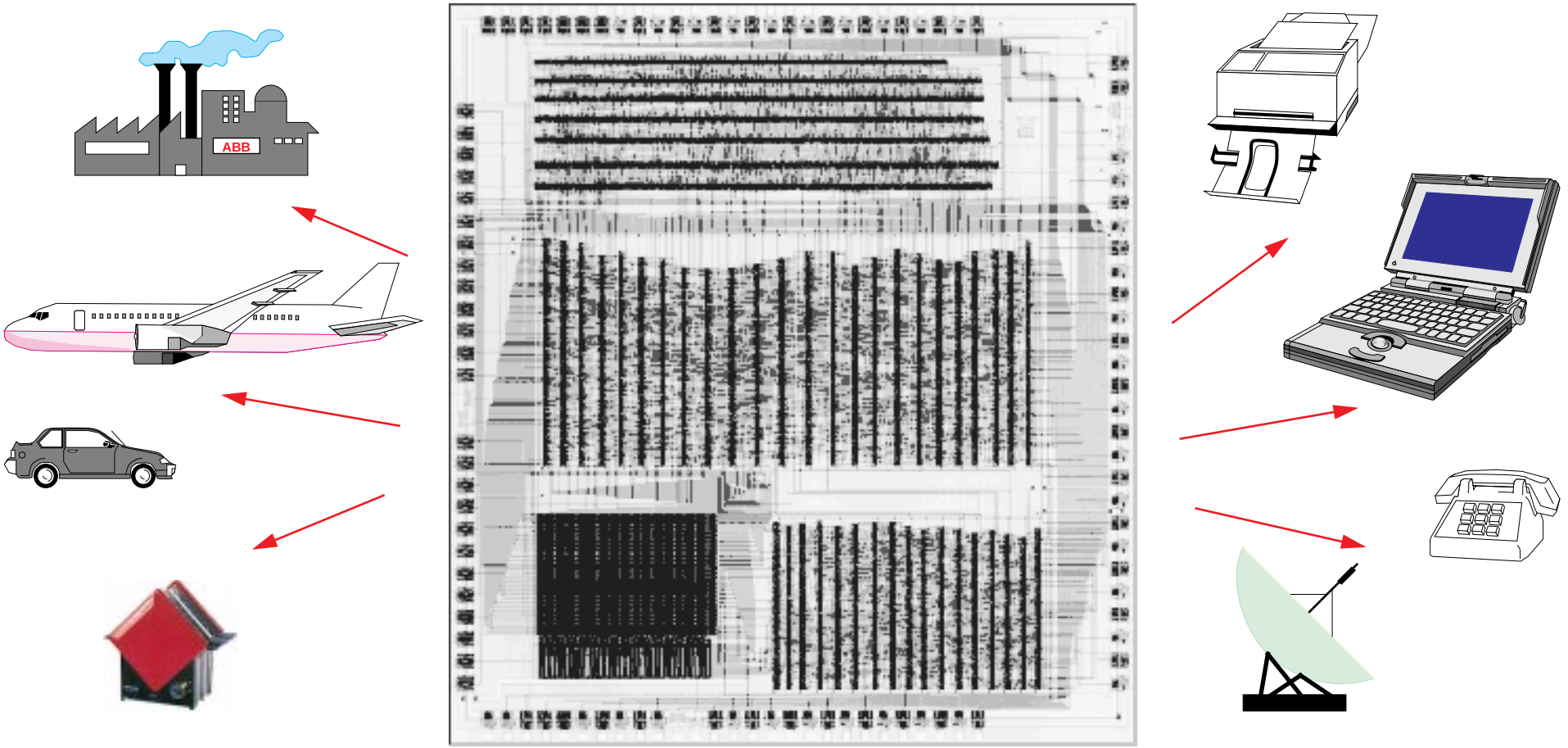
Timing-driven Floorplanning beim hierarchischen VLSI-Entwurf

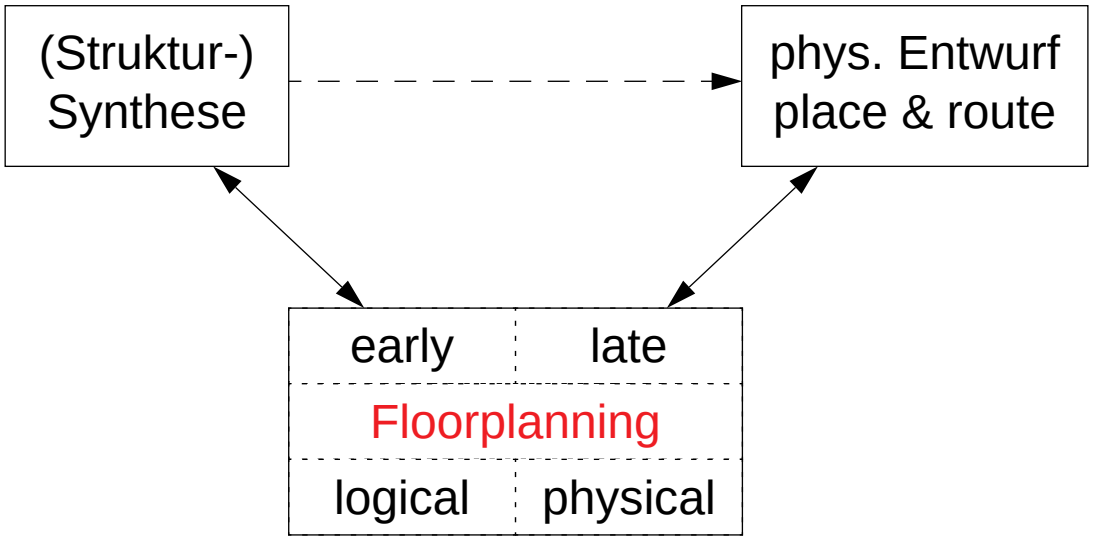
Manfred Schölzke

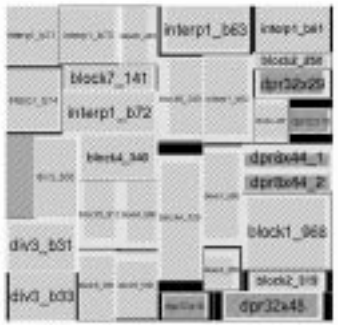
ÜBERSICHT

- Einleitung
- Kernfunktionalitäten im Floorplanning
- Timing-driven Placement
- Messungen und Ergebnisse
- Zusammenfassung

EINLEITUNG







Globalziele:

- Kürzere Designzeiten
- bessere Entwurfsergebnisse
- weniger Entwurfszyklen

durch:

- engere Kopplung
- verbesserte Vorhersagen
- Wiederverwendung
- Hierarchie

MOTIVATION UND ZIELE

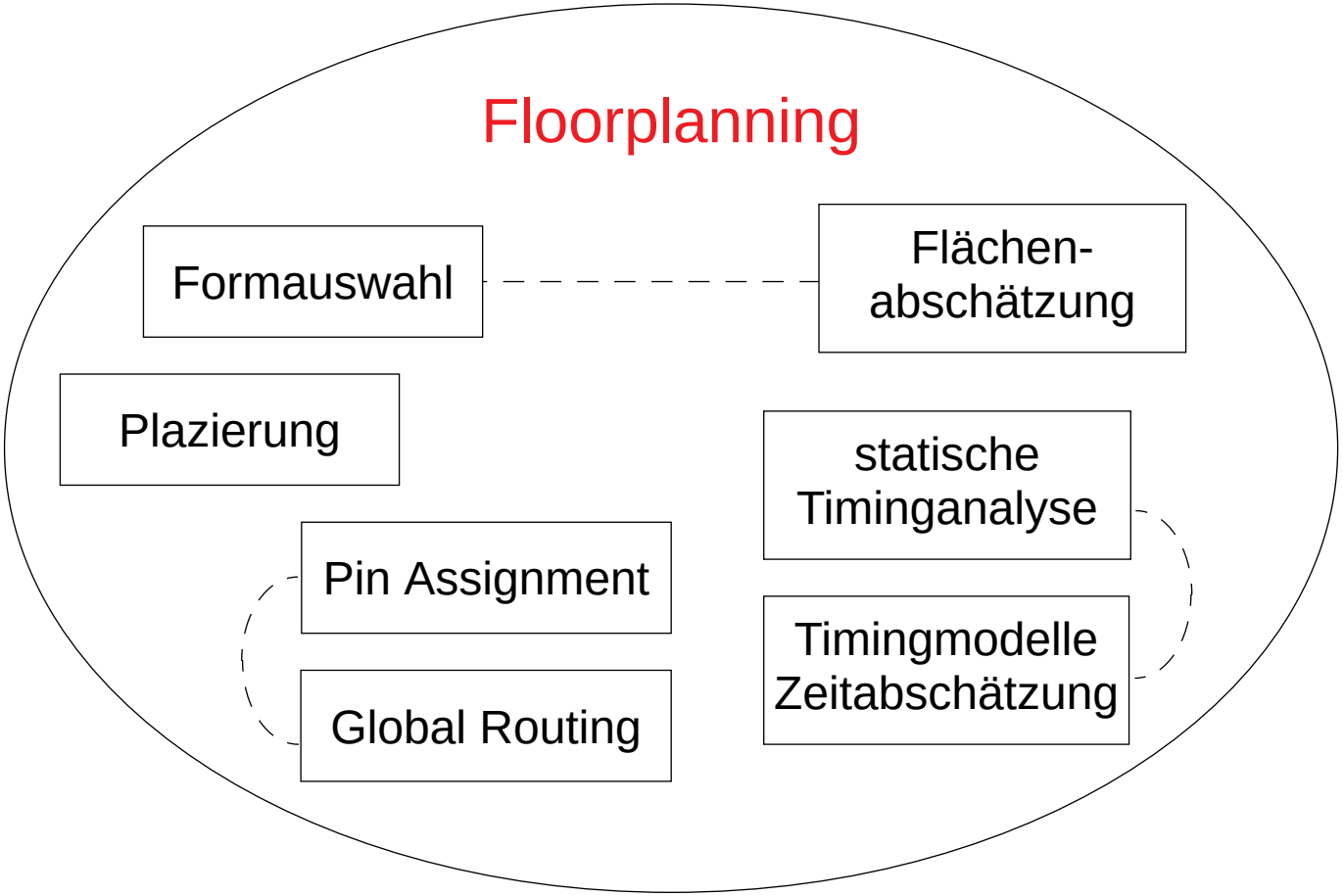
Fragen:

- nach der kleinsten Realisierung einer Schaltung?
- nach der schnellsten Realisierung?
- nach der günstigsten Realisierung bei vorgegebenem Kostenmaß?
- nach dem Trade-Off zwischen Chipfläche und Zeitverhalten?
- verknüpft mit der Suche nach guten Physical Design Algorithmen

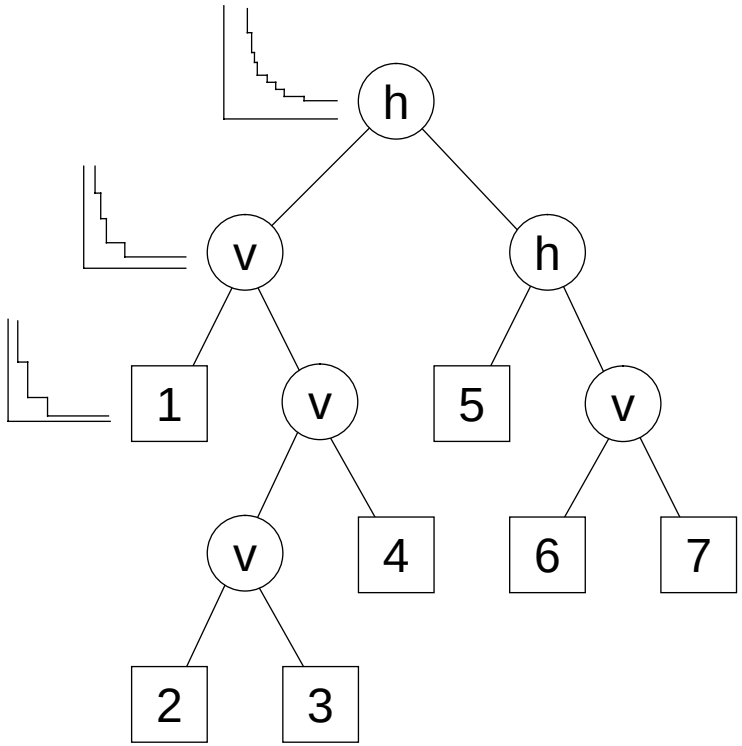
Thesen:

- Die kleinsten Schaltungen sind die schnellsten
- Im (riesigen) Lösungsraum gibt es viele Lösungen mit kleiner Fläche und darunter einige wenige mit gutem Zeitverhalten -> diese gilt es zu finden
- Um sehr schnelle Schaltungen zu erreichen, muß man zusätzliche Fläche opfern

KERNFUNKTIONALITÄTEN IM FLOORPLANNING



FLÄCHENABSCHÄTZUNG



Slicingbaum

Operationen auf Formkurven:

h — horizontale Addition

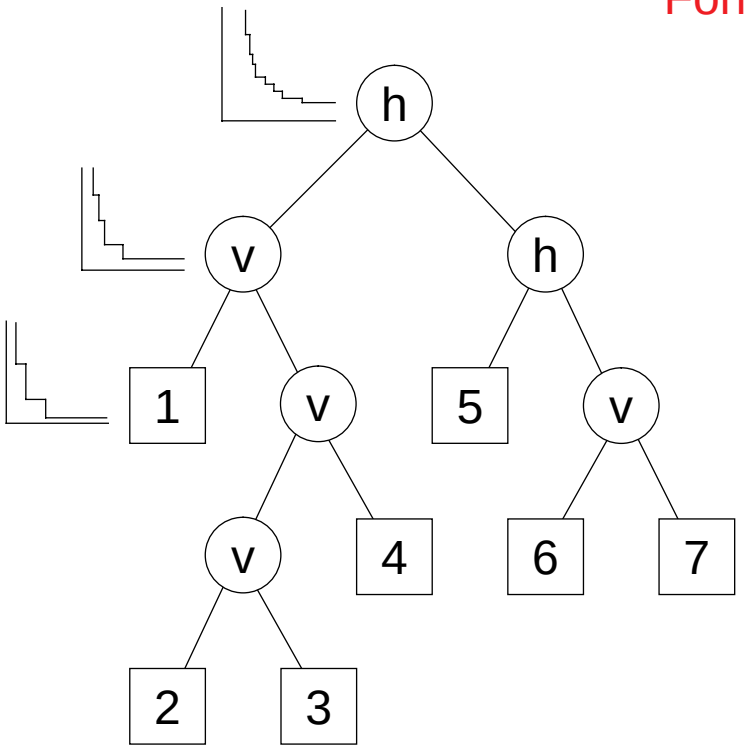
v — vertikale Addition

optimale Überlagerung

Verdrahtungsaufschläge

Flächenabschätzung für Slicingstrukturen

FORMAUSWAHL



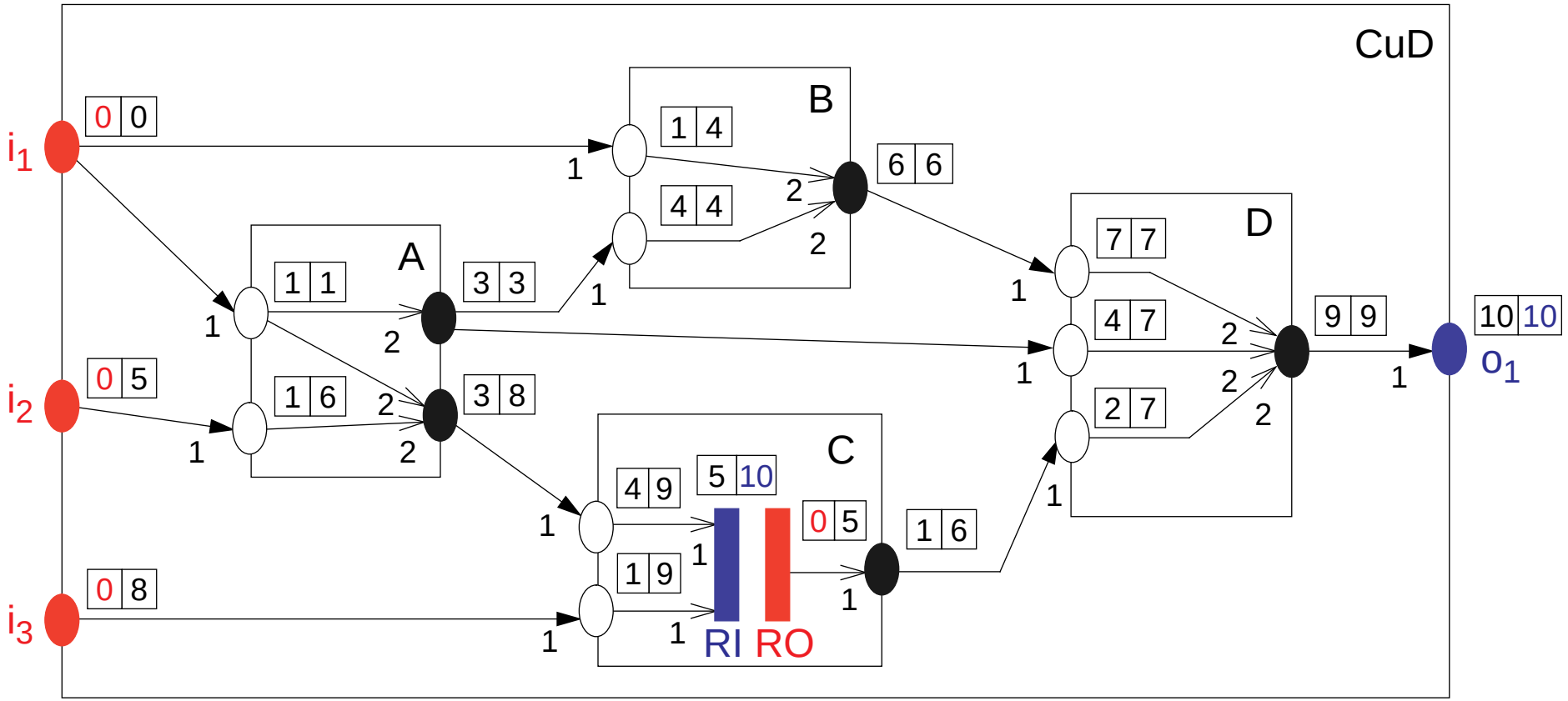
Formauswahl

h teile horizontal auf

v teile vertikal auf

teile entsprechend der bei der optimalen Überlagerung gemerkten Schnittrichtung auf

