

Algorithmen im Chip-Entwurf 10

Verbesserungen

Andreas Koch
FG Eingebettete Systeme
und ihre Anwendungen
TU Darmstadt

- **Verbesserung der Platzierung**
- **Vorschläge**
 - Verdrahtungsorientierte Platzierung
 - Parallele Platzierung
- **Brainstorming**

Verbesserungen

■ Basisfunktionen

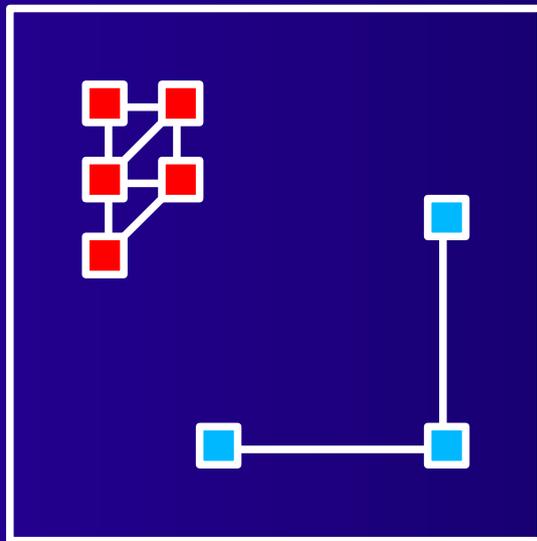
- Platzierung
- Verdrahtung

■ Vorschläge zur Verbesserung

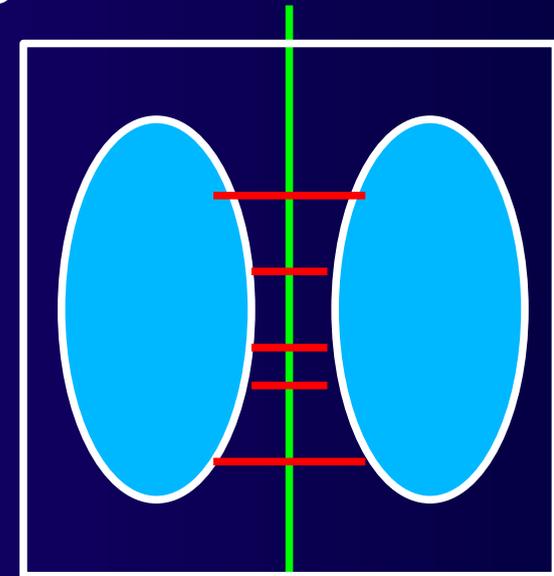
- Bessere Ergebnisse
 - ◆ Weniger Tracks
 - ◆ Weniger Verzögerung
- Schnellere Laufzeiten
 - ◆ Um welchen Preis?

Verdrahtungsorientierung

- Während der Platzierung
- Bisher nur sehr einfach
 - Gesamtverdrahtungslänge
- Nicht berücksichtigt
 - Örtliche Verdrahtungsdichte
 - Schnitt-basierte Verdrahtungsdichte



Örtlich



Schnitt-basiert

Örtliche Verdrahtungsdichte

- **Abschätzungen**
- **Über Netzdichten**
 - Flächenorientiert
 - Ansatz von RISA
 - ◆ Cheng: ICCAD 1994
- **Statistische Modelle**
 - Pfadorientiert
 - Ansatz von fGREP und fGREP2
 - ◆ Bhatia: FPL 2001, DAC 2002

Berechnung von Netzdichten

■ Idee: Netzlänge pro Ressource

- Abschätzung Netzlänge $L(n)$
 - ◆ Anzahl Terminals
 - ◆ Korrekturfaktor q
- Fläche ist bekannt
 - ◆ Netzumspannendes Rechteck BB
 - ◆ Daraus Anzahl Verdrahtungsressourcen bestimmen
 - ◆ $R(BB(n)) = \text{Anzahl Metallsegmente in Fläche } BB(n)$

