Algorithmen für Chip-Entwurfswerkzeuge

Platzierungsverfahren

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorlesung
WS 2018/2019

Florian Stock

Eingebette Systeme und Anwendungen
Technische Universität Darmstadt

Embedded Systems & Applications

Klausur

TECHNISCHE UNIVERSITÄT DARMSTADT

- Klausurtermin:
  27.02.19 von 10-12 Uhr in S101/A02

Platzierungsproblem allgemein
Design Stile

TECHNISCHE UNIVERSITÄT DARMSTADT

- Design Stile:
  - Standardzellen
  - Gate Arrays
  - Makroblock
  - Mixed-Size
  - "*UPP*" - Einfaches Beispielproblem
- Alle Gleich:
  Platziere die zur Verfügung stehenden Module Überlappungsfrei!
- Unterschiedlich in Zielfunktionen und Randbedingungen

Design Stil
Standarzellen-Design

TECHNISCHE UNIVERSITÄT DARMSTADT

- Semi-Custom
- Bibliothek mit Logikmodule mit einfachen Funktionen (AND, OR, Inverter, . . .)
  - Alle Module haben die gleiche Höhe
  - Module habe variable Breite
- Es gibt für das Platzieren vordefinierte Reihen
- Sehr beliebter Design-Ansatz

$\Rightarrow$ Viele Algorithmen gehen von Standardzellen Design aus

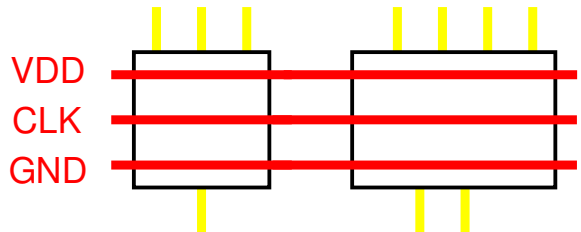
- Platzierung überlappungsfrei innerhalb der Reihen

Design Stil
Standarzellen-Design

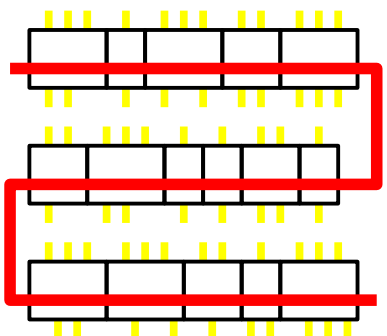
TECHNISCHE UNIVERSITÄT DARMSTADT

Routing:

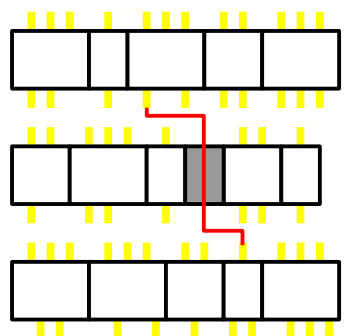
- Infrastruktur (V_{DD}, *CLK*, *GND*) durch alle Reihen



- Verdrahtung zwischen Reihen
- Ausnahmen:



Angrenzende Verbindungen (abutment)

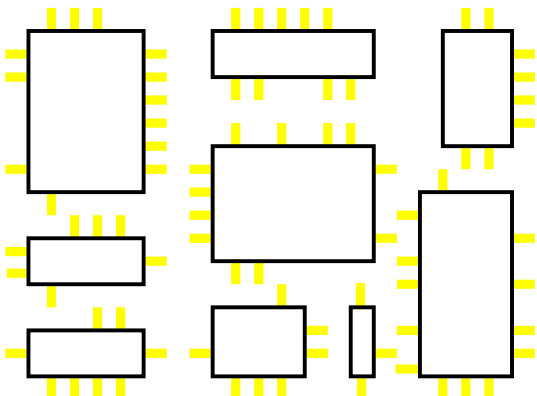


Durchleitungen (feedthroughs)

Design Stil
Makroblock-Design

TECHNISCHE UNIVERSITÄT DARMSTADT

- Alle Module sind Makroblöcke (Building-Blocks) fester Größe, Form und Ausrichtung
  - Kann auch Full-Custom Teile enthalten
  - Automatisch generierte Blöcke (z.B. RAM)
- Verdrahtungskanäle an allen Seiten
- Alle sind überlappungsfrei zu platzieren
- Ähnlich zu Floorplanning (dort sind i.d.R. Form und Orientierung variabel).



Design Stil
Mixed-Size-Design

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sehr häufig benutzt
- Vereint
  - Makroblöcke und
  - Standardzellen
- Makroblöcke $\gg$ Standardzellen
  $\Rightarrow$ Schwer Überlappungen zu vermeiden

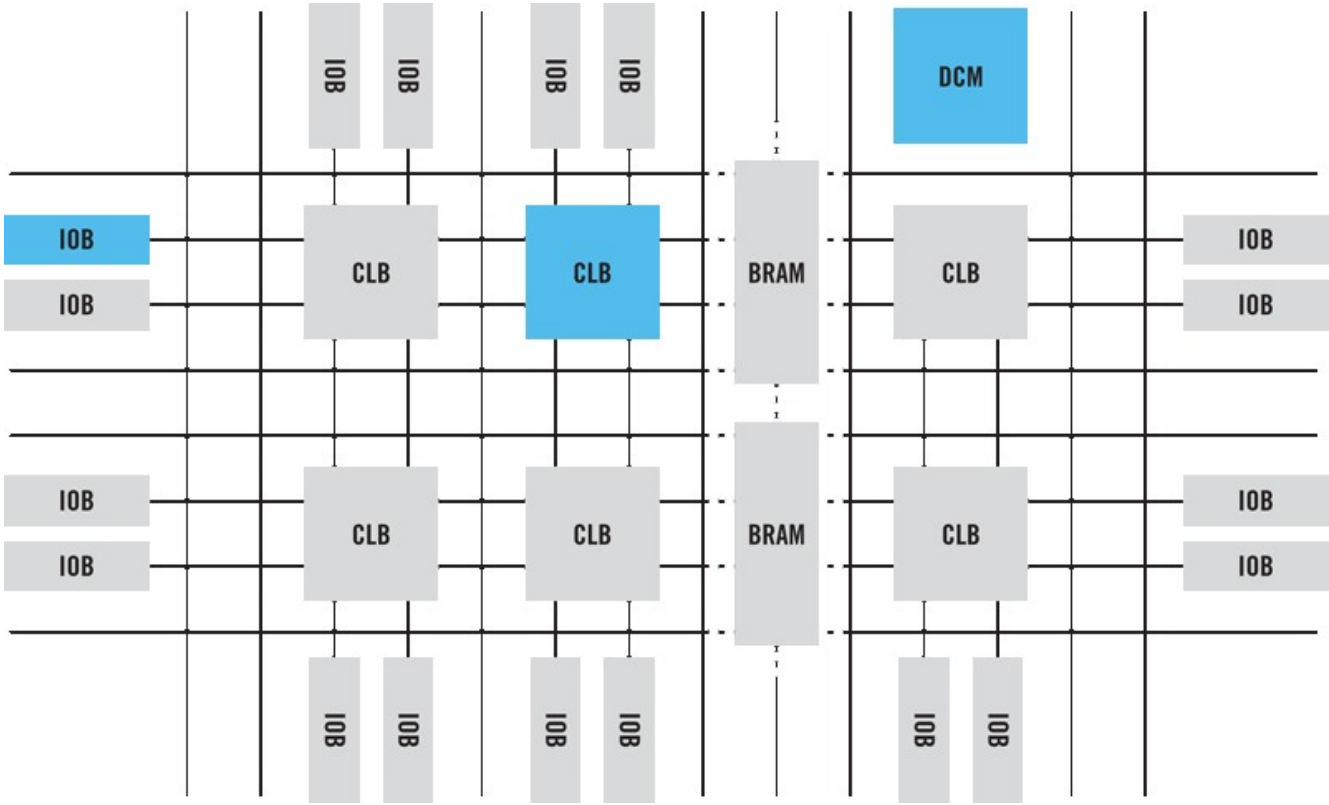
Zielarchitektur
MPGA/FPGA

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gate Arrays
  - Field Programmable Gate Array (FPGA)
    Wird beim Anwender an Funktion angepasst $\Rightarrow$ Programmierung
  - Mask Programmable Gate Array (MPGA)
    Andere Bezeichnung: Structured ASIC
    Wird beim Hersteller an Funktion angepasst $\Rightarrow$ Metallagen Herstellung
- Reguläre Struktur mit fester Anordnung von
  - Programmierbarer Logik
  - Festen Funktionsblöcken
  - Speicher
  - Verdrahtung

Zielarchitektur
MPGA/FPGA

TECHNISCHE UNIVERSITÄT DARMSTADT



[Quelle: Xilinx]

Zielarchitektur
MPGA/FPGA

TECHNISCHE UNIVERSITÄT DARMSTADT

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

Platzierungsproblem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Platzierungsproblem in nahezu allen Varianten NP-Vollständig
- Zur Lösung:
  - Heuristische Algorithmen
    Simulated Annealing
  - Analytische Algorithmen
    kräftebasiert, partionierungsbasiert, ...