auf hierarchischen Timinggraphen



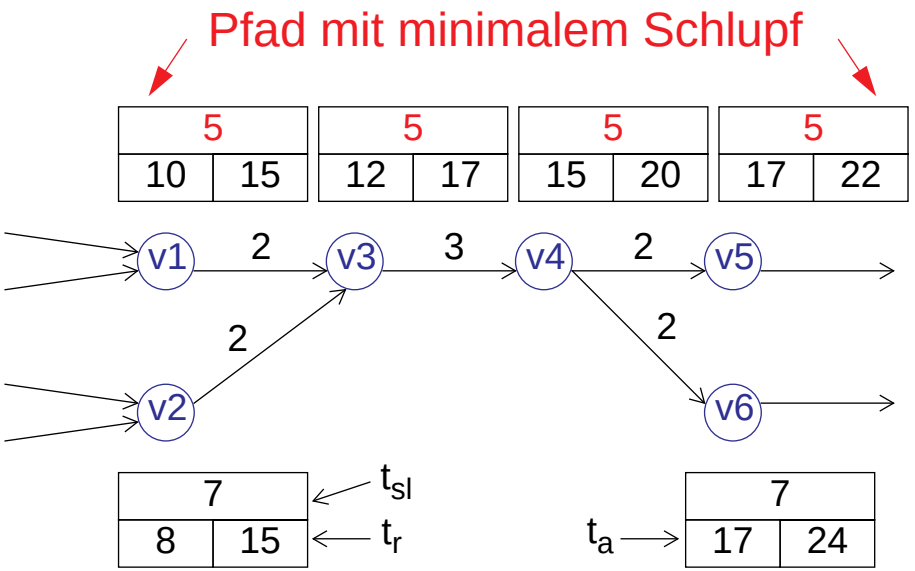
Vorwärtsrechnung

$t_a(v_j)$ $t_r(v_j)$ = “Ist-Zeit, Soll-Zeit”

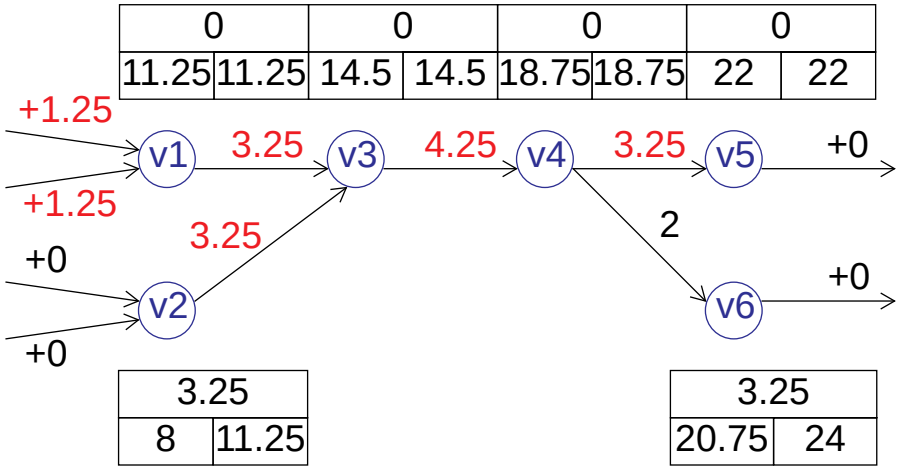
$t_r(v_j) - t_a(v_j)$ = “Schlupf”

Rückwärtsrechnung

SCHLUPFAUFTEILUNG



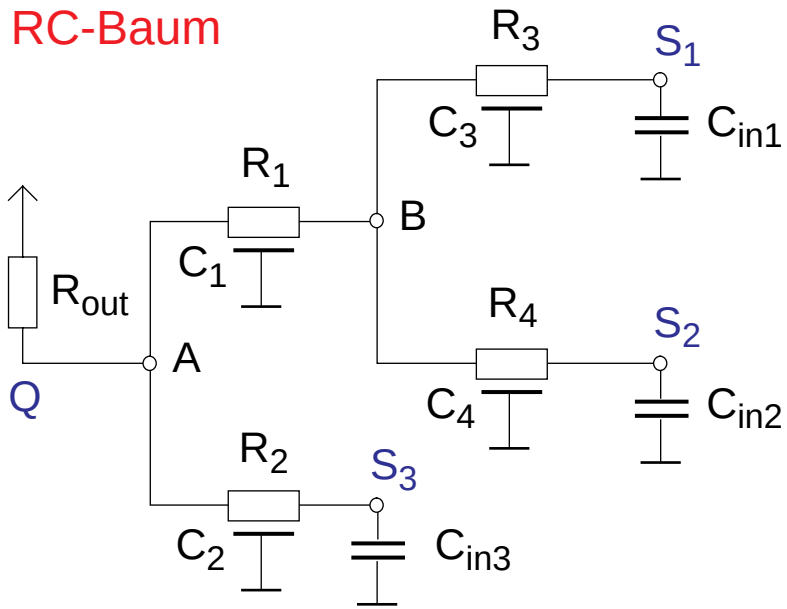
vor der Schlupfaufteilung



nach der Schlupfaufteilung

auf die eingehenden Kanten

RC-Baum



$\alpha = 0{,}5$ und $\beta = 1$

-> Elmore Delay-Modell

$\alpha = 0{,}59$ und $\beta = 1{,}21$

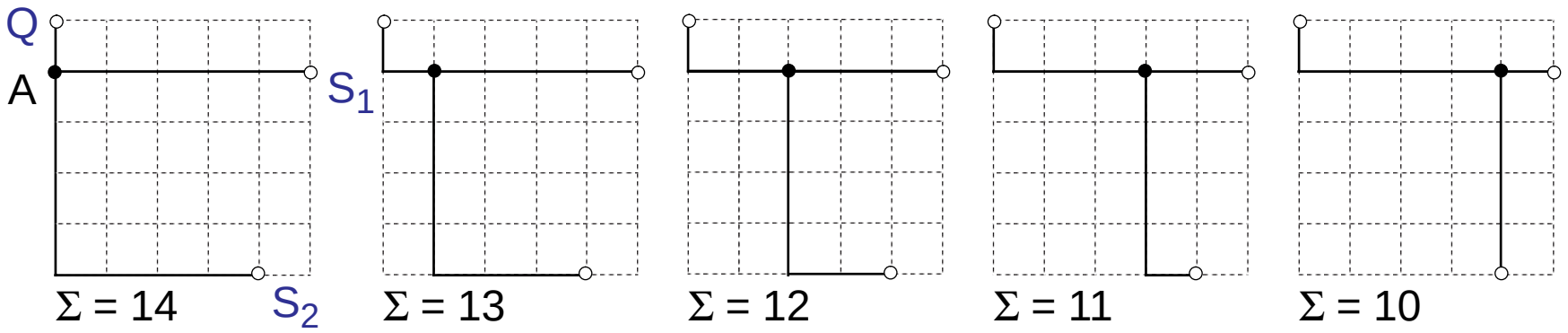
-> $t_{0,7}$-Modell

$$d(A) = \beta R_{out}\left(C_1 + C_2 + C_3 + C_4 + C_{in1} + C_{in2} + C_{in3}\right)$$

$$d(B) = d(A) + \alpha R_1 C_1 + \beta R_1\left(C_3 + C_4 + C_{in1} + C_{in2}\right)$$

$$d\left(S_1\right) = d(B) + \alpha R_3 C_3 + \beta R_3 C_{in1}$$

TIMINGMODELLE II



	Σ = 14	Σ = 13	Σ = 12	Σ = 11	Σ = 10
$d_{elmore}(S_1)$	1,518	1,621	1,679	1,691	1,658
$d_{elmore}(S_2)$	2,016	2,052	2,041	1,985	1,883
$t_{0,7}(S_1)$	1,827	1,955	2,026	2,040	1,998
$t_{0,7}(S_2)$	2,418	2,464	2,454	2,388	2,265