■ Annahme

- Netzlänge verteilt sich homogen auf Verdrahtungsressourcen

$$d(n) = \frac{L(n)}{R(BB(n))}$$

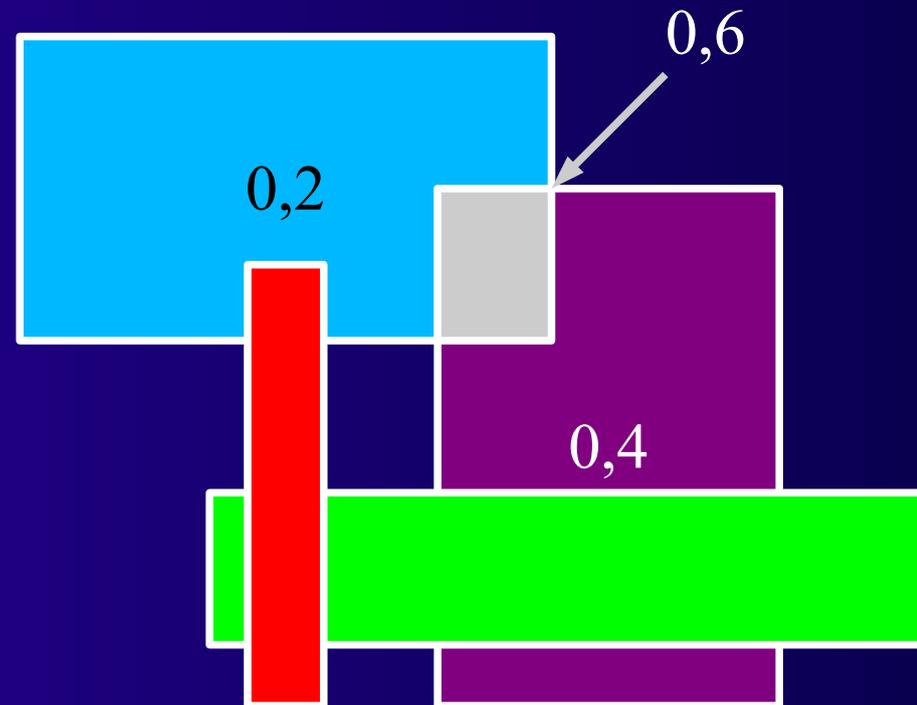
Berechnung von Netzdichten

■ Vorgehen

- Berechne $d(n)$ für alle Netze n
 - ◆ Anhand aktueller Platzierung (z.B. je SA Schritt/e)
- Berechne örtliche Dichte für CLB c_{ij}
 - ◆ Summiere $d(n)$ für alle Netze n mit (i,j) in $BB(n)$

■ Optimierungsziel

- Zusätzlich zu Verzögerung und Gesamtlänge
- Minimiere örtliche Dichte
 - ◆ Präziser: Überschreitung der vorhandenen Tracks
 - ◆ Theoretisch: $d(n) > 1$
 - ◆ Aber: Abschätzung sehr ungenau
 - ◆ Korrekturfaktor durch Analyse realer Verdrahtungen
 - ◆ Vergleich mit Abschätzungen
 - ◆ qi.txt



■ Aufsummieren der Netzdichten

- Je Block
- Verfeinerung
 - ◆ Getrennt nach H und V Segmenten
 - ◆ Vorzugsrichtung des Netzes

Statistische Modellierung

- **Pfadwahrscheinlichkeiten**
 - Nicht mehr flächenorientiert
 - Berücksichtigen pro Netz n
 - ◆ Anzahl der Terminals
 - ◆ Lage der Terminals in $BB(n)$
- **Genauer als flächenorientierter Ansatz**
 - 3,8% Fehler statt 34% (RISA, ICCAD 1994)
- **Aber langsamer**
 - Praktisch ca. 660x
 - ◆ Bhatia, FPL 2001
 - Verbesserbar bis auf 37x langsamer
 - ◆ Bhatia, DAC 2002
- **Beide Papers auf Web-Site**