Simulated Annealing
TimberWolf (1985)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Standard Cell-Placer
- Start mit $T = 4.000.000$
- Stop bei $T < 0.1$
- Equilibrium abhängig von Problemgröße
  - 100 Züge pro Zelle bei 200 Zellen
  - 700 Züge pro Zelle bei 3000 Zellen
- Abkühlen
  - Anfangs mit $T_n = 0.8T$
  - Im Mittelbereich mit $T_n = 0.95T$
  - Gegen Ende mit $T_n = 0.8T$

  $\Rightarrow$ Cooling Schedule

VPR

TECHNISCHE UNIVERSITÄT DARMSTADT

- Versatile Place and Route
  - Betz und Marquardt, University of Toronto
  - Ab hier Auszüge aus Paper (auf Web-Seite)
- Platzierer
  - Simulated Annealing-basiert
    - Mit adaptivem cooling schedule
  - Optimiert gleichzeitig
    - Leitungslänge
    - Verzögerung

VPR
Züge

TECHNISCHE UNIVERSITÄT DARMSTADT

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

VPR
Starttemperatur

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wird automatisch bestimmt
  passend für aktuelle Schaltung
- Idee:
  - Anfangs fast alle Züge akzeptieren
  - Wie hoch muß die Starttemperatur sein?
- Vorgehen:
  - N_{blocks} Blöcke paarweise austauschen
  - Beobachte Änderung der Kostenfunktion c (Standardabweichung)

$$s_c = \sqrt{\frac{1}{n-1}((\sum_i c_i^2) - n\bar{c}^2)}$$

  - Starttemperatur = $20 \cdot s_c$

VPR
Thermisches Gleichgewicht

TECHNISCHE UNIVERSITÄT DARMSTADT

- Anzahl von Schritten pro Temperaturstufe:

$$10 \cdot N_{blocks}^{\frac{4}{3}}$$

- $10\times$ schneller, aber nur ca. 10% schlechter:

$$N_{blocks}^{\frac{4}{3}}$$

VPR
Abkühlen

TECHNISCHE UNIVERSITÄT DARMSTADT

Beobachtung:

- Anfangs: T hoch, fast alle Züge akzeptiert
  - Im wesentlichen zufälliges Bewegen
  - Keine echte Verbesserung der Kosten
- Ende: T niedrig, kaum Züge akzeptiert
  - Fast keine Bewegung mehr
  - Wenig Veränderung in Kosten

Idee:

- Meiste Optimierung passiert zwischen Anfangs- und Endphase
- Bringe T schnell in den produktiven Bereich
- Halte T möglichst lange im produktiven Bereich

Vorgehen:

- Steuere T anhand der Akzeptanzrate R_a
  Akzeptanzrate R_a: Anteil der Züge die akzeptiert wurde (egal, ob verbesserend oder verschlechternd)

VPR
Abkühlen

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

VPR
Abkühlen

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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 Kosten
    - Kurze Züge: Kleine Änderung der Kosten
- Vorgehen:
  - Variiere Zugweite R_{limit}, um $R_a \approx 0.44$ zu halten

VPR
Abkühlen

TECHNISCHE UNIVERSITÄT DARMSTADT

R_{limit} klein

- Kleine Zugreichweite
- Kleine Änderungen der Kosten
- Kleine Verschlechterungen
  (Werden eher angenommen)
- R_a steigt

R_{limit} groß

- Große Zugreichweite
- Große Änderungen der Kosten
- Große Verschlechterungen
  (Werden eher abgelehnt)
- R_a sinkt

VPR
Abkühlen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Anfangs:

$$R_{limit} = \text{ ganzer Chip } L_{Chip}$$

- Bei jedem Abkühlschritt:

$$R_{limit}^{new} = R_{limit}^{old}(1 + R_a^{old} - 0.44) \qquad \text{mit } 1 \leq R_{limit}^{new} \leq L_{Chip}$$

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

VPR
Abbruchbedingung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Wann Abkühlung beenden?
- Idee:
  - Stillstand erkennen
- Vorgehen:
  - Jeder Zug beeinflußt mindestens ein Netz
  - Bestimme die durchschnittlichen Kosten pro Netz
  - Wenn T kleiner als ein Bruchteil davon ...
    - Nur noch kleine Chance, dass Zug akzeptiert wird
    - $T < 0.005 \cdot \mathit{AvgCostPerNet}$
  - Auch einfachere Realisierung möglich
    - Letzte k Züge ohne akzeptierten Zug
    - Letzte k Züge ohne Verbesserung von BSF
    - ...

VPR
Kostenfunktion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gleichzeitiges optimieren von
  1. Verdrahtungslänge
  2. Zeitverhalten

$\Rightarrow$ Kombination von 2 Kostenfunktionen

1. Korrigierter HPWL: $c_w = \sum_{n \in N} q(n_{pincount}) HPWL(n)$
   Korrekturfaktor q_n um Unterschätzung vorzubeugen
   ($q(1) = 1, \ldots, q(50) = 2.79$, für Details siehe Paper auf Web-Seite [Cheng 1994])
2. Zeitverhaltensabschätzung c_t

VPR
Optimierung HPWL

TECHNISCHE UNIVERSITÄT DARMSTADT

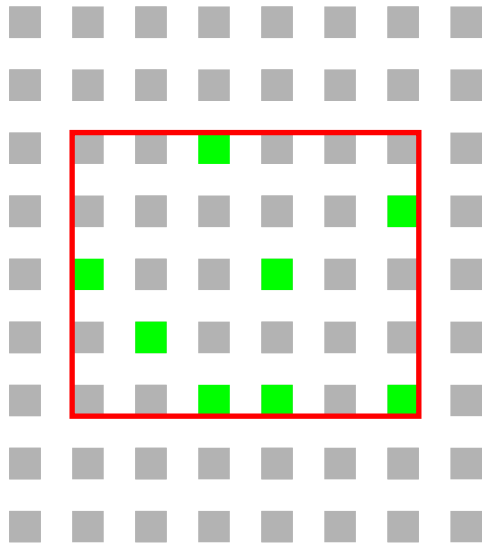
- Berechnung HPWL
  - Simpel: $\mathcal{O}(k)$, k Anzahl der Pins
  - Problem: $k = 100 \ldots 1000$ realistisch
  - Nach jedem Zug neu berechnen
- Besser:
  - Nach Möglichkeit nur bewegte Pins neu berechnen
    - Ein Pin ist nur in einem Netz
    - Ein Block hat aber mehrere Pins
  - Vorgehen:
    -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

VPR
Optimierung HPWL

TECHNISCHE UNIVERSITÄT DARMSTADT

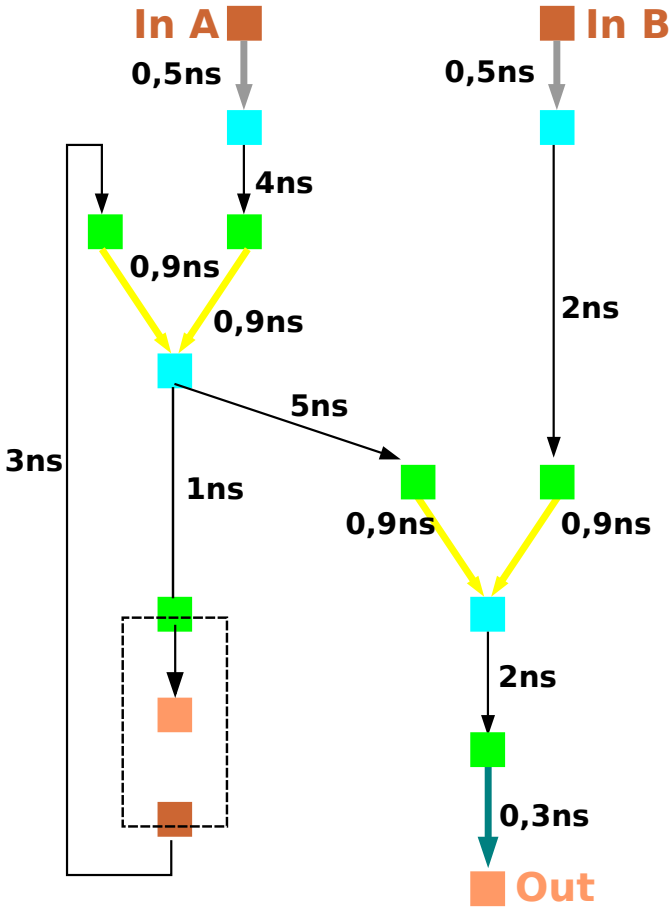
- Als Beispiel nur linke Seite
- Bewege Terminal von x_{old} nach x_{new}
- Netz an Terminal: n

if $x_{new} \neq x_{old}$ **then**
  if $x_{new} < n.xmin$ **then**
    n.xmin := x_{new} ;
    n.Nxmin := 1
  else if $x_{new} = n.xmin$ **then**
    n.Nxmin++
  else if $x_{old} = n.xmin$ **then**
    if *(n.Nxmin > 1)* **then**
      n.Nxmin–
    else BruteForce(n) ;



$(x_{min}, x_{max}, y_{min}, y_{max},) = (2, 7, 3, 7)$
$(N_{xmin}, N_{xmax}, N_{ymin}, N_{ymax},) = (1, 2, 3, 1)$

VPR
Kostenfunktion Zeitverhalten



- Betrachte Platzierungsabhängiges Zeitverhalten
- Punkt-zu-Punkt Verbindungen
- Von Netzquelle *u*
- Zu jeder Netzsenke *v*
- Sicht: *Two-Terminal-Nets*
- Zeitverhalten:
  - Bestimmt aus Slacks
  - Nicht auf Pfaden (langsam)

VPR
Kostenfunktion Zeitverhalten

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

  - (u, v) auf kritischem Pfad:
    $slack(u, v) = 0 \Leftrightarrow Criticality(u, v) = 1$
  - (u, v) absolut unkritisch:
    $slack(u, v) = D_{max} \Leftrightarrow Criticality(u, v) = 0$
- Timing Cost: $Delay(u, v)$ ist Schätzung!
  Noch kein *echtes* Routing!