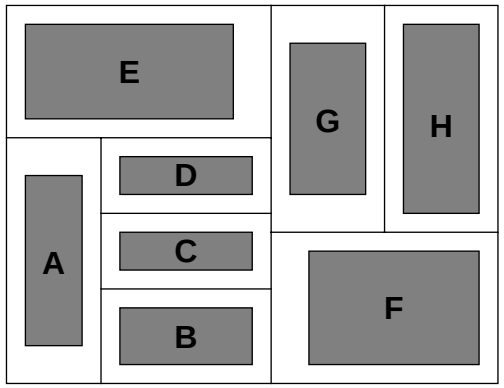
$r_1 = 0{,}12\Omega/\mu m$

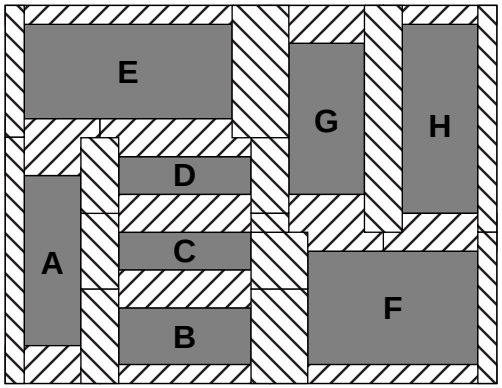
$c_1 = 0{,}19fF/\mu m$

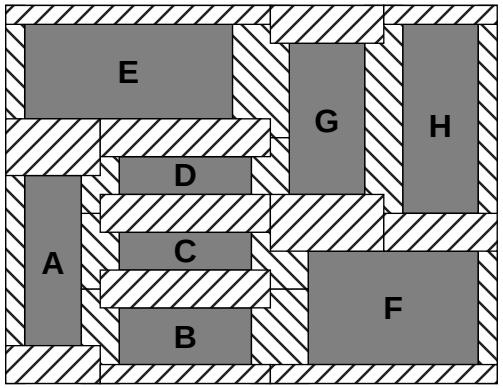
$R_{out} = 270\Omega$

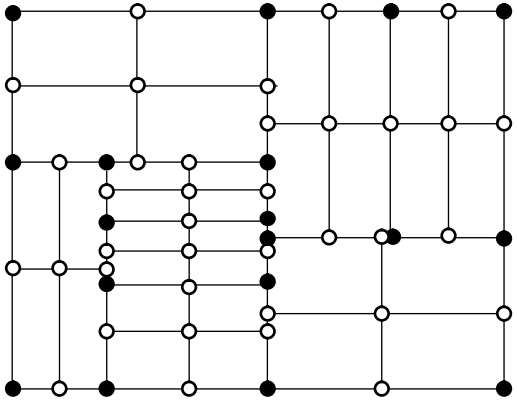
$C_{in} = 0{,}15pF$

GLOBAL ROUTING I

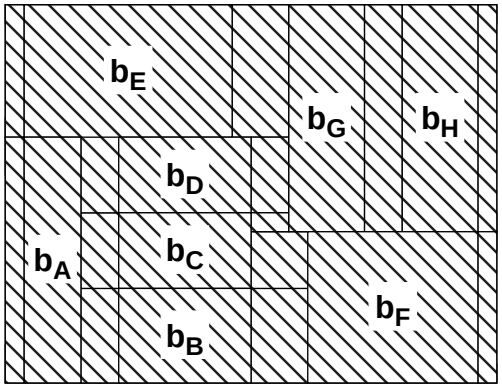


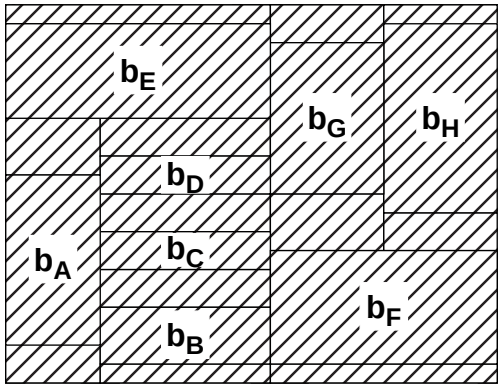






Routinggraph





GLOBAL ROUTING II

Algorithmus:

- sequentiell, prioritätsgesteuert
- mit dynamischer Bewertung der Kanalkosten
  1. Ziel: Optimierung des Zeitverhaltens
  2. Ziel: Minimierung der Chipfläche
  (3. Ziel: Minimierung der Gesamtnetzlänge)
- konstruktive Steinerbaumheuristik mit
  - verschiedenen Optimalitätskriterien:

    SERT-C -> pdGR (min. die Delays nur zu den zeitkritischen Senken)

    SERT -> pdGR-A (min. die Delays zu jeder Senke eines jeden Netzes)

    SMT -> pdGR-N (keine Minimierung der Delays, sondern der gewichteten Netzlängen)

PDGR - ALGORITHMUS

Eingabe: Netzliste N (sortiert), Routinggraph RG

Ausgabe: globale Verdrahtung von N auf RG

Algorithmus pdGR:

- für alle n aus N in Sortierreihenfolge:
  - erweitere RG in Abhängigkeit von n
  - berechne (neue) Kantengewichte
  - separiere die Terminals $\{s_0, \ldots, s_{k-1}\}$ von n in S_0, S_{crit}, S_{uncrit}
  - T_1 <- (S_0, {})
  - falls S_{crit} <> {}
    - T_2 <- RoutingTree (T_1, S_{crit}, SERT)
    - T <- RoutingTree (T_2, S_{uncrit}, SERT-C)
  - sonst
    - T <- RoutingTree (T_1, S_{crit} + S_{uncrit}, SMT)
  - entferne netzspezifische Erweiterungen aus RG
  - berechne (neue) Belegungszustände der Kanäle

PROZEDUR ROUTINGTREE

Eingabe: Quelle s_0 (oder Teilbaum), Senken s_1, ... , s_{k-1},
Optimalitätskriterium K_{opt}, Routinggraph RG

Ausgabe: Steinerbaum $T = (V_T, E_T)$

Prozedur RoutingTree:

```
T <- initialer Baum
S <- Menge der verbleibenden Senken (die nicht in V_T sind)
solange S <> {}
    finde s aus S und v aus V_T mit T + minpath(v, s) ist optimal gemäß K_opt
    T <- T + minpath(v, s)
    S <- S - {s}
output << T
```

Anmerkung: K_{opt} aus {SMT, SERT, SERT-C}

ALGORITHMUS SERT-C

Eingabe: Quelle s_0 (oder Teilbaum), Senken s_1, ... , s_k, wobei s_c zeitkritisch ist

Ausgabe: Steiner Elmore Routing Tree = (V, E)

Algorithmus SERT-C:

V <- $\{s_0, s_c\}$; E <- $\{e_{0,c}\}$; V_{quer} <- $\{s_1, \ldots, s_k\}$ - $\{s_c\}$

solange V_{quer} <> {}

suche s_i aus V_{quer} und (v, v') aus E und einen neuen Punkt w, die

del(s_0, s_c) minimieren im Baum bestehend aus V + $\{s_i, w\}$

und E - {(v, v')} + {(v, w), (w, v'), (s_i, w)}

V <- V + $\{s_i, w\}$; V_{quer} <- V_{quer} - $\{s_i\}$

E <- E - {(v, v')} + {(v, w), (w, v'), (s_i, w)}

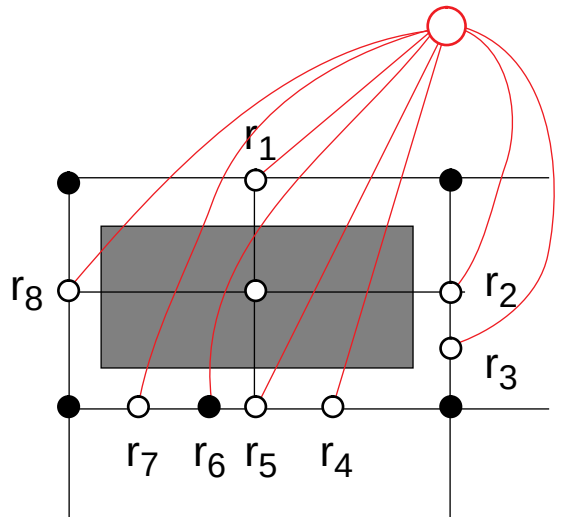
PIN ASSIGNMENT

Aufgabe: Intervallfestlegungen für flexible Subzellen

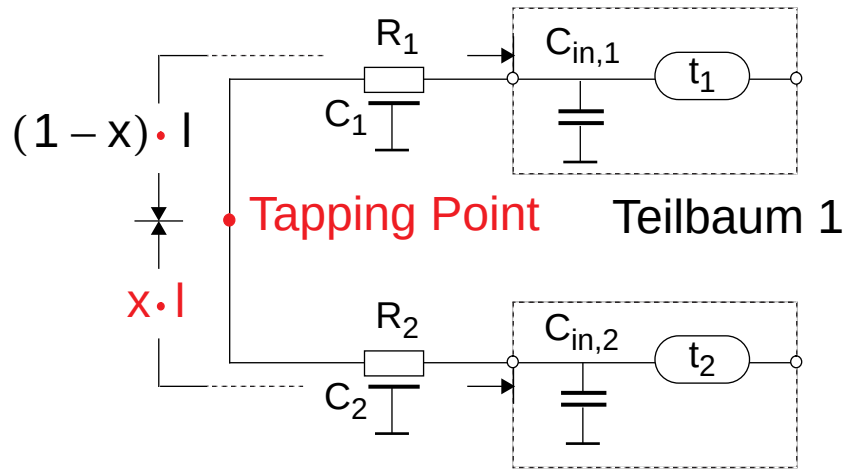
entspricht verallgemeinertem Steinerbaumproblem (mehrere Repräsentanten je Terminal)

