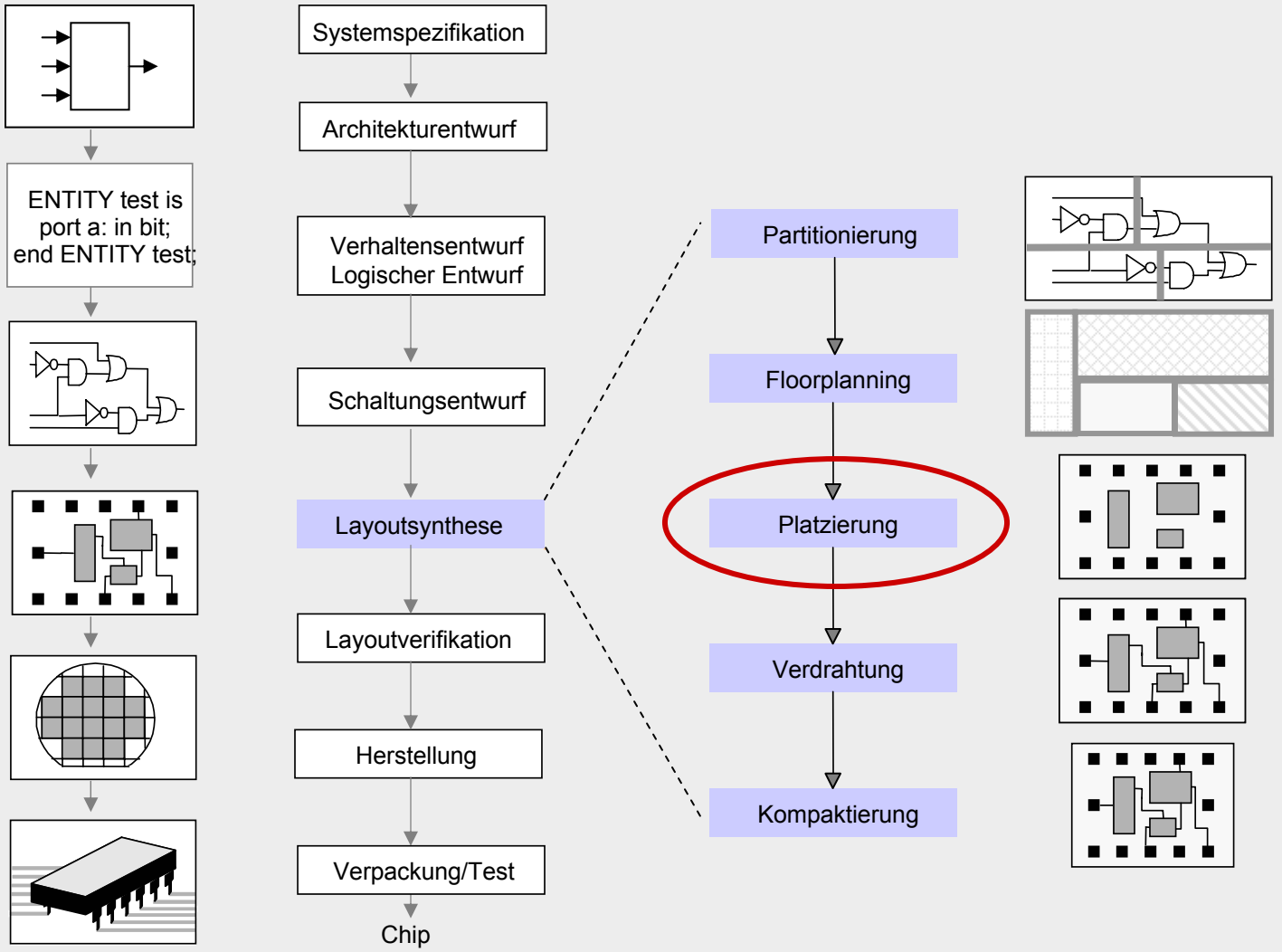
Gliederung Kapitel 4 – Platzierung

4.1 Einführung

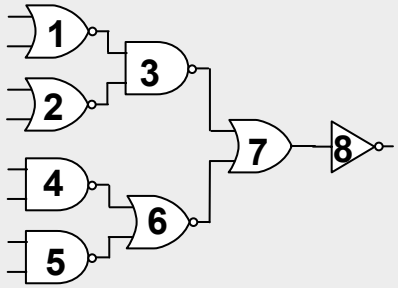


4.1 Einführung

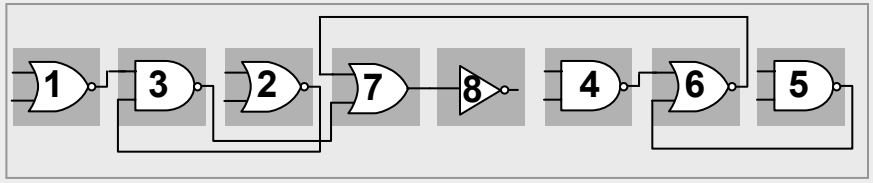
Die Aufgabe der Platzierung ist die Anordnung der einzelnen Schaltungselemente (z.B. Zellen und Bauelemente) auf der zur Verfügung