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

VPR
Kostenfunktion Zeitverhalten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Criticality Exponent *ce*
  - Gewichtet kritischere Verbindungen höher
  - Untergeweichtet unkritischere Verbindungen
- Idee:
  - Gegen Ende auf kritische Netze konzentrieren
- Vorgehen:
  - Steigern von $ce_{start} = 1$ auf $ce_{final} = 8$ (experimentell)

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

VPR
Kostenfunktion Zeitverhalten

TECHNISCHE UNIVERSITÄT DARMSTADT

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

VPR
Gesamtkostenfunktion

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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- gegenüber Längenoptimierung
  - Aber $\lambda = 1$ erzeugt nicht die schnellste Lösung
  - Netze wechselnd kritisch/unkritisch
    Nicht erkannt, da Timing-Analyse nur $1\times$ pro Temperaturstufe
  - Besser $\lambda = 0.5$
    Längenmaß wirkt als Dämpfer für Oszillation

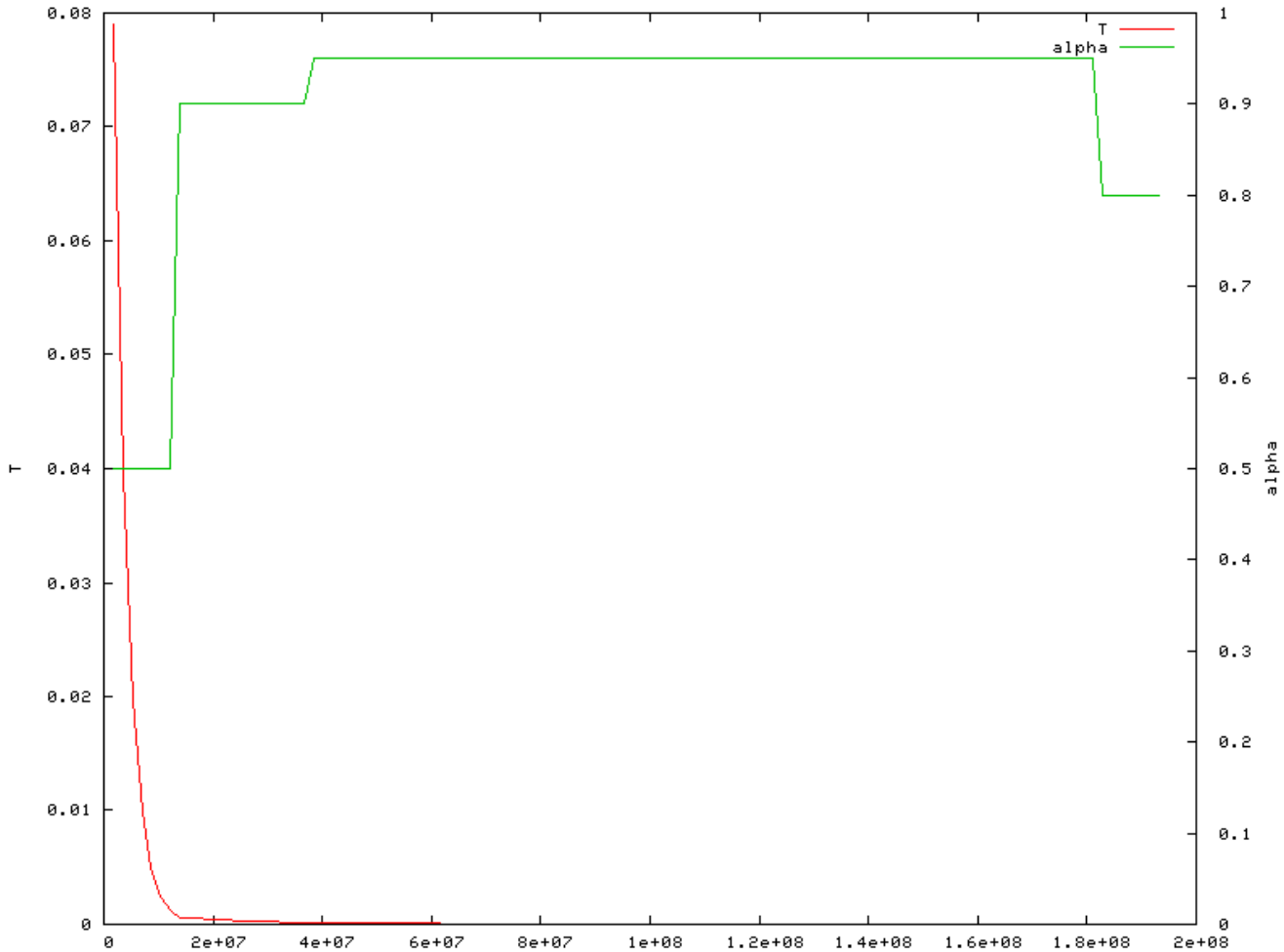
VPR
Gesamtalgorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
S := RandomPlacement() ;
T := InitialTemperature() ;
Rlimit := InitialRlimit() ;
CritExp := ComputeNewExponent(Rlimit);
repeat
    /* Bestimme Ta, Tr und slack()          */
    TimingAnalyze() ;
    /* für Normalisierung der Kostenterme
       */
    OldWiringCost := WiringCost(S) ;
    OldTimingCost := TimingCost(S) ;
    while !InnerLoopCriterion() do
        .
        .
        .
        /* eine Temperaturstufe             */
        .
        .
        .
    T := UpdateTemp() ;
    Rlimit := UpdateRlimit() ;
    CritExp := ComputeNewExponent(Rlimit) ;
until ExitCriterion();
```

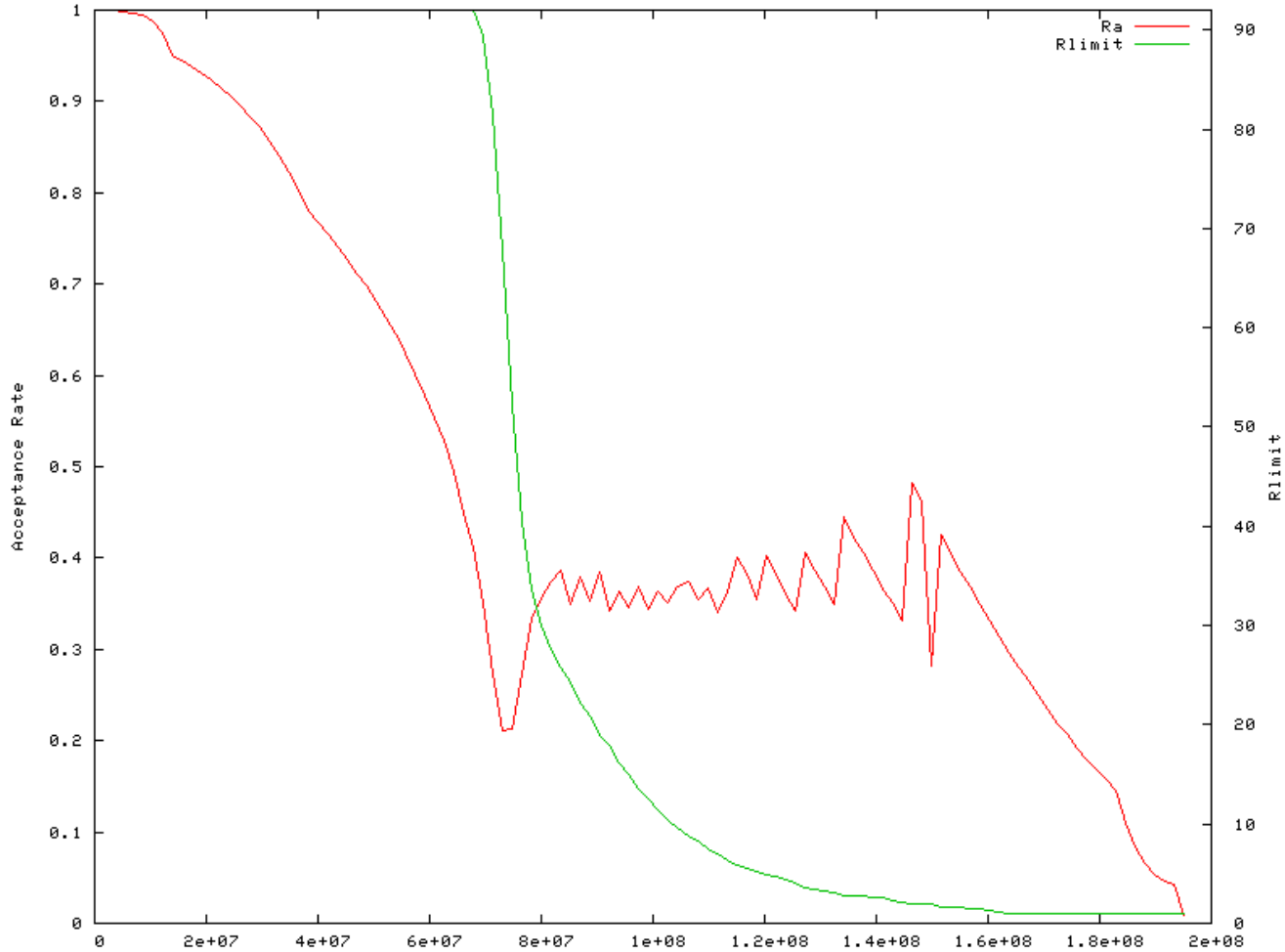
```
.
.
.
/* eine Temperaturstufe:                    */
Snew := GenerateSwap(S, Rlimit) ;
ΔtimingCost := TimingCost(Snew) – TimingCost(S) ;
ΔwiringCost := WiringCost(Snew) – WiringCost(S) ;
ΔC := λ (ΔtimingCost/OldTimingCost) +
  (1-λ)(ΔwiringCost/OldWiringCost);
if ΔC ≤ 0 then
    S = Snew
else
    if random(0,1) < exp(-ΔC/T) then
        S = Snew