Vorgehen:

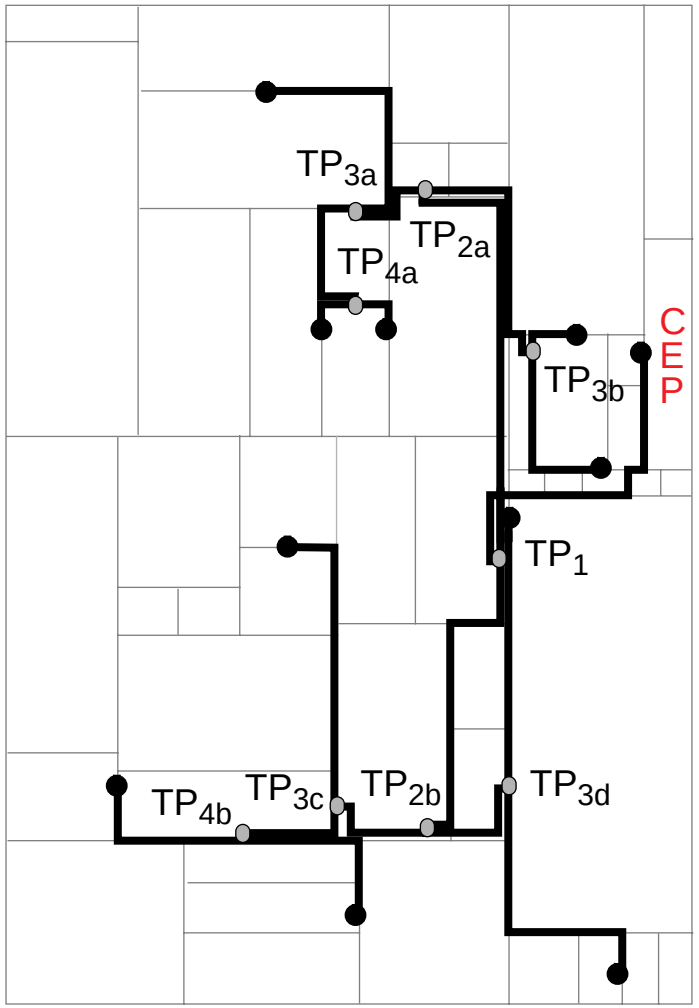
1. Hinzunahme von Superknoten, die mit allen Repräsentanten verbunden sind
2. Lösung eines “normalen” Steinerbaumproblems
3. Reduzierung des errechneten Steinerbaums auf eine Netzroute



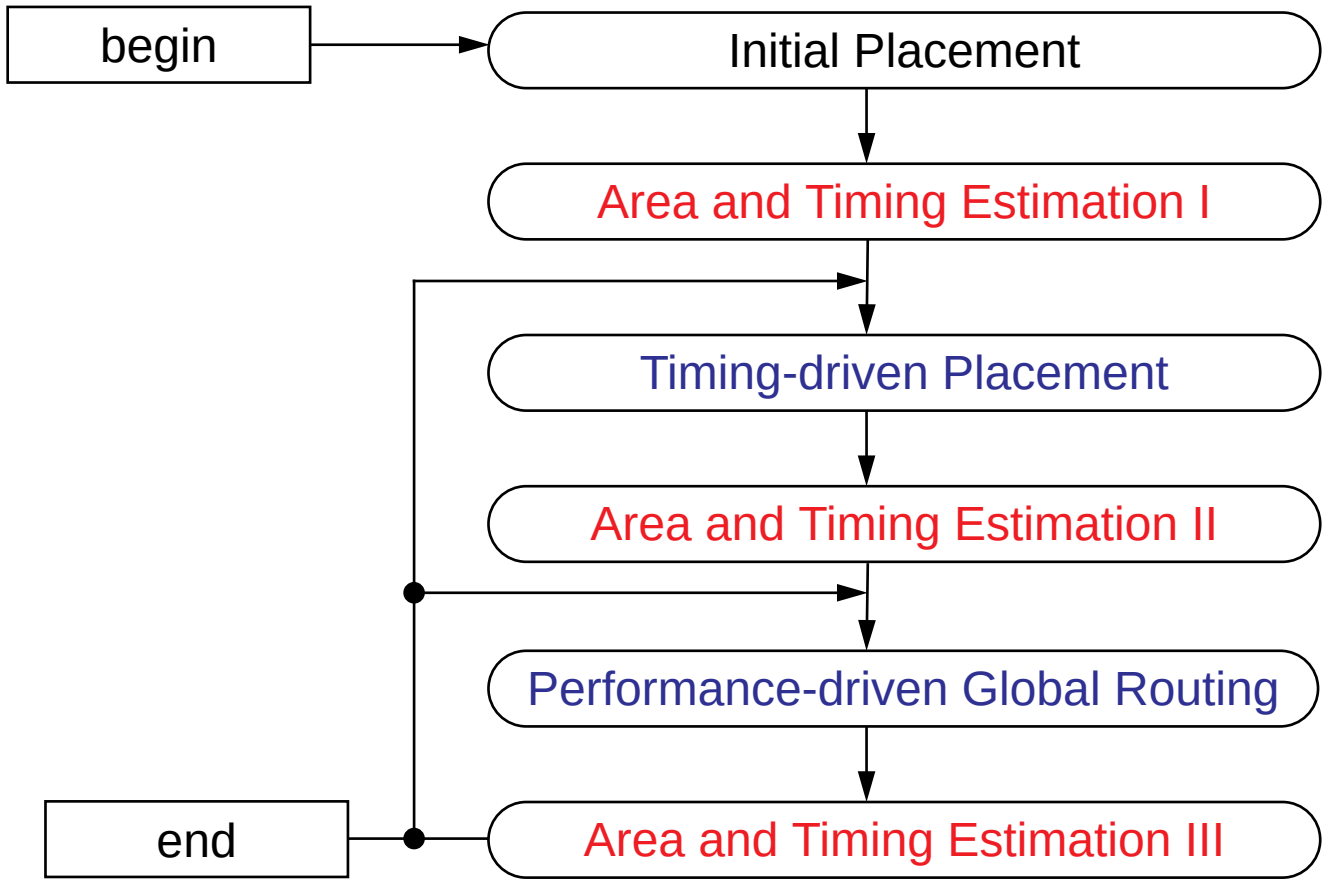
TAKTNETZVERDRAHTUNG



Beispiel für eine Taktnetz-verdrahtung mit 10 Senken und 9 Tapping Points



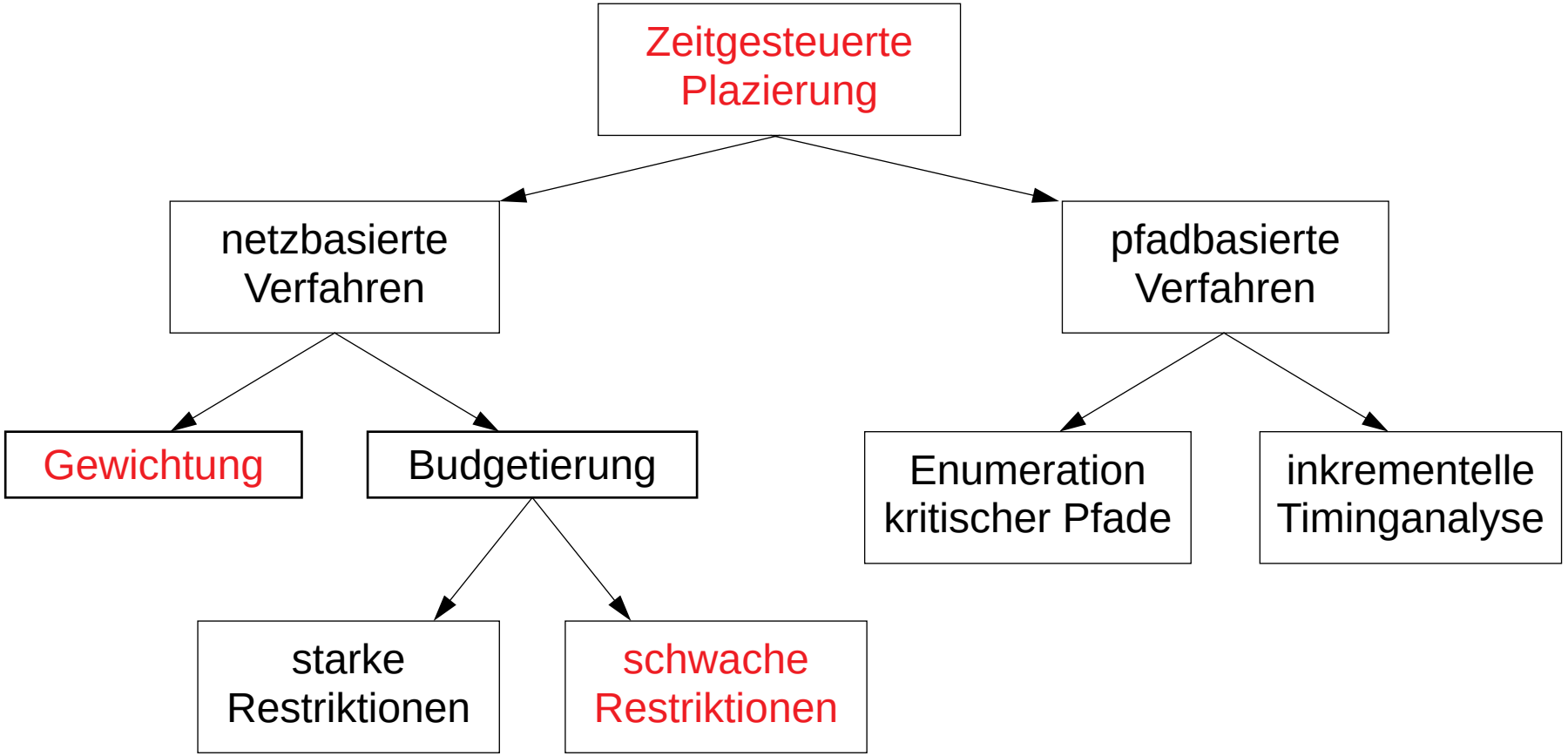
VORGEHENSMODELL ZUM TIMING-DRIVEN FLOORPLANNING