stehenden Layoutfläche unter Berücksichtigung von

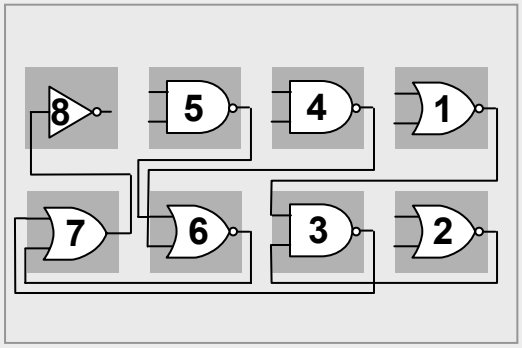
- Randbedingungen (u.a. Überlappungsfreiheit) und
- Optimierungszielen (z.B. minimale Verbindungslänge).

a) Schaltungs-
beispiel

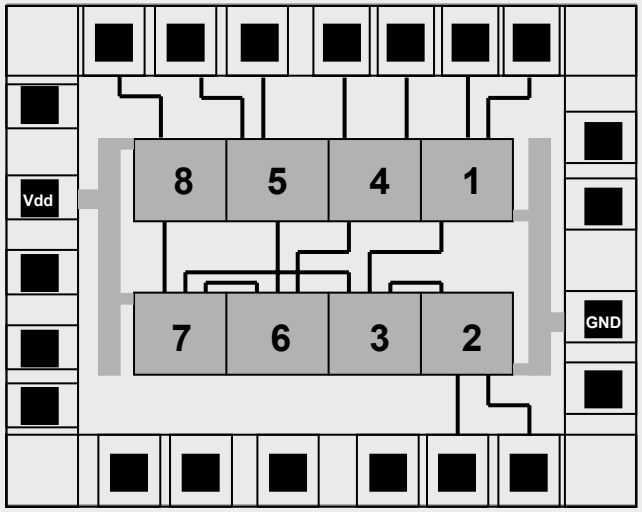


b) Eindimensionale
Platzierung
(Reihenanordnung)