- **Bei wachsender Entfernung vom Ursprung**
 - Mehr Möglichkeiten der Pfadführung
- **Benutzungswahrscheinlichkeit für gegebene Verdrahtungsressource (VR)**
 - Umgekehrt proportional zu Möglichkeiten
- **Bei mehreren Ursprüngen**
 - Nahegelegenster bestimmt die Nachfrage

- Von Terminal t equidistante VR r der Distanz q

$$LS(t, q) = \{r \in R \mid \text{dist}(t, r) = q\}$$

- Terminalnachfrage von t nach VR r

$$TD(t, r) = \frac{1}{|LS(t, \text{dist}(t, r))|}$$

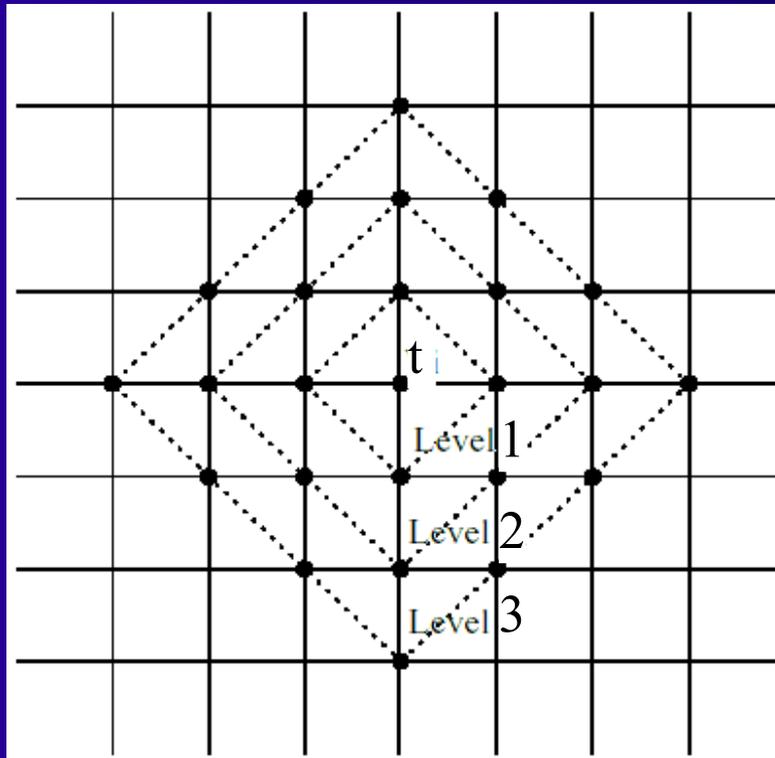
- Netznachfrage von n nach VR r

$$ND(n, r) = \max_{t \in \{u \in n \mid \text{dist}(u, r) = \min_{v \in n} \text{dist}(v, r)\}} TD(t, r)$$