.
.
.
```

VPR
Temperatur- und α-Graph

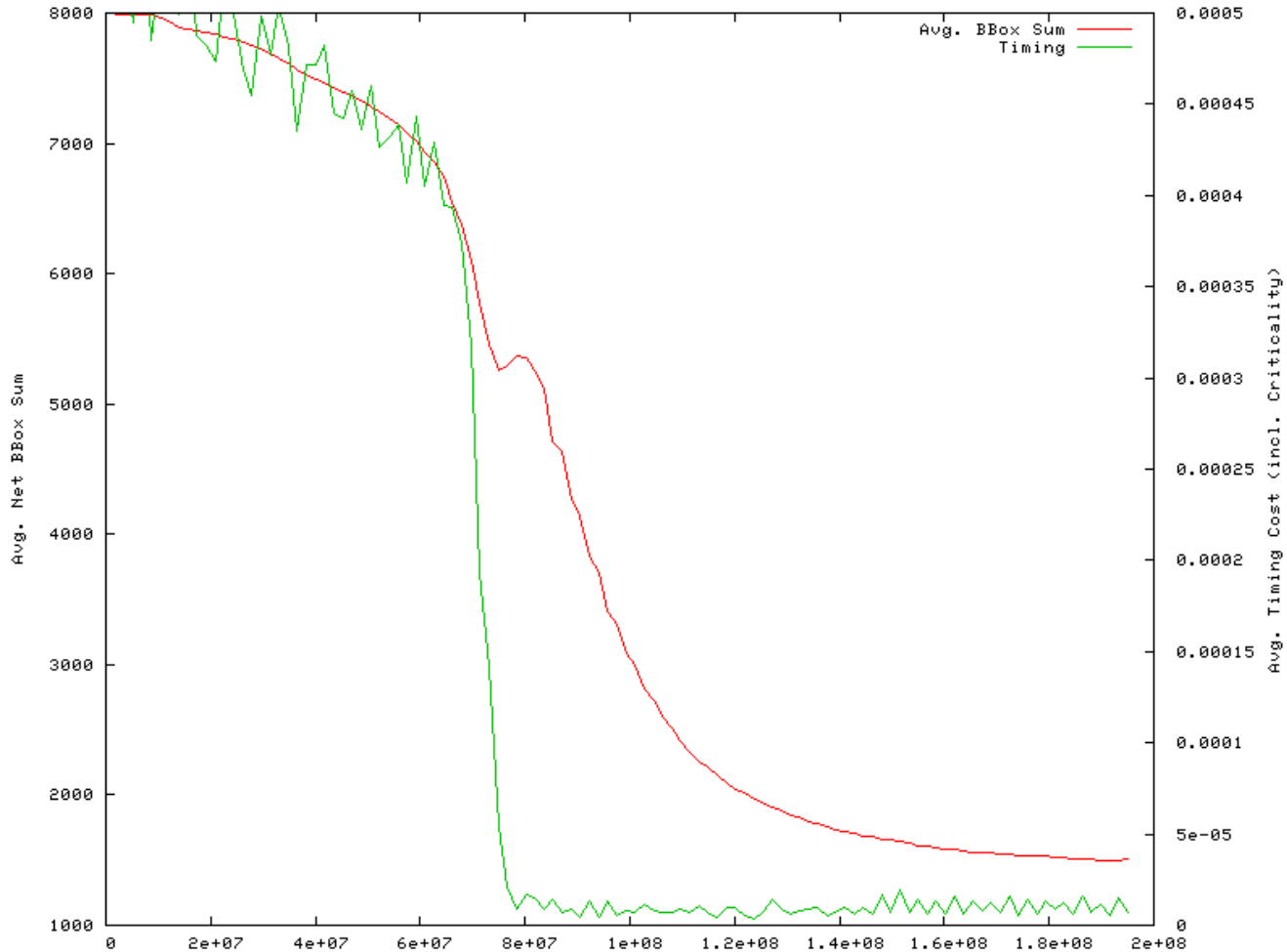


VPR
R_a- und R_{limit}-Graph

TECHNISCHE UNIVERSITÄT DARMSTADT



VPR
Kosten-Graphen



DAST
STUN

TECHNISCHE UNIVERSITÄT DARMSTADT

- Weitere SA Verbesserung
- Dynamisch Adaptives STUN $\Rightarrow$ DAST
  (Lin & Warzynek 2010)
- Idee: Verbessertes entkommen lokaler Minima
- Stochastisches Tunneln $\Rightarrow$ STUN
  (Hamacher 1999)
- Zusätzlich:
  - Multimodale Bewegungen
  - Lokale Minima Detektion $\Rightarrow$ Nur dann STUN benutzen

STUN
Stochastic Tunneling

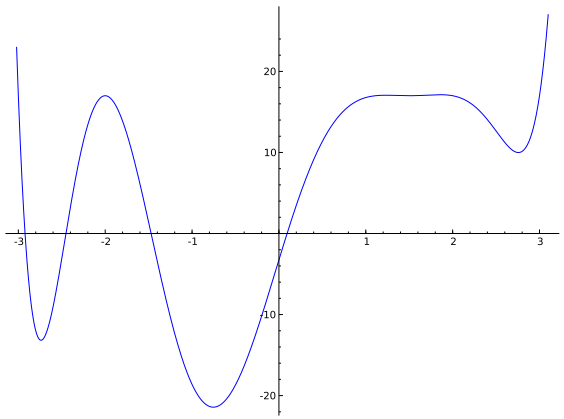
TECHNISCHE UNIVERSITÄT DARMSTADT

- SA kann immer noch in lokalen Minima stecken bleiben (Freezing Problem)
- Idee: Tunneln durch (lokales) Maximum
- Kostenfunktion wird modifiziert: $c'(x) = g(x) \circ c(x)$
  - *Große* Werte werden geglättet
  - *Kleine* werden hervorgehoben
  - *Klein* und *groß* relativ zu der bisher besten Lösung mit Kosten c_{BSF}
  - Mehere Funktionen möglich (γ Tunnel Parameter der die Stärke des Glättens steuert): $1 - e^{-\gamma(c-c_{BSF})}$, $\frac{1-sgn(c-c_{BSF})}{2}c$, $\tanh(-\gamma(c-c_{BSF}))$, $\sinh(-\gamma(c-c_{BSF}))$, ...
  - Empirisch festgestellt:
    - Funktion beeinflußt Ergebnis bis zu 20%
    - Für das Platzierungsproblem am besten: $1 - e^{-\gamma(c-c_{BSF})}$
    - $\gamma \in [0, ..., 5]$ beeinflußt Ergebnis bis zu 30%
      Wird manuell angepaßt
- Veränderung der Kostenfunktion

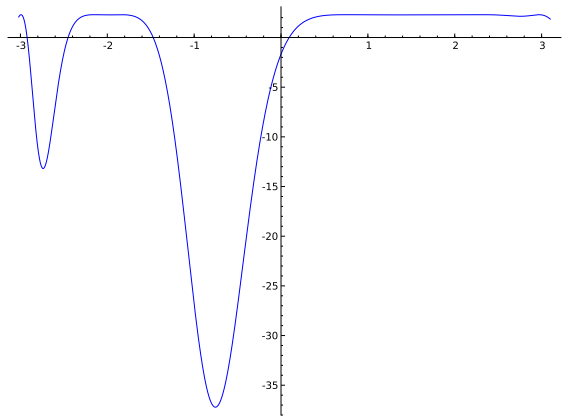
STUN
Glättungsfunktion

TECHNISCHE UNIVERSITÄT DARMSTADT

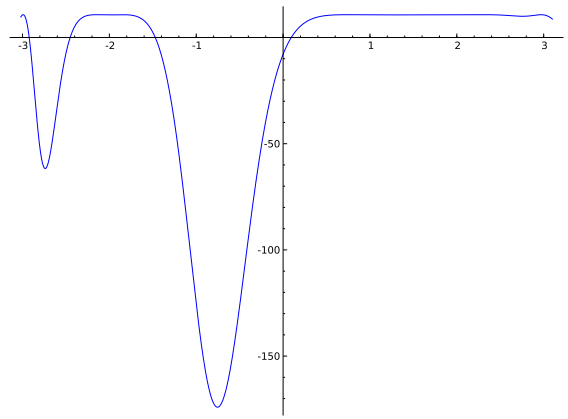
$$c_{STUN}(x) = 1 - e^{-\gamma(c(x)-c_{BSF})}$$



1-D Kostenfunktion



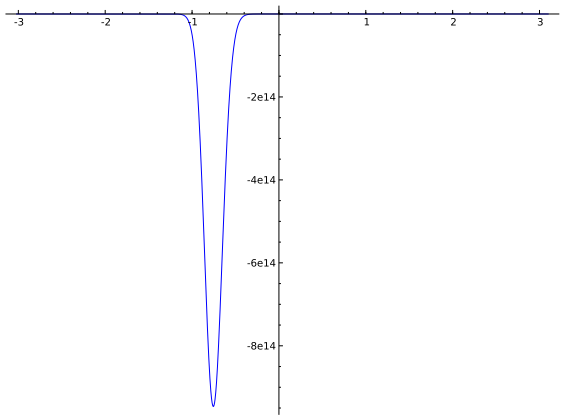
Kostenfunktion mit Glättung, BSF bei $x_{BSF} = 2.76$



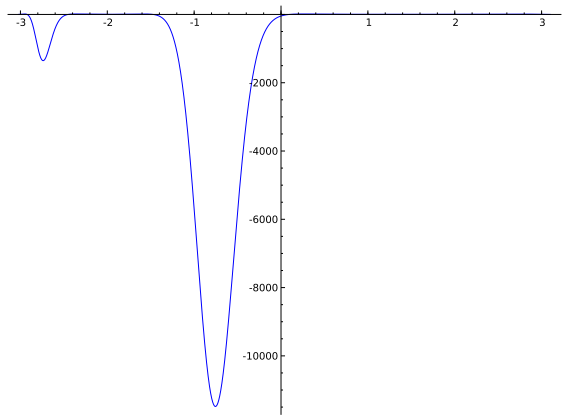
Kostenfunktion mit Glättung, BSF bei $x_{BSF} = -2.73$

Glättungsfunktion
Einfluß γ

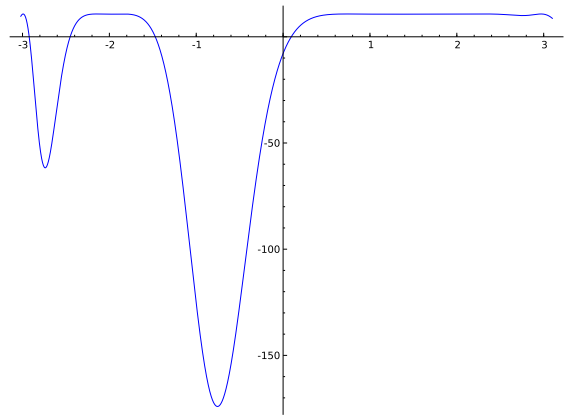
TECHNISCHE UNIVERSITÄT DARMSTADT



$\gamma = 1$



$\gamma = 5$



$\gamma = 15$

In DAST: Adapativ, so dass $\frac{c - c_{BSF}}{\gamma} \approx 0.05$

DAST
Multimodale Bewegungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verallgemeinerung von VPRs R_{limit}
- Es gibt verschiedene Arten von Zügen
  (z.B. Züge mit maximalen Reichweiten: $\frac{2}{3}L, \frac{4}{3}L, 2L$)
- Akzeptanzrate a_i für jede Zugart m_i mitprotokollieren
- Gibbs-Sampling (spez. Metropolis-Hastings-Algorithmus)
  - Anfänglich werden Züge mit gleicher Wahrscheinlichkeit gewählt
  - Später mit Wahrscheinlichkeit $p(m_i) = \frac{a_i}{\sum_{i=0}^{n} a_i}$

Für Details: Siehe Paper

DAST
Lokale Minima Erkennung

TECHNISCHE UNIVERSITÄT DARMSTADT

- STUN besonders gut wenn in lokalem Minimum
- Permanentes stochastisches Tunneln negativ:
  - Golfkurs-Effekt (Ebene mit einem Loch)
  - Lokale Züge sind weniger effektiv

$\Rightarrow$ Versuche Minima zu erkennen
und nur wenn nötig STUN zu benutzen