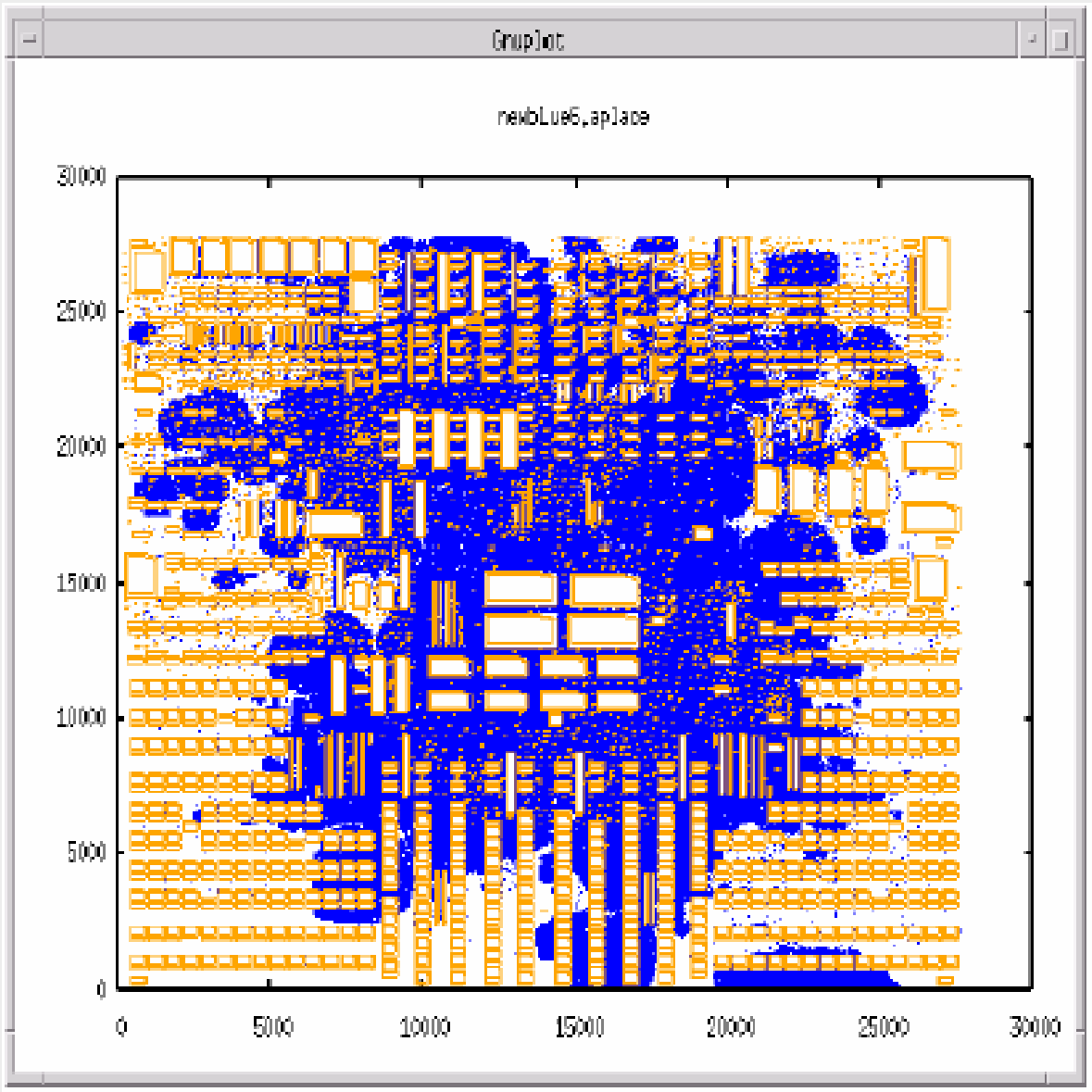


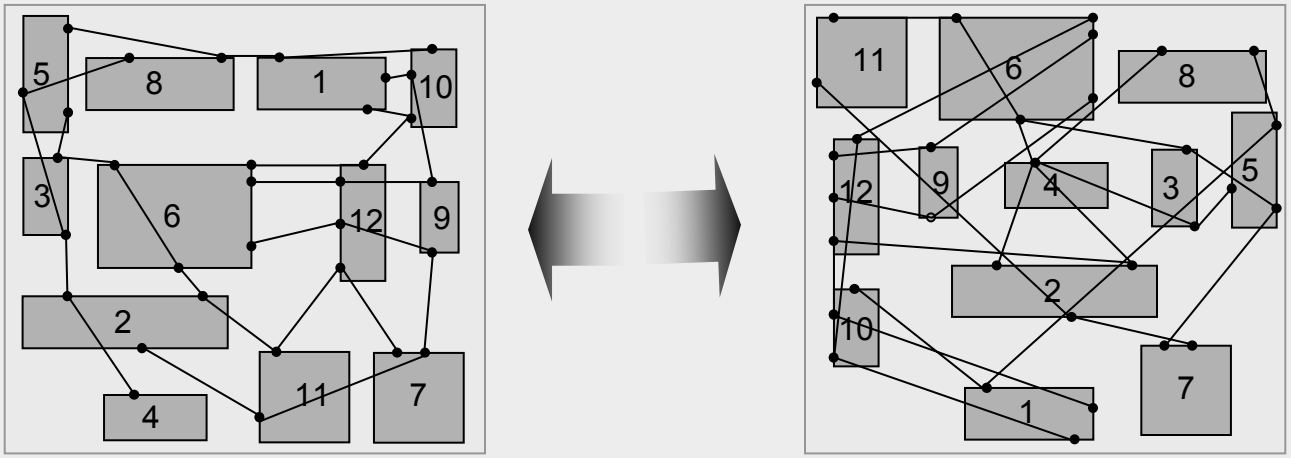


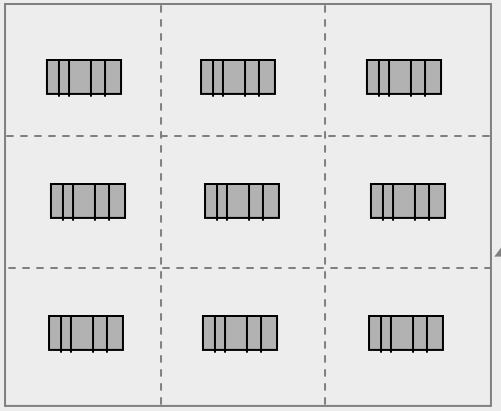
c) Zweireihige Platzierung



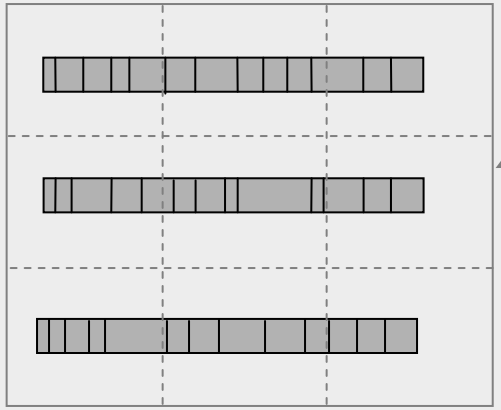
d) Platzierung und Verdrahtung im
Standardzellenlayout


Gnuplot
newblue6.aplace
30000
25000
20000
15000
10000
5000
0
0
5000
10000
15000
20000