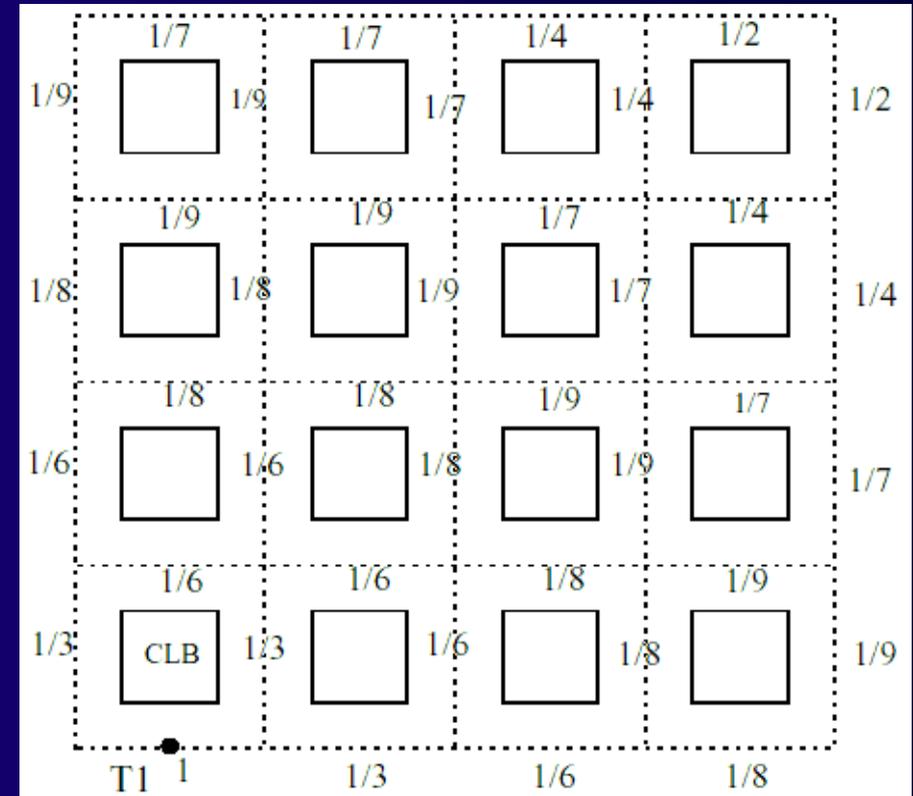
- Gesamtnachfrage nach VR r

$$D(r) = \sum_{n \in N} ND(n, r)$$

Beispiel 1

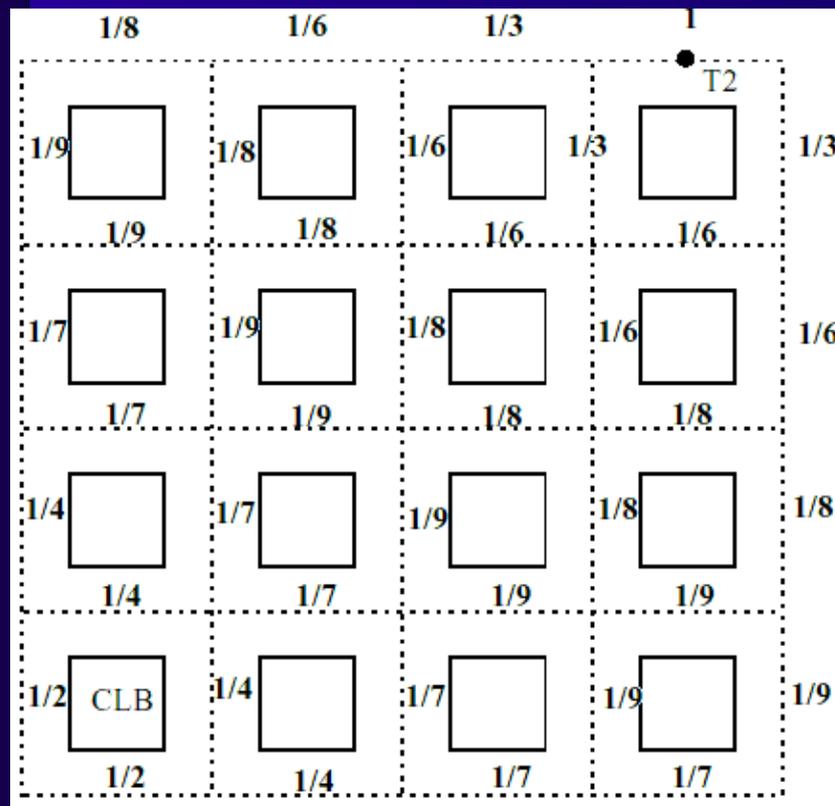


Level-Set $LS(t, q)$ für
 $q=0, 1, 2, 3$

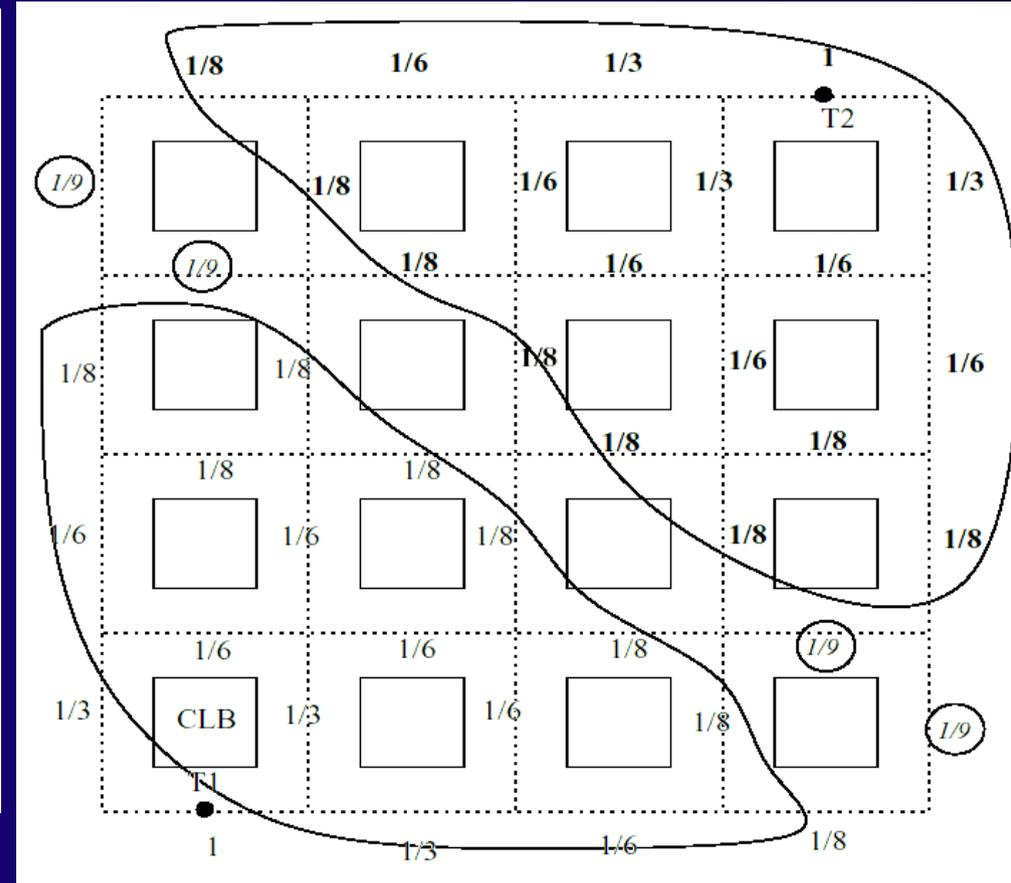


Terminalnachfrage $TD(T1, r)$
für alle r in $BB(n)$

Beispiel 2



Terminalnachfrage $TD(T2, r)$
für alle r in $BB(n)$

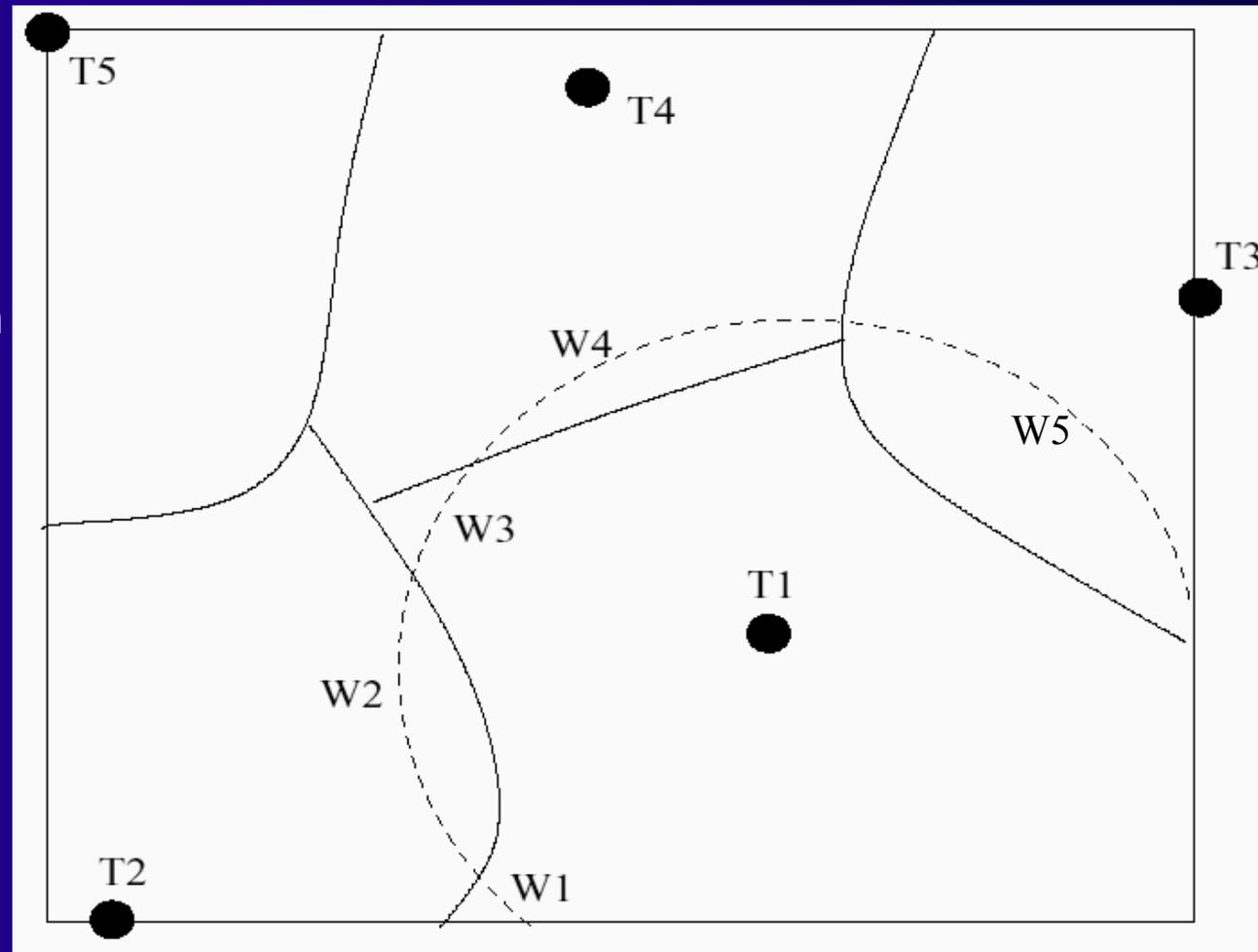


Netznachfrage $ND(n, r)$
für alle r in $BB(n)$
mit $n = \{T1, T2\}$