ÜBERSICHT

- Einleitung
- Kernfunktionalitäten im Floorplanning
- Timing-driven Placement
  - Klassifikation der Verfahren
  - FM-Mincut
  - Terminal-Propagierung (TP)
  - TPmK (mit Klassifikation)
  - TPmB (mit Budgetierung)
- Messungen und Ergebnisse
- Zusammenfassung

VERFAHREN ZUR ZEITGESTEUERTEN PLAZIERUNG



FM-MINCUT

Struktur des Mincut-Algorithmus von Fiduccia / Mattheyses:

starte mit einer zufälligen Bipartitionierung A, B von V
solange es Verbesserungen gibt:
vertausche die beiden Teilmengen X von A, Y von B, die den höchsten Gewinn bringen (unter Einhaltung der Balancebedingung)

Ein Durchlauf (Bestimmung von X und Y):

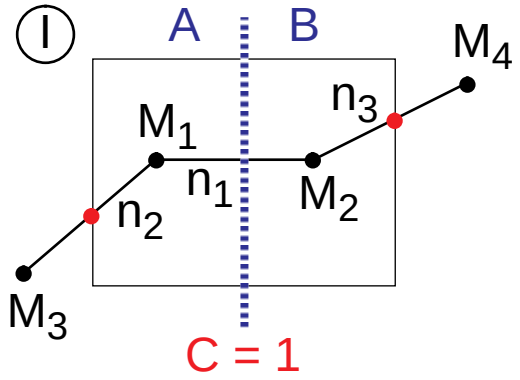
solange es bewegliche Knoten gibt:
bewege den Knoten mit dem höchsten erzielbaren Gewinn (unter Einhaltung der Balancebedingung)
sperre den bewegten Knoten für weitere Bewegungen
berechne die geänderten Gewinne und aktualisiere die Bucket-Datenstruktur

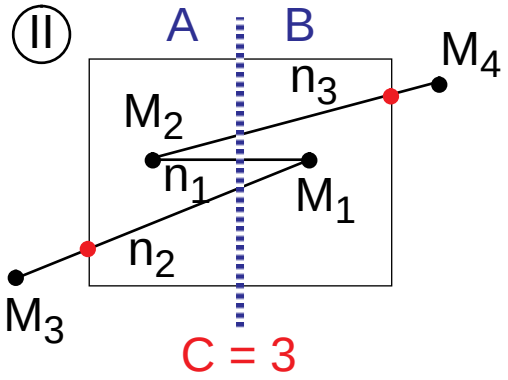
Prinzip:

Berücksichtigung geometrischer Information, d. h. Lage externer Module

Technik:

Hinzufügen gesperrter Dummyknoten mit Gewicht 0.



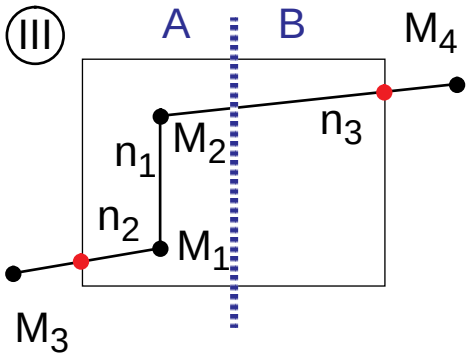


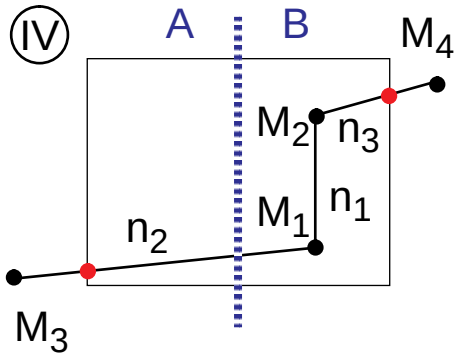
Vier Gewichtungsfaktoren:

- zeitkritische interne Netze: 1,0
- zeitkritische externe Netze: 1,0
- unkritische interne Netze: 0,5
- unkritische externe Netze: 0,1

(I) wie oben

(II) wie oben



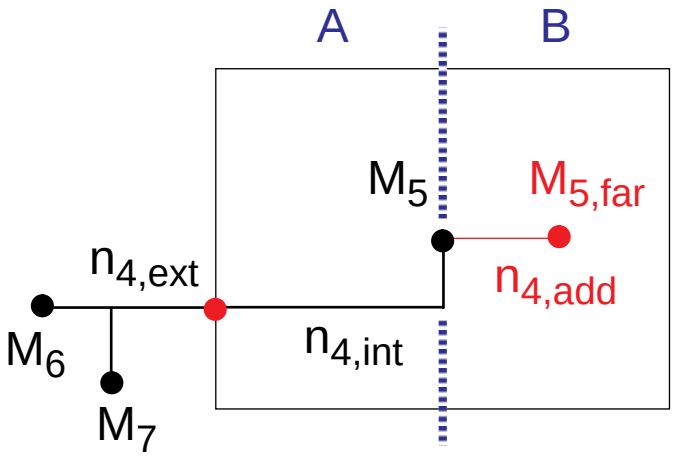


Schnittkosten:

Fall	Schnitte	FM	TP	TPmK	000	001	010	011	100	101	110	111
I	n_1	1	1	0,5 - 1,0	0,5	0,5	0,5	0,5	1,0	1,0	1,0	1,0
II	n_1,n_2,n_3	1	3	0,7 - 3,0	0,7	1,6	1,6	2,5	1,2	2,1	2,1	3,0
III	n_3	0	1	0,1 - 1,0	0,1	1,0	0,1	1,0	0,1	1,0	0,1	1,0
IV	n_2	0	1	0,1 - 1,0	0,1	0,1	1,0	1,0	0,1	0,1	1,0	1,0

Idee:

Werte eine versuchsweise Zuodnung der Module in die jeweils von den Dummyknoten entfernte Partition aus, um bei ggfs. festgestellter Verletzung eines Zeitbudgets das entsprechende Modul an die nahe Partition zu binden
(binden zu wollen -> weak restrictions)



TPMB - ALGORITHMUS

Initialisierungsschritt (in der Timinganalyse mit Schlupfverteilung):

für alle Kanten e_j im Timinggraphen TG berechne das Zeitbudget $w(e_j)$

Terminal-Propagierung-mit-Zeitbudgets:

für alle zu partitionierenden Module M_j:

für alle externen Netze n_i, die an M_j anschließen:

bilde einen low-cost Steinerbaum (als Näherung für kostenmin. Steinerbaum)

berechne die durch n_i induzierten Dummyknoten

falls (mindestens) ein Dummyknoten in A *(in B)* liegt:

führe die (temporäre) Verschiebung M_j -> $M_{j,far}$ durch

falls es eine Kante e_k zu n_i gibt mit $del(e_k) > w(e_k)$

vermerke den Bindungswunsch "M_j soll nach A *(nach B)*"

rufe die Konfliktlösungskomponente auf

ÜBERSICHT

- Einleitung
- Kernfunktionalitäten im Floorplanning
- Timing-driven Placement
- Messungen und Ergebnisse
  - Beispielentwürfe
  - Skalierungen
  - Vergleiche
  - Bewertungsmethodik
  - Ergebnisse
- Zusammenfassung

Beispielentwürfe:

- ES2_0.7-Technologie: adr, grp, lon (DIV^{2}A-Projekt)
- ACMOS4h-Technologie: minproc

Charakterisierung:

Beispiel	# Zellen	# Netze	# Pins	Area	GNL	TG: |V|,|E|	MaxDelay
adr	12 (0)	1484	229 (083)	8x8 mm^2	8 m	3286,11030	14 ns
grp	32 (6)	2030	234 (234)	8x8 mm^2	11 m	4379,9707	25 ns
lon	13 (6)	1178	195 (195)	6x6 mm^2	7 m	2627,4816	19 ns
minproc	15 (0)	48	368 (035)	4x4 mm^2	0,3 m	129,144	15 ns