25000
30000


5
8
1
10
3
6
12
9
2
11
7
4
11
6
8
12
9
4
3
5
2
10
7
1



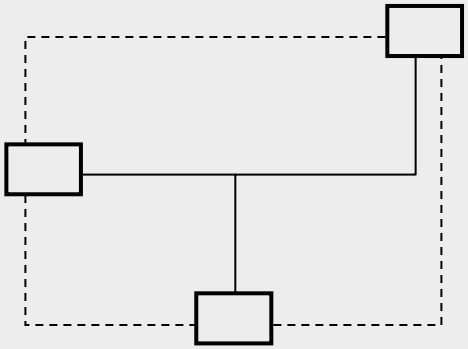
Globale Platzierung:
Zellen werden den Mittelpunkten von Platzierungsbereichen (Tiles, Bins usw.) zugewiesen



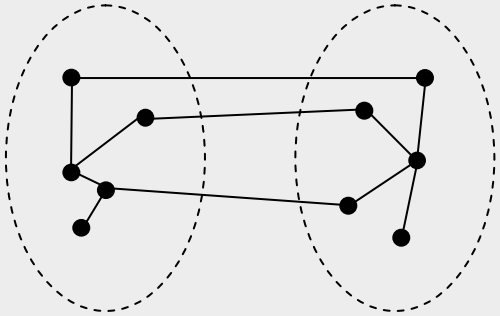
Detailplatzierung / Feinplatzierung:
Zellen werden ohne Überlappungen platziert