- DAST: Nutzt jede 10000. Iteration DFA (Detrended Fluctuation Analysis)
  - Für Details: Siehe Paper
  - Alternativen möglich: z.B. Ableitung

DAST
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Stochastisches Tunneln
- Multimodale Züge
- Minima Erkennung
- Vergleich mit VPR5: Verbesserung von ...

Laufzeit:	$\approx$ 30%
Verzögerung:	(kritischer Pfad) $\approx$ 12%
Verdrahtbarkeit:	(min. Tracks) 3%

SPA
Structured Parallel Annealer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Zug basiert immer auf (Nicht-)Akzeptanz des Vorhergenden

$\Rightarrow$ SA sehr sequentiell, schlecht parallelisierbar

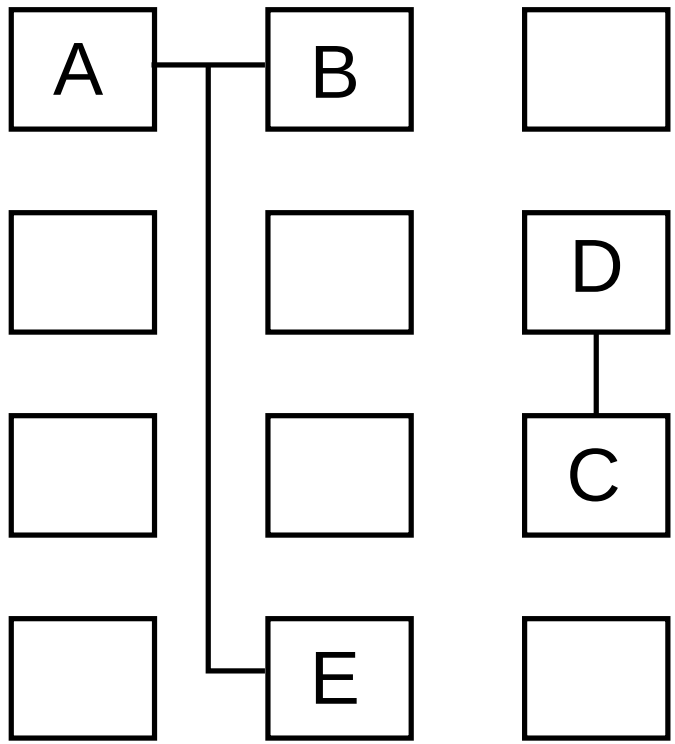
- Verbessertes SA: SPA von Fobel, et al., 2014 FPL
- Nutzt Star+ Metrik
- Geschickte Wahl der Züge erlaubt Parallelisierung

Paralleles SA
Konflikte

TECHNISCHE UNIVERSITÄT DARMSTADT

- Angenommen 2 Züge parallel:
  - Thread Blau
  - Thread Rot
- Harter Konflikt:
  - Beide wählen zufällig gleichen Block für Zug aus
  - Kritisch, Folgezustand wäre fehlerhaft
- SPA vermeidet Harte Konflikte

Harter Konflikt

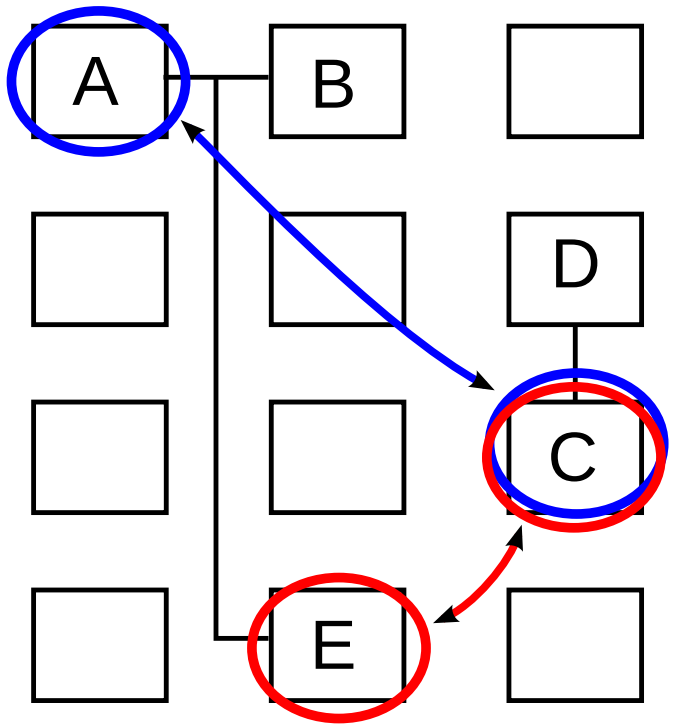


Paralleles SA
Konflikte

TECHNISCHE UNIVERSITÄT DARMSTADT

- Angenommen 2 Züge parallel:
  - Thread Blau
  - Thread Rot
- Harter Konflikt:
  - Beide wählen zufällig gleichen Block für Zug aus
  - Kritisch, Folgezustand wäre fehlerhaft
- SPA vermeidet Harte Konflikte

Harter Konflikt

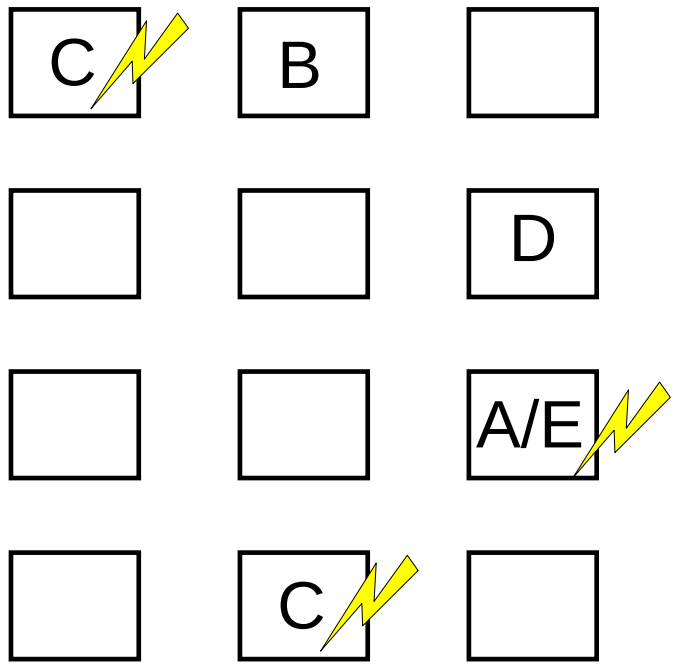


Paralleles SA
Konflikte

TECHNISCHE UNIVERSITÄT DARMSTADT

- Angenommen 2 Züge parallel:
  - Thread Blau
  - Thread Rot
- Harter Konflikt:
  - Beide wählen zufällig gleichen Block für Zug aus
  - Kritisch, Folgezustand wäre fehlerhaft
- SPA vermeidet Harte Konflikte

Harter Konflikt

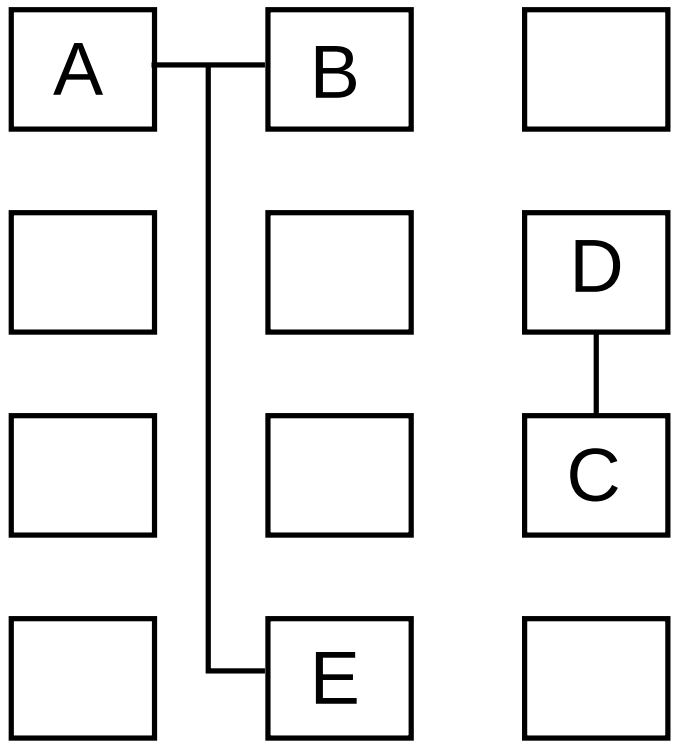


Paralleles SA
Konflikte

TECHNISCHE UNIVERSITÄT DARMSTADT

- Angenommen 2 Züge parallel:
  - Thread Blau
  - Thread Rot
- Sanfter/Weicher Konflikt:
  - Beide wählen zufälligen Block aus, der an gleichem Netz hängt
  - Unkritisch, Nur geschätzte Kosten wären leicht fehlerhaft
- Zulassen von Weichen keine negativen Auswirkungen

Weicher Konflikt

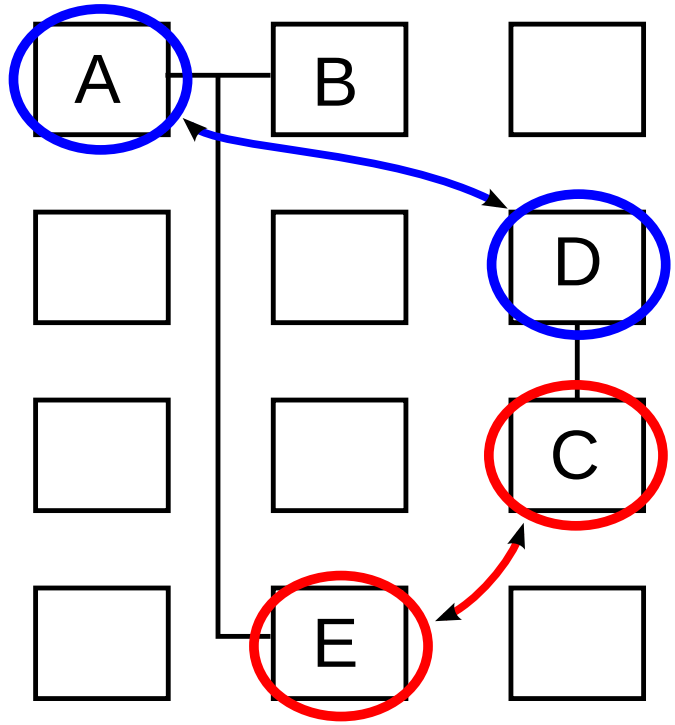


Paralleles SA
Konflikte

TECHNISCHE UNIVERSITÄT DARMSTADT

- Angenommen 2 Züge parallel:
  - Thread Blau
  - Thread Rot
- Sanfter/Weicher Konflikt:
  - Beide wählen zufälligen Block aus, der an gleichem Netz hängt
  - Unkritisch, Nur geschätzte Kosten wären leicht fehlerhaft
- Zulassen von Weichen keine negativen Auswirkungen

Weicher Konflikt

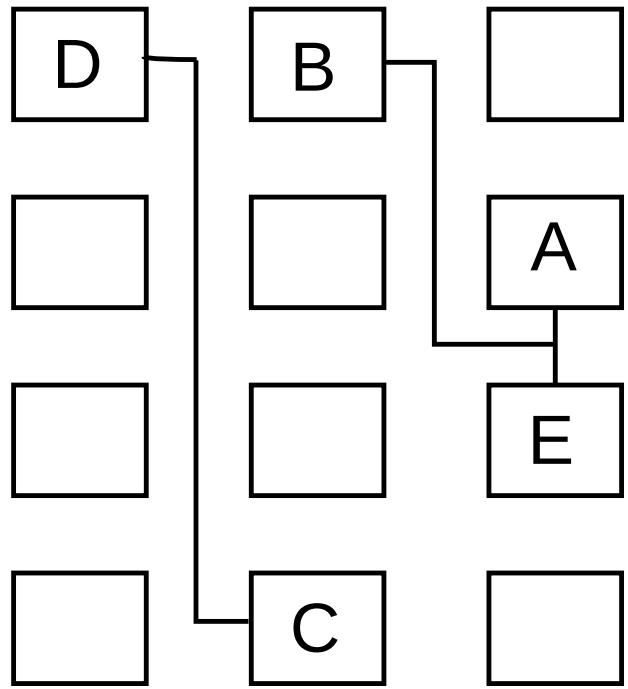


Paralleles SA
Konflikte

TECHNISCHE UNIVERSITÄT DARMSTADT

- Angenommen 2 Züge parallel:
  - Thread Blau
  - Thread Rot
- Sanfter/Weicher Konflikt:
  - Beide wählen zufälligen Block aus, der an gleichem Netz hängt
  - Unkritisch, Nur geschätzte Kosten wären leicht fehlerhaft
- Zulassen von Weichen keine negativen Auswirkungen

Weicher Konflikt

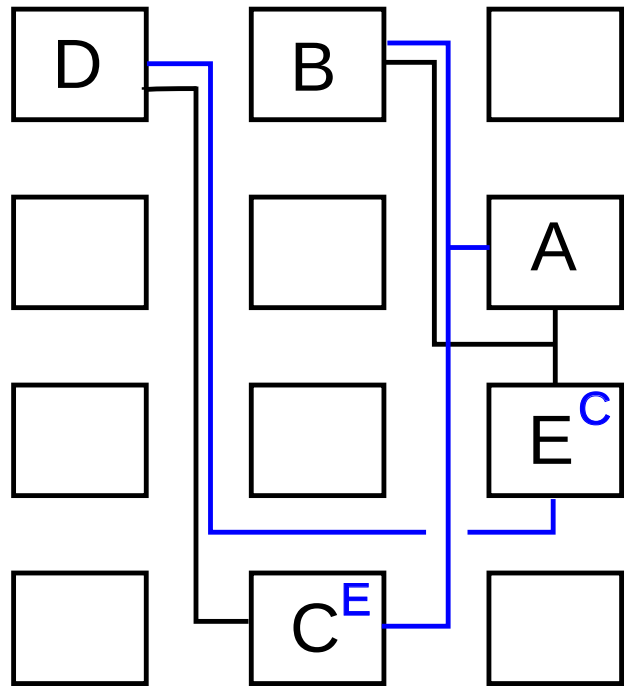


Paralleles SA
Konflikte