- **Gesamtnachfrage berechnen**
 - Aufsummieren über alle Netze
- **Algorithmus**
 - Breitensuche ausgehend von Terminal
 - ◆ Bis alle r in $BB(n)$ besucht
 - Komplexität $O(|n| \cdot |R|)$
- **Laufzeit**
 - 5-20x schneller als VPR Router (für geg. W_h, W_v)
- **Verbesserung (DAC 2002)**
 - Einflusszonen um Terminal herum
 - ◆ Nachfrage wird nur durch ein Terminal bestimmt
 - ◆ Ausnutzen: Nachfrage nach jedem r nur 1x berechnen
 - ◆ **Nicht mehr 1x pro Netz**
 - ◆ Komplexität dann nur noch $O(|R|)$

Beispiel Verbesserung

Voronoi-Regionen



Einflusszonen um Terminals $T_1 \dots T_5$
Fragmentierung der Wellenfront in $W_1 \dots W_4$

■ Parallele BFS von allen Terminals t in n

- Erreichen einer Zonengrenze
 - ◆ Zwei Wellenfronten treffen aufeinander

■ Problem

- Andere Teile der Wellenfront sind noch in Zone
- Nicht einfach aufhören
 - ◆ $LS(t, q)$ wäre falsch (zu klein!)
 - ◆ $|W_1| + |W_3|$ statt $|W_1| + |W_2| + |W_3| + |W_4| + |W_5|$

