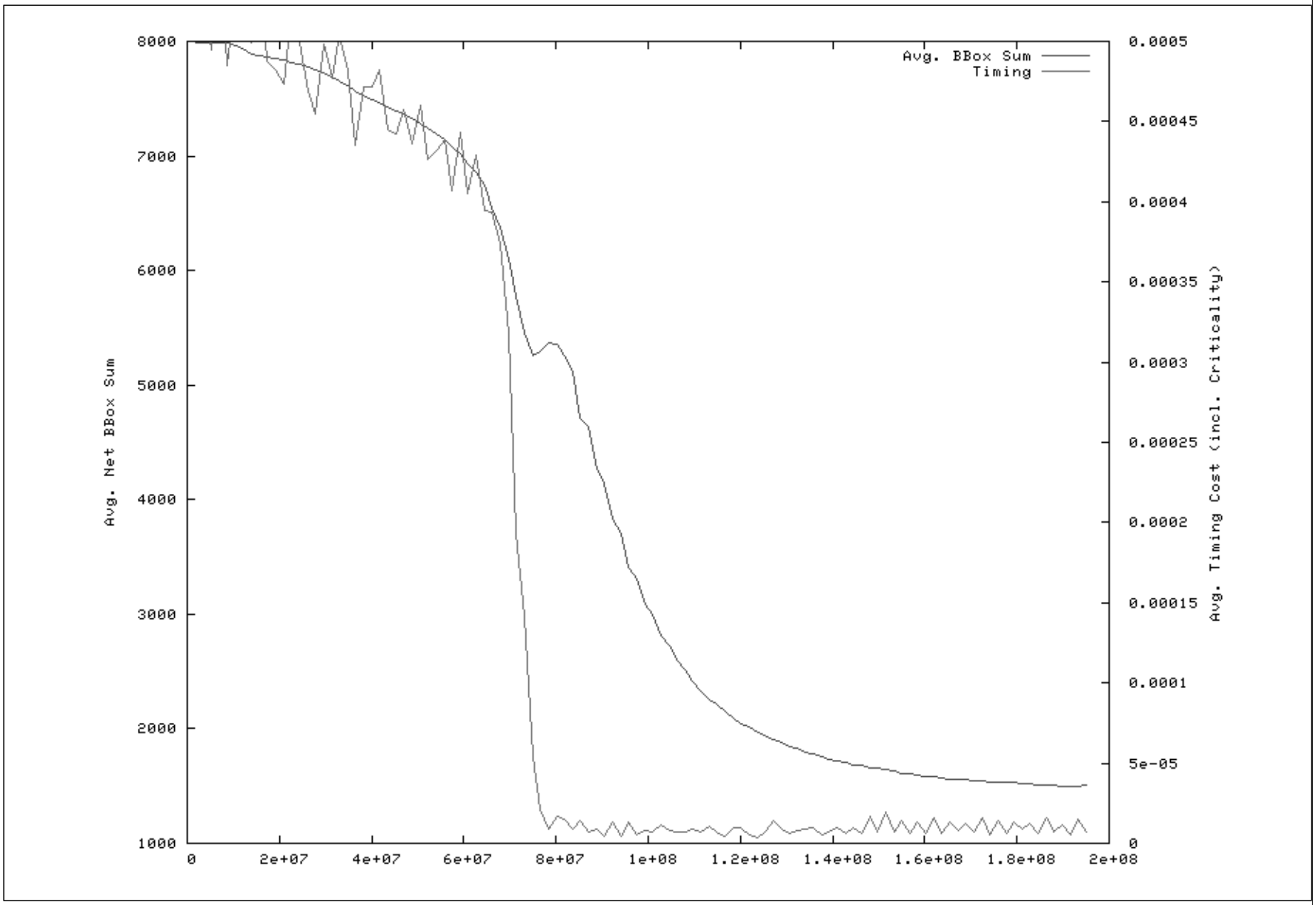
VPR Simulated Annealing 1



VPR Simulated Annealing 2



VPR Simulated Annealing 3



Weiteres Vorgehen

- Für 1. Abgabe der 5 SWS'ler: 15.11.2010
  - Donnerstag, 18.11.: Kolloquien
    - Termine mit Florian Stock vereinbaren
  - Freitag, 26.11.: Vorträge (je ca. 20 Minuten)
    - In gleicher Reihenfolge

Zusammenfassung

- Längenmaße
- VPR
  - Adaptives Simulated Annealing
  - Selbstnormalisierende Kostenfunktion
  - Schnelle Netzumfangsberechnung
  - Gesamtalgorithmus
- Papers auf Web-Seite
  - Cheng 1994: q(i) Korrekturfaktoren
    - ... sonst eher schlecht zu lesen
  - Marquardt & Betz: VPR
    - 1997 Grundlagen
    - 2000 Timing-gesteuerte Betriebsart (Criticality, etc.)