TECHNISCHE UNIVERSITÄT DARMSTADT

- Angenommen 2 Züge parallel:
  - Thread Blau
  - Thread Rot
- Sanfter/Weicher Konflikt:
  - Beide wählen zufälligen Block aus, der an gleichem Netz hängt
  - Unkritisch, Nur geschätzte Kosten wären leicht fehlerhaft
- Zulassen von Weichen keine negativen Auswirkungen

Weicher Konflikt

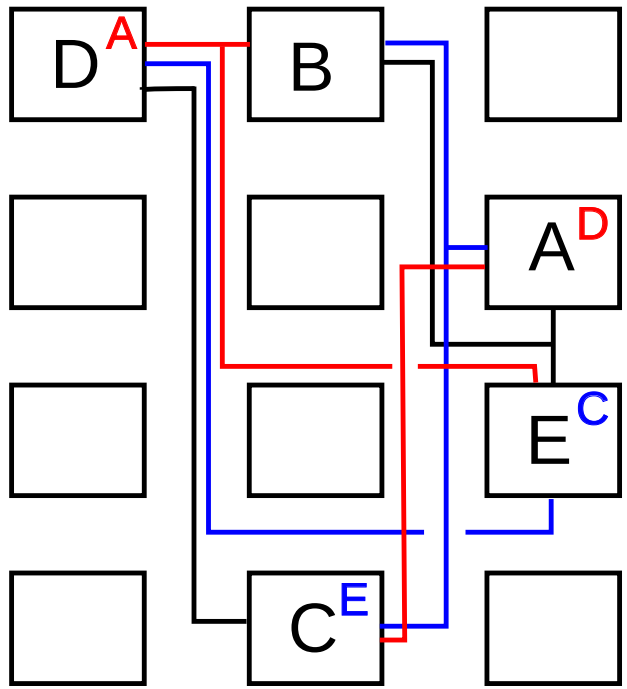


Paralleles SA
Konflikte

TECHNISCHE UNIVERSITÄT DARMSTADT

- Angenommen 2 Züge parallel:
  - Thread Blau
  - Thread Rot
- Sanfter/Weicher Konflikt:
  - Beide wählen zufälligen Block aus, der an gleichem Netz hängt
  - Unkritisch, Nur geschätzte Kosten wären leicht fehlerhaft
- Zulassen von Weichen keine negativen Auswirkungen

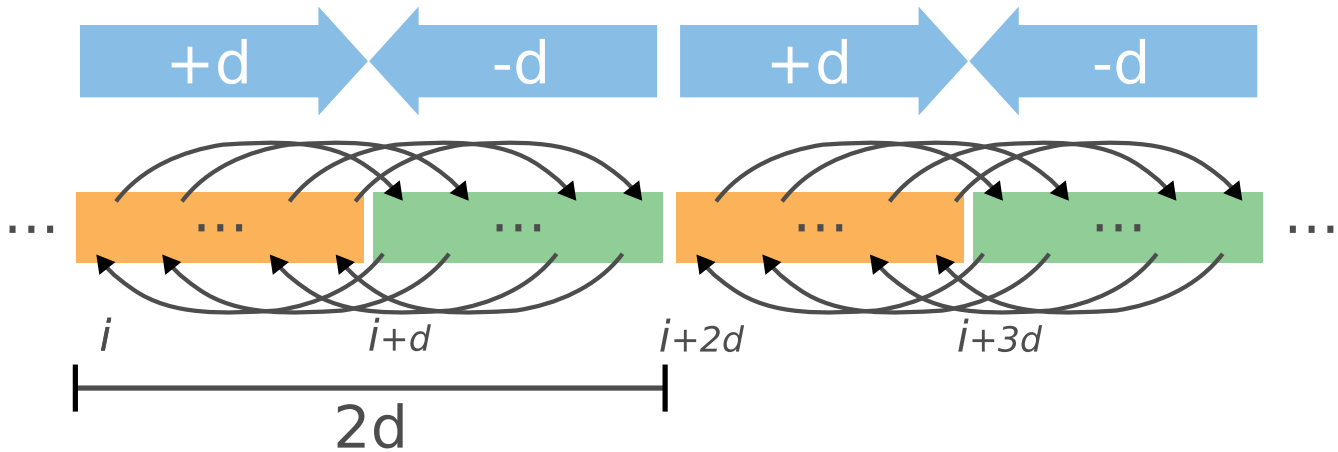
Weicher Konflikt



SPA
Konfliktvermeidung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Idee:
  Alle Threads/Blöcke machen den *gleichen* Tausch
- Generierung von Mengen von Tauschpaaren,
- Zum Tausch über Distanz *d*:



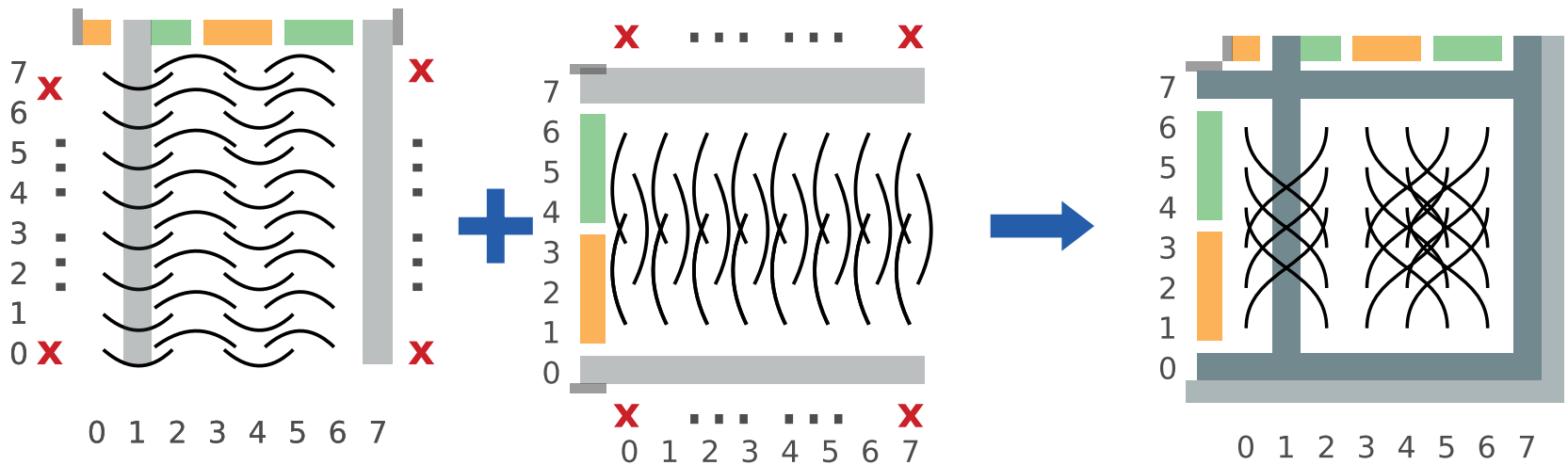
[Quelle: Fobel et al.]

- Garantiert, dass nie 2 verschiedene Blöcke auf die gleiche Position getauscht werden

SPA
Konfliktvermeidung 2-Dimensional

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verallgemeinerung auf 2-D:
  Jeweils für jede Dimension und zusammenführen



[Quelle: Fobel et al.]

SPA
Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Übernimmt alles von VPR (inkl. Temperatur, Coolding-Schedule und r_{lim})
- Statt S_t sequentiell auf einer Temperaturstufe

$\Rightarrow$ parallele Züge bis S_t erreicht

- Ablauf einer parallelen Iteration:
  1. Parallel: Kosten aller Kanten (Knoten-Netz-Paare)
  2. Parallel: Züge bestimmen
  3. Parallel: Delta-Kosten für alle Züge
     (Nur von Tauschpartner mit größerem Index)
  4. Parallel: (Nicht) aktzeptieren
  5. Ergebnis verteilen

SPA
Ergebnisse

TECHNISCHE UNIVERSITÄT DARMSTADT

- Skaliert hervorragend
- Funktioniert mit einfachen Parallelprogrammierungs-Direktiven
  $\Rightarrow$ Sehr einfach umzusetzen (GPGPU/SSE/...)
- Gute Lösungsqualität (5% besser als VPR (HPWL) im fast-Modus)
- Durchschnittlich 19$\times$ schneller als VPR (fast)
- Für mehr Details: Siehe Paper

Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Allgemeines Platzierungsproblem
- Design-Stile & Zielarchitekturen
  - Standardzellen
  - Makro-/Buildingblock
  - FPGA/MPGA
  - Mixed-Mode
- VPR
  - Adaptives Simulated Annealing
  - Kostenfunktion (timingbasiert, selbstnormalisierend)
  - Gesamtalgorithmus
- DAST
  - STUN
- SPA
  - Paralleles SA, durch geschickte Zug-Generierung

Partionierungsbasierte Platzierer
Idee

TECHNISCHE UNIVERSITÄT DARMSTADT

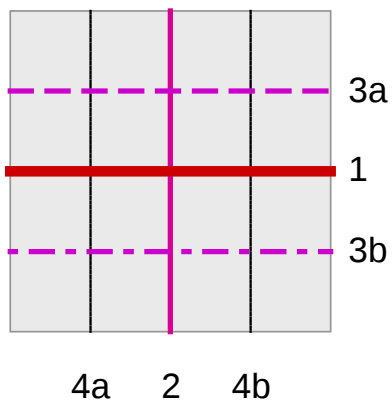
- Ansatz:
  1. Verfügbarer Platz wird halbiert
  2. Schaltung wird partitioniert (mittels FM, KL, . . .)
  3. Halbe Schaltung wird auf jeweils die hälfte platziert
  4. Rekursiv bis auf das kleinste Element runter
- Anzahl Netze über die Schnittlinien wird minimiert, darum auch Min-Cut-Platzierung