Kapitel 4 – Platzierung

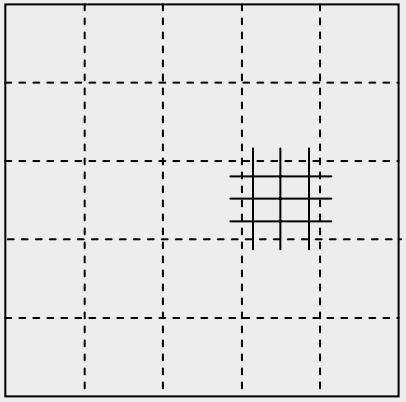
4.2 Optimierungsziele



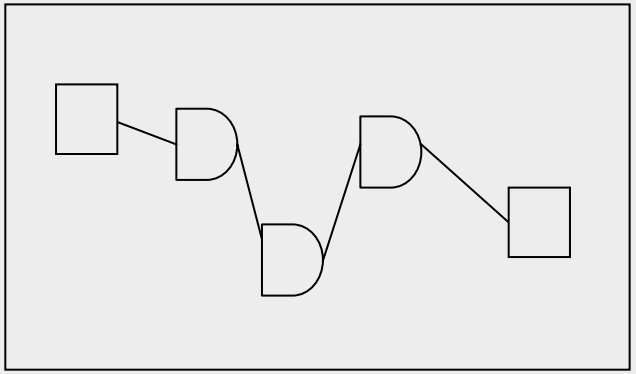
Gesamtverbindungslänge



Anzahl der geschnittenen Netze



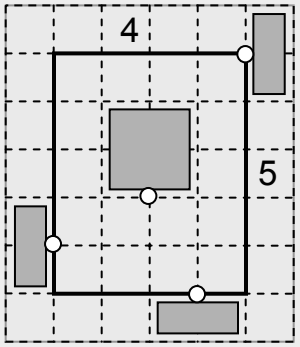
Lokale
Verdrahtungsdichte



Signalverzögerungen

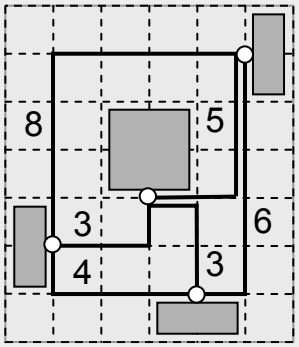
Abschätzung der Verdrahtungslängen bei Mehrpunktnetzen

Halber Umfang des minimal umschließenden Rechtecks (Semi-perimeter method)



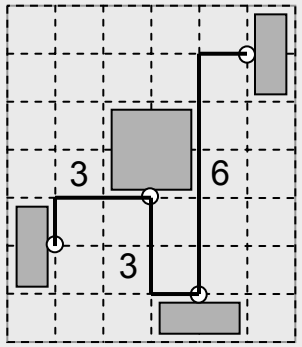
Länge halber Netzumfang = 9

Kompletter Graph (Complete graph)



Länge kompletter Graph * 2 / Pinanzahl = 14,5

Minimale Kette (Minimum chain)

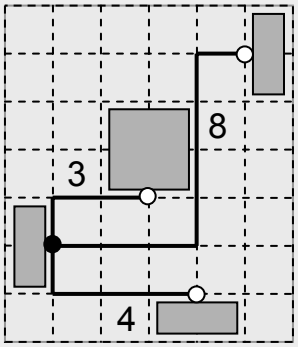


Kettenlänge = 12

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

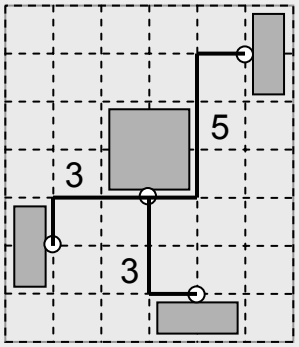
Abschätzung der Verdrahtungslängen bei Mehrpunktnetzen (2)

Quelle-Senken-Verbindung (Source to sink connection)



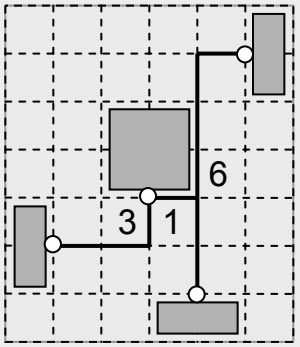
Quelle-Senken-Länge= 15

Minimaler rektilinearer Spannbaum (Minimum rectilinear spanning tree)



Spannbaum-Länge = 11

Steinerbaum-Abschätzung (Steiner tree approximation)