Skalierungen:

- ES2_0.7 -> SK1, SK2, SK3
- Begrenzung / Verringerung der Subzell-Delays
- Skalierung auf die 5x5-fache Fläche

Technologie-Parameter:

	ACMOS4h	ES2_0.7	SK1 (0.7)	SK2 (0.25)	SK3 (0.25)
R_{out} [Ω]	224,0	93,0	93,0	111,6	111,6
r_1 [Ω/µm]	0,022	0,058	0,058	0,232	0,232
c_1 [fF/µm]	0,046	0,030	0,190	0,012	0,076
C_{in} [fF]	871	462	462	154	154

VERGLEICHE

Plazierungsalgorithmen:

- planner.v6: TP, TPmK, TPmB
- planner.v5: Quadropartitionierung (QP) mit Vorgaben vhh und hvv
- (random, manuelle Modifikation)
- verschiedene Parametereinstellungen (z. B. für Mincut, Ratiocut) und Iterationen

Global Routing-Algorithmen:

- pdGR, pdGR-A, pdGR-N
- mit und ohne OTC-Verdrahtung
- versch. Parametereinstellungen (Kosten für Flächentypen, Kongestion) und Iterationen

Betrachtete Kriterien und Korrelationen:

- Chipfläche (Area)
- maximale Verzögerungszeit (MaxDelay, MaxD)
- Gesamtnetzlänge (GNL)

BEWERTUNGSMETHODIK

Form eines Meßergebnisses:

<k>	x	y	Area	MaxD	MinD
B-Pl	100,0	100,0	100,0	14,21	12,56
A-Pl	101,3	102,0	103,3	14,74	11,93
GR-1	97,1	96,7	93,9	14,16	12,45
GR-2	98,7	94,1	92,9	14,18	12,48
GR-3	99,2	94,9	94,1	14,18	12,48
GR-4	99,0	95,1	94,1	14,18	12,48
∅	99,0	94,7	93,7	14,18	12,48

x: in % der Breitenvorgabe (x_{spec})

y: in % der Höhenvorgabe (y_{spec})

Area: in % der Flächenvorgabe ($Area_{spec}$)

MaxD, MinD: in ns

aus den letzten 3 GR-Iterationen: ∅

Definition eines Gütemaßes:

$$G_k = \frac{\bar{x}_k}{x_{spec}} \cdot \frac{\bar{y}_k}{y_{spec}} \cdot \frac{\overline{MaxD}_k}{MaxD_{spec}} \cdot 100$$

$$= \frac{\overline{Area}_k}{Area_{spec}} \cdot \frac{\overline{MaxD}_k}{MaxD_{spec}} \cdot 100$$

=> Tabelle (mit $MaxD_{spec}$ = 14,32 ns):

<k>	$\bar{x}$	$\bar{y}$	$\overline{Area}$	$\overline{MaxD}$	G_k
1	99,0	94,7	93,7	14,18	92,8
2	99,3	97,8	97,1	13,83	93,8
3	94,3	97,0	91,5	13,92	88,9
4	97,2	101,5	98,7	14,32	98,7

Qualitative Bewertung bzgl. der Minimierung von MaxDelay:

- Die Algorithmen finden die den Schaltungen inhärente Lokalität und nutzen sie aus
- Innerhalb des Lösungsraums werden durch die unterschiedlichen Algorithmen (und unterschiedliche Parametereinstellungen) eine Reihe guter Löungen mit akzeptablem Trade-Off zwischen Area und MaxDelay gefunden
- Bei größerem Einfluß der CUD-Netze auf das Zeitverhalten spielen TPmK und TPmB ihre Stärken zunehmend aus

Leider keine Aussage möglich, was geschieht,

- wenn ein großer Anteil der Netze zeitkritisch ist (konform / nicht konform zur “inhärenten Lokalität”)
- wenn die Eingabebeispiele eine große Anzahl an Modulen enthalten (Vermutung: TPmB wird besser als TPmK)

Bekannt [Fri96]: TP ist nicht schlechter als QP bzgl. Flächenminimierung

VERGLEICH VON TP UND QP

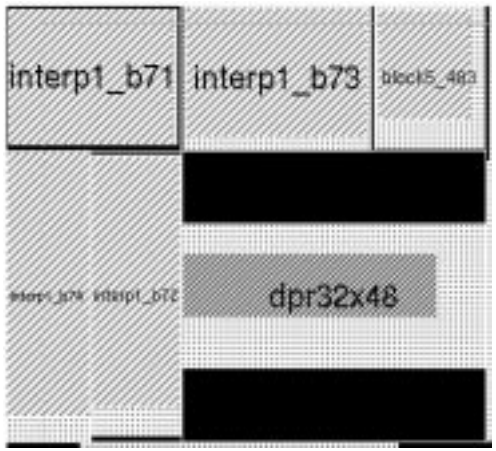
Vergleich der Flächen und Gesamtnetzlängen [Fri96]:

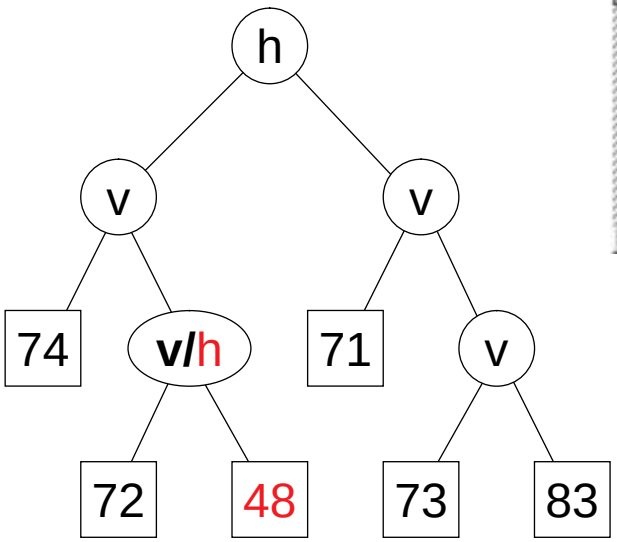
Beispiel	# Zellen	# Netze	$Area_{QP}$: $Area_{TP}$	GNL_{QP} : GNL_{TP}
cebit_1	22 (0)	82	70 : 65	2,0 : 1,7
minproc	15 (0)	46	77 : 75	0,7 : 0,6
backrotation	48 (2)	114	94 : 82	3,0 : 2,0
chip.flex	18 (1)	103	111 : 93	1,7 : 1,5
soar_intern	18 (3)	103	84 : 80	3,1 : 2,8
repa_1167	12 (2)	98	86 : 94	0,5 : 0,9
small.macro	10 (10)	203	104 : 90	1,2 : 0,9
ami33	33 (33)	123	101 : 134	0,16 : 0,18

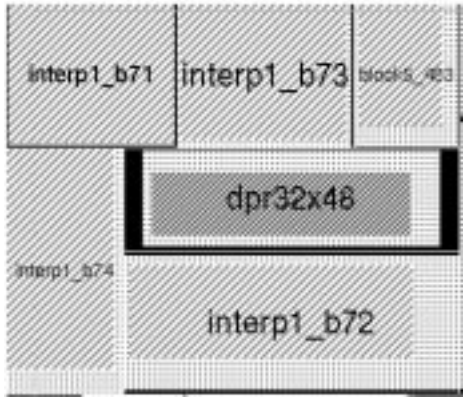
QP = Quadropartitionierung (planner.v5), TP = Terminal-Propagierung (planner.v6)

Area: in % der Flächenvorgabe, GNL in m

ORIENTIERUNG VON SLICINGLINIEN AN MAKROZELLEN







ERGEBNISSE - GLOBAL ROUTING

Qualitative Bewertung bzgl. der Minimierung von MaxDelay:

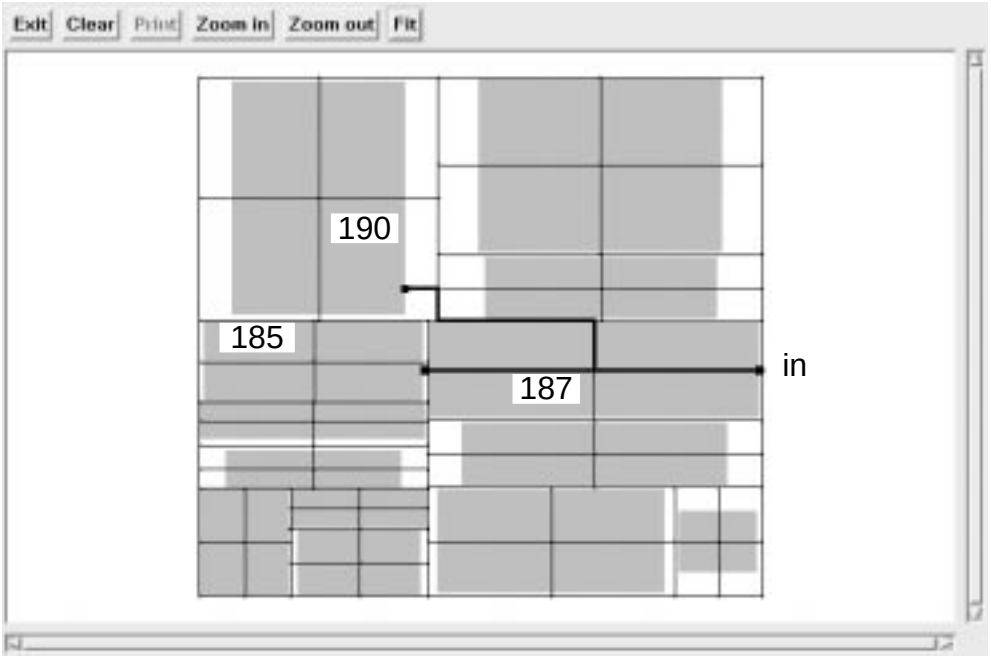
- pdGR-N ist nicht konkurrenzfähig
- pdGR-A weist eine schwankende Ergebnisqualität aus
- pdGR ist fast immer die beste Wahl

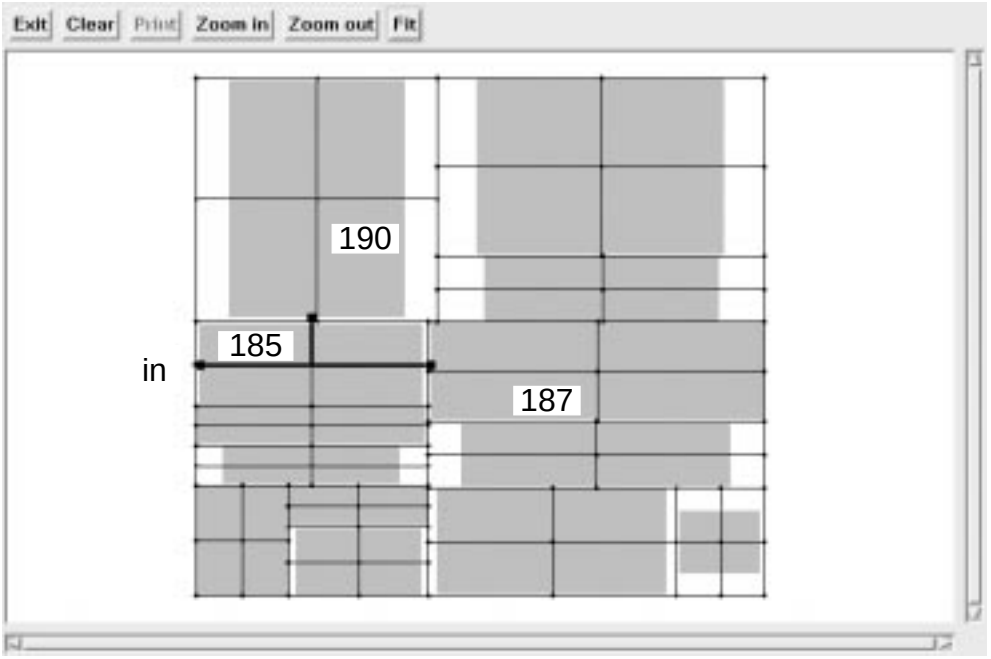
OTC bringt 1,1 - 2,5 % bei MaxDelay, 7,0 - 8,3 % bei Area und 8,2 - 9,7 % bei GNL

Beispielhafter Vergleich von pdGR und pdGR-A mit Durchschnittswerten aus 25 Plazierungen von adr (ES2_0.7):

Algorithmus	mit OTC?	x	y	Area	MaxDelay	GNL
pdGR	nein	99,1	99,0	98,2	14,02	7,84
pdGR-A	nein	99,1	99,2	98,5	14,06	7,93
pdGR	ja	95,3	95,5	91,3	13,87	7,20
pdGR-A	ja	97,9	97,4	95,6	13,91	7,22

VERGLEICH VON PDGR UND PDGR-A AN EINEM SPEZIELLEN BEISPIEL





pdGR: del (in,190) = 6,4 ns

del (in,187) = 0,5 ns

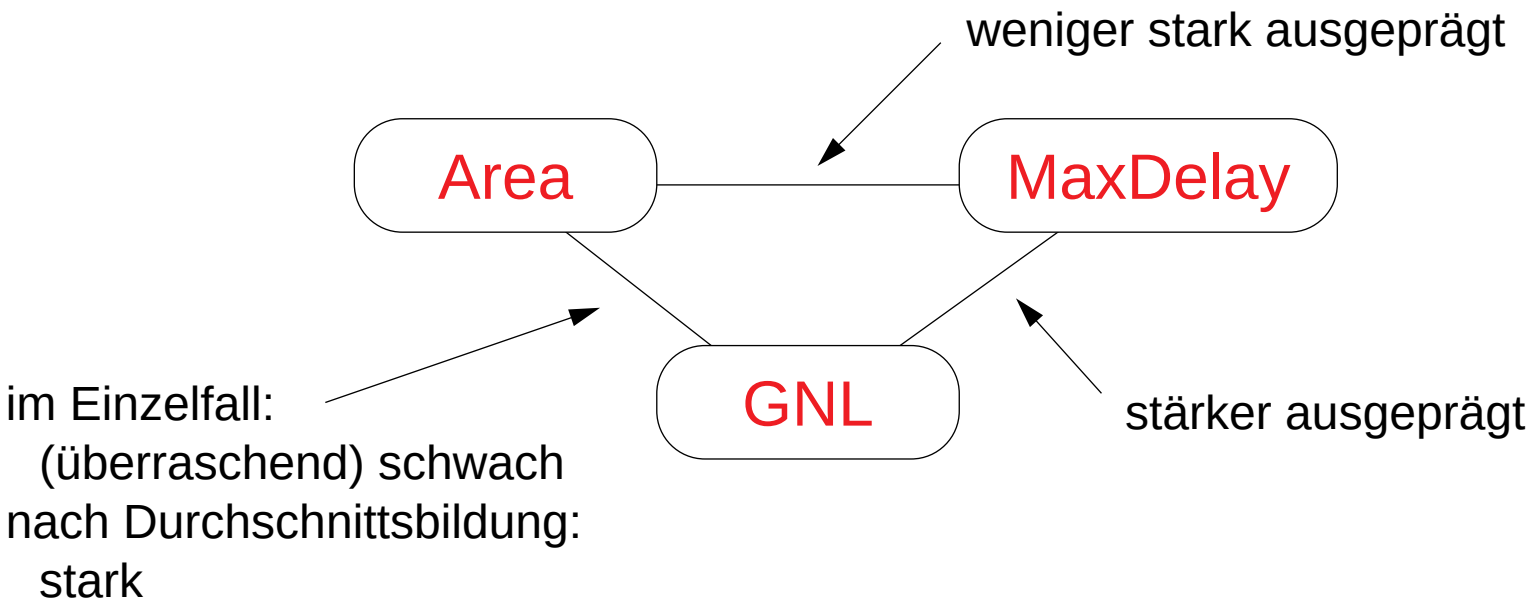
del (in,185) = unkritisch

pdGR-A: del (in,190) = 1,6 ns

del (in,187) = 2,0 ns

del (in,185) = “kritisch”

Gefundene Korrelationen: Theorie vs. Empirie



(*) Beobachtung: Die Kriterien “Gute Formauswahl mit wenig Verschnitt” und “Lokalität stark verbundener Module” widersprechen sich häufig (korrelieren zufällig)

ERGEBNISSE - VARIABILITÄT

Technologie	Beispiel	min (MaxD)	max (MaxD)	%	min (Area)	max (Area)	%
ES2_0.7	adr (OTC)	13,74 ns	13,96 ns	1,6	83,4 %	94,3 %	13,1
“	grp	24,86 ns	25,15 ns	1,2	89,0 %	103,6 %	16,4
“	lon	11,53 ns	11,67 ns	1,2	95,4 %	112,4 %	17,8
SK1	adr	17,02 ns	18,30 ns	7,5	92,2 %	99,4 %	7,8
“	grp	25,32 ns	26,35 ns	4,1	89,2 %	113,7 %	27,5
“	lon	12,70 ns	13,48 ns	6,1	87,0 %	111,2 %	27,8
SK2	adr	4,901 ns	5,315 ns	8,4	91,8 %	99,0 %	7,8
“	grp	8,37 ns	8,50 ns	1,6	88,8 %	112,0 %	26,1
“	lon (OTC)	3,979 ns	4,210 ns	5,8	76,1 %	110,4 %	45,1
SK3	adr	6,700 ns	8,005 ns	19,5	91,7 %	99,0 %	8,0
“	grp (OTC)	8,50 ns	9,04 ns	6,4	77,6 %	101,3 %	30,5
“	lon	4,871 ns	5,248 ns	7,7	83,8 %	110,4 %	31,7
ACMOS4h	minproc	14,77 ns	14,97 ns	1,4	83,4 %	94,1 %	12,8

ZUSAMMENFASSUNG