- Problematisch: Partitionsübergreifende Verbindungen
  $\Rightarrow$ Terminal propagation (Dunlop, 1995):
  Fixierte Dummy Terminals werden eingefügt

Partionierungsbasierte Platzierer
Rekursion

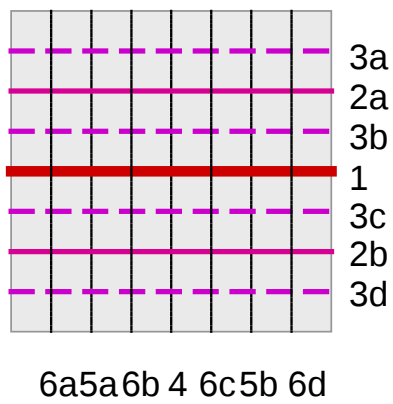
TECHNISCHE UNIVERSITÄT DARMSTADT

- Verschiedene Ansätze für die Rekursion möglich:

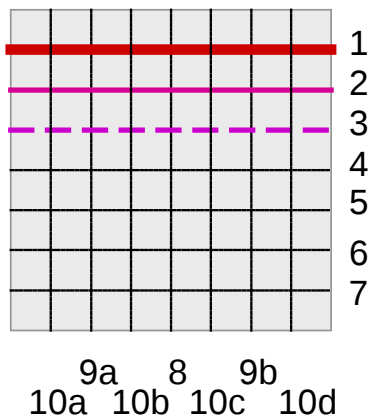
Quadratur-Platzierung
(Quadrature placement)



Halbierungs-Platzierung
(Bisection placement)



Reihen-/ Halbierungs-Platzierung
(Slice/bisection placement)



Nach Sait, S. M., Youssef, H.: VLSI Physical Design Automation

Partionierungsbasierte Platzierer
Übersicht

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sehr schnell
- Hierarchisch $\Rightarrow$ sehr gut für große Schaltungen und parallelisierbar
- Unterhalb bestimmter Größen wird oft herkömmlicher Platzierer verwendet
- Optimierung immer nur bzgl. einer Schnittlinie
- Waren in 90ern nicht annähernd so gut wie SA oder analytische Platzierer
- Stark verbessert mit den Multilevel Partitionierern
  2 Akademische Platzierer: Capo (Caldwell, 2000), Fengshui (Yildiz, 2001)

Kräftebasierte Placer

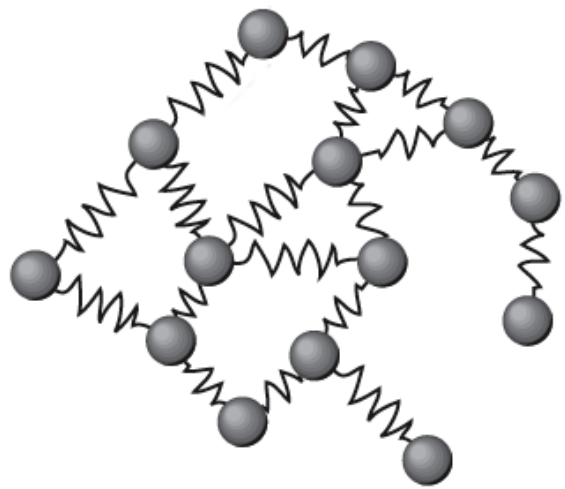
TECHNISCHE UNIVERSITÄT DARMSTADT

- Zielkostenfunktion: Quadratische Verdrahtungslänge

  Vorteil: Im Gegensatz zu HPWL stetig differenzierbar

  Nachteil: Lange Netze gegenüber kurzer übergewichtet

- Entspricht physikalisch einer Anordnung von Objekten die mittels Federn verbunden sind
  $\Rightarrow$ Kräftebasierte (Force-Driven) Platzierung
- Optimum entspricht Kräftegleichgewicht (Überlappung erstmal ignoriert)
- Optimum finden durch lösen eines LGS
  $\Rightarrow$ sehr einfach $\Rightarrow$ sehr beliebt
- Für zusätzliche Ziele zusätzliche Kräfte möglich



[Quelle: Wikimedia]

Kräftebasierte Placer

TECHNISCHE UNIVERSITÄT DARMSTADT

- Überlappungsfreiheit herstellen
  - Erst ignorieren, hinterher legaliserien
    (z.B. Tetris Legalisierung)
  - Zustäzliche Kräfte die für Verteilung wirken hinzufügen
    (Schwerpunkte oder andere explizite Kräfte)
    Iterativ wiederholen bis keine Überlappung
- Implementierungen Kräftebasierter Platzierer:
  - GORDIAN (Kleinhans, 1991)
  - BonnPlace (Brenner, 2005)
  - hATP (Nam, 2006)

Analytische Placer
StarPlace

TECHNISCHE UNIVERSITÄT DARMSTADT

- StarPlace(Xu, Grewal, Areibi 2010)
- Star+-Netzmodel
- Benutzt CG- oder SOR-(Successive Over-Relaxation)-Löser
- Zur Vermeidung von Triviallösungen:
  ShrubPlace (vorplatziert alle I/O-Logik am Außenrand)

StarPlace
Star+ Performance

TECHNISCHE UNIVERSITÄT DARMSTADT

- In VPR: Star+ vs. HPWL
  - Benutzt mit empirisch ermittelten Werten

    $\gamma = 1.59$
    $\phi = 1.0$
  - Ergebnisse

Verdrahtbarkeit:	Ähnlich gut
Verdrahtungsressourcen:	Ähnlich gut
	(2.4% besser als VPR-`fast`)
Kritischer Pfad:	$6\% - 9\%$
Laufzeit:	Kaum ein Unterschied

StarPlace
Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