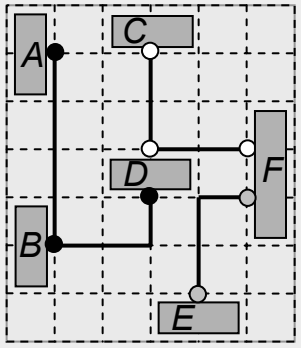


Steinerbaum-Länge = 10

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

4.2.1 Optimierungsziel: Gewichtete Gesamtverbindungslänge – Beispiel

Netze	Gewicht
$N_1 = (A, B, D_1)$	$w_1 = 2$
$N_2 = (C, D_2, F_1)$	$w_1 = 4$
$N_3 = (F_2, E)$	$w_1 = 1$



$$L(P) = \sum_{n \in N} w_n \cdot d_n = 2 \cdot 7 + 4 \cdot 4 + 1 \cdot 3 = 33$$

4.2.2 Optimierungsziel: Anzahl der geschnittenen Netze – Beispiel

Netze

$N_1 = (A, B, D_1)$

$N_2 = (C, D_2, F_1)$

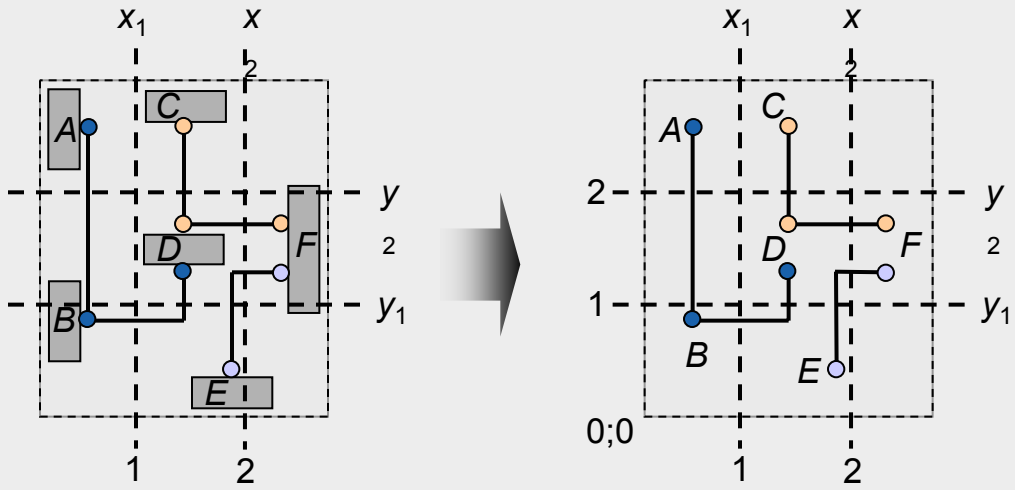
$N_3 = (F_2, E)$

$\psi_P(x_1) = 1$

$\psi_P(x_2) = 2$

$\psi_P(y_1) = 3$

$\psi_P(y_2) = 2$



4.2.2 Optimierungsziel: Anzahl der geschnittenen Netze – Beispiel

Netze

$N_1 = (A, B, D_1)$