■ Lösung

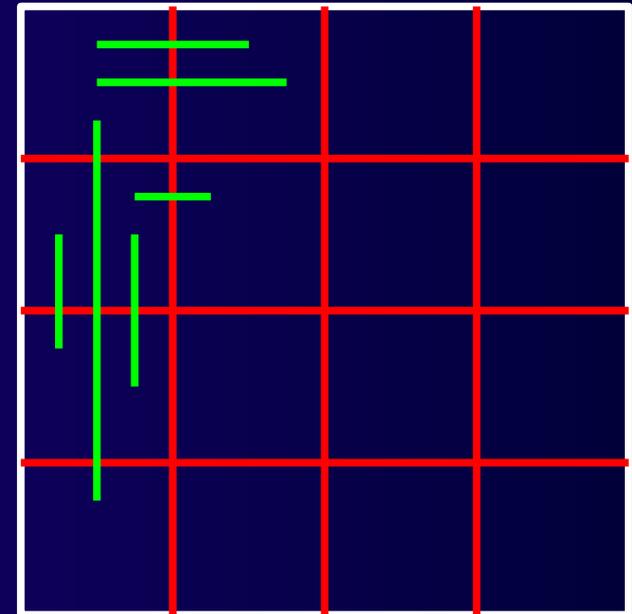
- Wellenfront W von t aus weiter ausbreiten
 - ◆ Solange auch nur ein r aus W in der Zone von t liegt
 - ◆ Aber keine Dichten eintragen, nur $LS(t, q)$ korrekt aufbauen

■ Resultat

- Gleiche Ergebnisse, 17x schneller

Schnittdichten

- Einfachere Idee, aber ungenauer
- Chip in mehrere Regionen aufteilen
 - 4x4, 16x16
- Je Schnittsegment
 - Horizontal und vertikal
 - Anzahl der kreuzenden Leitungen berechnen
 - Falls grösser als Anzahl Tracks
 - ◆ In Kostenfunktion bestrafen



Parallele Platzierung

- Ziel ist kürzere Programmlaufzeit
- Vorgehen
 - Aufteilung des Platzierungsproblems
 - Verteilen auf N CPUs/Rechner
 - Lösen der Teilprobleme
 - Zusammenfügen zur Gesamtlösung
- Im Idealfall
 - Beschleunigung um Faktor N
 - Praktisch aber nicht:
 - ◆ Aufteilen: auf einer CPU
 - ◆ Kommunikation: Zusätzlicher Overhead
 - ◆ Zusammenfügen: I.d.R. auf einer CPU
 - ◆ Auch hierarchisches Vorgehen möglich

■ Einfache Lösung

- Partitionierung mittels Min-Cut
- Problem aufteilen
 - ◆ In möglichst unabhängige Unterprobleme

■ In heterogenen Systemumgebungen

- Partitionsgröße an Rechenleistung anpassen

■ Selbst bei Aufteilung

- Kontextinformationen bestimmen
 - ◆ Relativ einfach: Topologie
 - ◆ Komplizierter: Delay
 - ◆ Verteilte Abschätzung oder zentrale Criticality-Berechnung
 - ◆ Über Partitions Grenzen verteilen

■ Aber: Aufteilung vermindert Qualität

- Prinzip der Optimalität gilt nicht!

Implementierung

- **Einlesen der Eingabedaten**
 - Aufteilen
 - Bestimmen von Kontextdaten
- **Verteilen auf Rechenknoten**
 - Dort ständig laufende Server-Prozesse
 - Nehmen Anfragen via Java RMI entgegen
 - ◆ Teilproblem und Kontextdaten
 - Berechnung mit üblichem Algorithmus
 - Ggf.: Zentrale Berechnung der Criticality
 - ◆ Kommunikation mit Master-Knoten
- **Zusammenfügen der Teilergebnisse**
 - Übertragen auf Master-Knoten
- **Schreiben der Ausgabedaten**

Zusammenfassung

- **Verbesserungen des P&R-Flusses**
- **Qualität**
 - Verdrahtungsorientierte Platzierung
- **Rechenzeit**
 - Parallele Platzierung