```
APlace()
begin
    Convert_Graph_Star+() ;
    PrePlace() ;
    repeat
        CG_Iteration() ;
        Legalize() ;
    until Max Anzahl Iterationen;
```

1. Graph mittels Star+ konvertieren
2. Vorplatzieren mit shrubPlace (I/O am Rand, Blöcke mit viel I/O nahe beieinander)
3. Konjugiertes Gradientenverfahren
4. In jeder Iteration Lösung legalisieren (Partitionsbasierter Ansatz)

StarPlace
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Benutzt zur Vorplatzierung shrubPlace $\Rightarrow$
  - 1.5% besseres Ergebnis bei Vedrahtungsressourcen gegenüber zufälliger Vorplatzierung
  - Benötigt 5% der Laufzeit von StarPlace
- Benutzt CG-Löser
  2% besserer kritischer Pfad, 4% mehr Verdrahtungsressourcen bei 56% längerer Laufzeit
- Besserer Löser: SOR (Successive Over-Relaxation)
  (CG *schwächelt* bei nichtquadratischen Funktionen)
  9% besserer kritischer Pfad bei 1% mehr Verdrahtungsressourcen bei 78% weniger Laufzeit

Analytische Placer
APlace

TECHNISCHE UNIVERSITÄT DARMSTADT

- Khang & Wang, 2005
- Zufällige Startplatzierung
- LSE-Zielkosten
- Zusätzliche Kosten/Kräfte zur Vermeidung von Überlappung
- Top-Down hierarchisches Vorgehen
- CG-Löser

APlace
Kostenfunktion

TECHNISCHE UNIVERSITÄT DARMSTADT

- LSE-Zielkosten
- Verteilung der Zellen (Reduzierung der Überlappung):
  Aufteilung der vorhandenen Fläche in Gitter g

$$DichteStrafe := \sum_{\text{Gitterelemente } g} (\text{GesamtZellFläche}(g) - \text{DurchschnittZellenFläche})^2$$

- Ziel: Möglichst gleichmäßige Verteilung aller Zellen über gesamte Fläche
- Nicht differenzierbar, darum Potenialfunktion

$$Potenial(c, g) := K_c \cdot p(Zelle_x - Gitter_x) \cdot p(Zelle_y - Gitter_y)$$

K_c Normierungsfaktor, so dass
$\sum_g Potenial(c, g) = A_c$ mit A_c Fläche der Zelle c

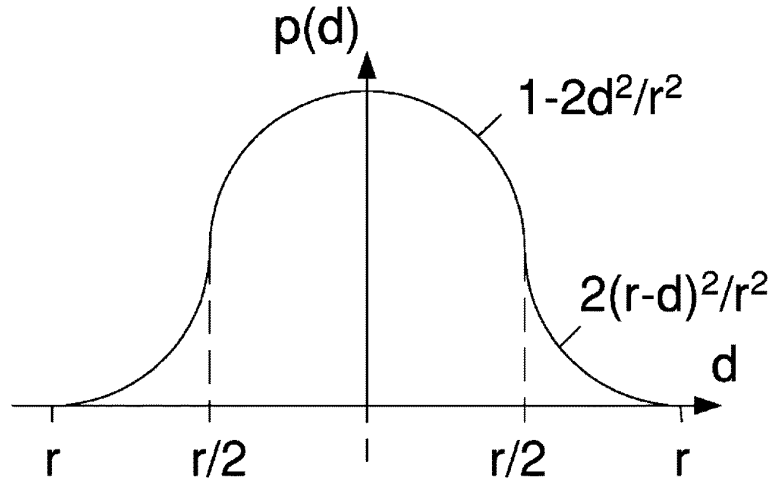
APlace
Glockenfunktion p

TECHNISCHE UNIVERSITÄT DARMSTADT

- Naylor et al., 2001
- Glockenfunktion

$$p(d) = \begin{cases} 1 - 2d^2/r^2 & 0 \leq d \leq r/2 \\ 2(d-r)^2/r^2 & r/2 \leq d \leq r \\ 0 & \text{sonst} \end{cases}$$

- r ist der Einflußradius (empirisch $r = 4$)



[Quelle: Kahng]

APlace
Kostenfunktion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Differenzierbare Straffunktion:

$$DichteStrafe := \sum_{\text{Gitterelemente } g} \left(\left(\sum_{\text{Zellen c}} Potenial(c, g) \right) - ExpPotenial(g) \right)^2$$

- Erwartetes Potenial von Gitterelement g:

$$ExpPotenial(g) := \frac{\text{GesamtZellFläche}}{\text{AnzahlGitterelemente}}$$

- Kombination von Verdrahtungslänge und Dichte:

$$Gewicht_{WL} \cdot c_{LSE} + Gewicht_{Dichte} \cdot DichteStrafe$$

APlace
Zusammenfassung

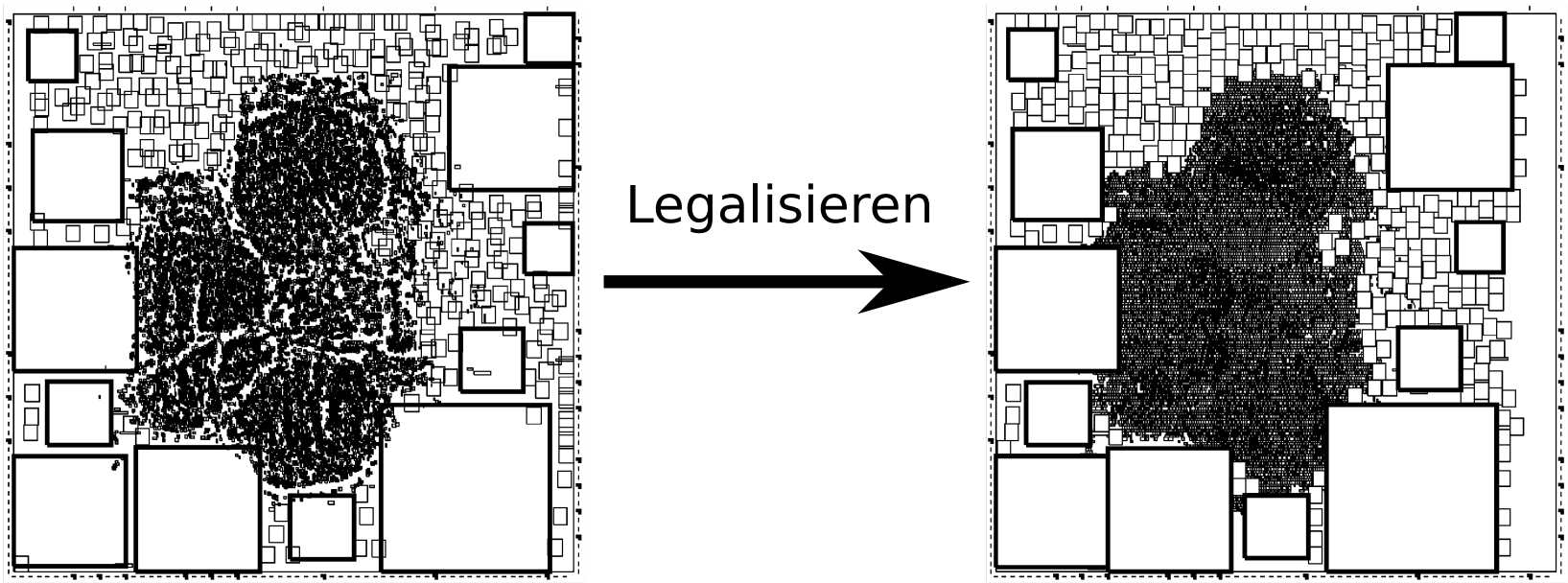
TECHNISCHE UNIVERSITÄT DARMSTADT

- LSE-Kosten
- Dichtefunktion zur Überlappungsreduzierung
- Ergebnisse bis zu 10% besser als bei anderen Placern
- Laufzeit −5% bis +1300% höher
- Varianten:
  - Auch andere Verdrahtungslängenfunktionen
  - Timing-Driven
  - Congestion-Driven
  - Mixed-Size-Design

Legalisierung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Legalisierung von Platzierungen mit Überlappungen
- Weit verbreitet bei analytischen Platzierern
  - Erst Problem ohne Nebenbedingungen optimieren
  - Hinterher gefundene Lösung legalisieren



Legalisierung
Tetris-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hill, 2002 für Standardzellen, verwendet z.B. bei APlace
- Greedy-Algorithmus:
  1. Module in absteigender x-Koordinate sortieren (Breitere bei unentschieden) $\rightarrow$ Abarbeitungsreihenfolge
  2. Für jedes Modul: Mögliche Kandidatenposition, ist die am weitesten links liegende Position in jeder Reihe $\Rightarrow$ die mit kleinstem Kosten (d.h. Abstand) wird Zielposition
- Sehr schnell
- Funktioniert auch gut bei Mixed-Size-Designs
- Varianten:
  - Erleichterte vertikale Bewegung $\Rightarrow$ Ausgeglichenere Reihen
  - Diffusionsbasierter Ansatz
    $\Rightarrow$ Ergebnis näher an Ausgangslösung

Legalisierung
Partionsbasierter Ansatz

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verwendung u.a. bei StarPlace oder GORDIAN
- Rekursiv abwechselnded horizontal und vertikal unterteilen
- Solange bis nur noch eine Zelle in einer Partition
  Diese eine Zelle legalisieren
- Details später bei Partitionierer

Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Analytische Placer
  - Kräftebasiertes Platzieren
  - StarPlace
  - APlace
- Legalisierung