$N_2 = (C, D_2, F_1)$

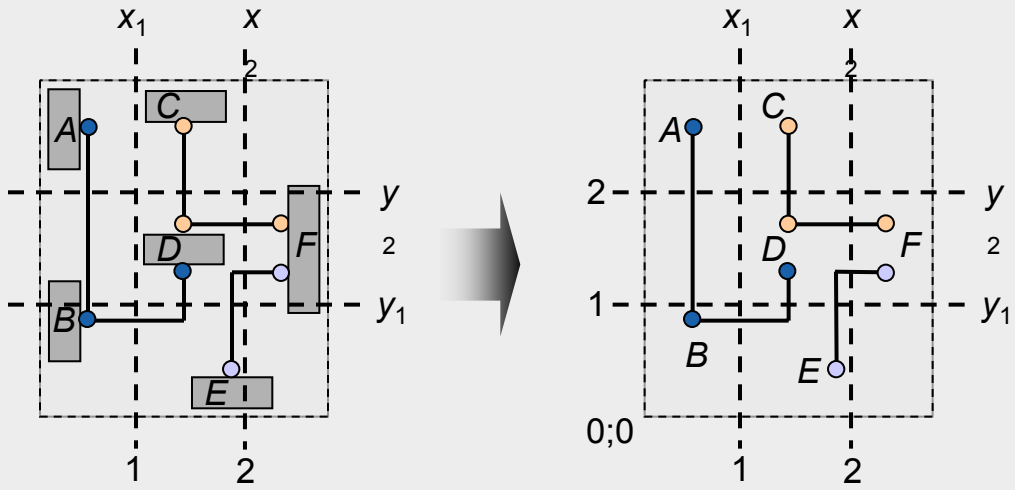
$N_3 = (F_2, E)$

$\psi_P(x_1) = 1$

$\psi_P(x_2) = 2$

$\psi_P(y_1) = 3$

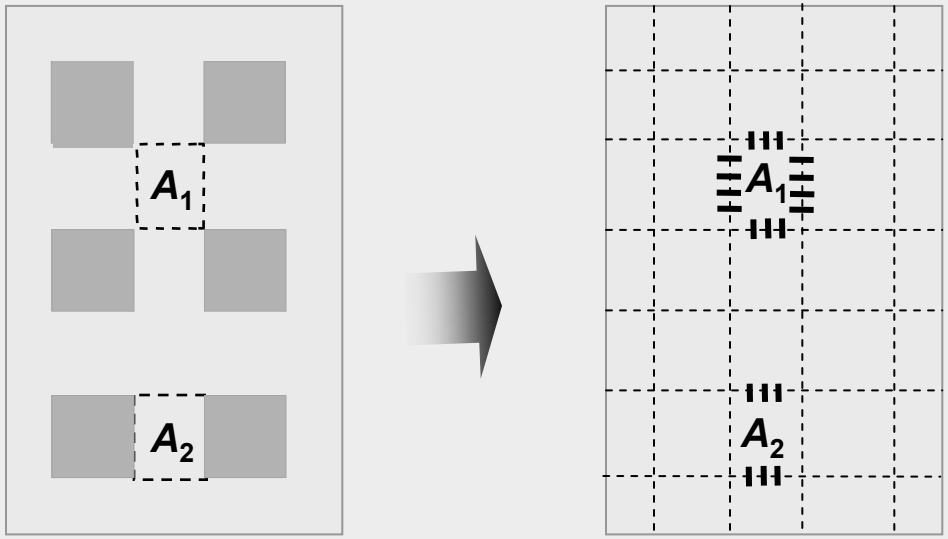
$\psi_P(y_2) = 2$



Zielfunktion (Gesamtverbindungslänge):

$$L(P) = \sum_i \psi_P(x_i) + \sum_j \psi_P(y_j) = \psi_P(x_1) + \psi_P(x_2) + \psi_P(y_1) + \psi_P(y_2) = 1 + 2 + 3 + 2 = 8$$

4.2.3 Optimierungsziel: Lokale Verdrahtungsdichte



4.2.3 Optimierungsziel: Lokale Verdrahtungsdichte – Beispiel

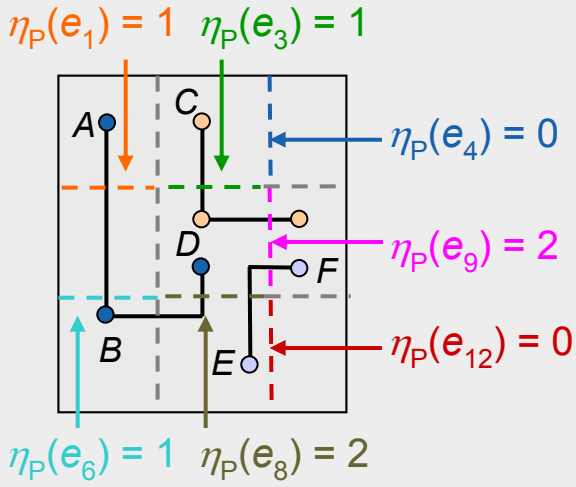
Netze

$N_1 = (A, B, D_1)$

$N_2 = (C, D_2, F_1)$

$N_3 = (F_2, E)$

Jede Kante besitzt eine Kapazität $\phi_P(e_i) = 3$.



4.2.3 Optimierungsziel: Lokale Verdrahtungsdichte – Beispiel

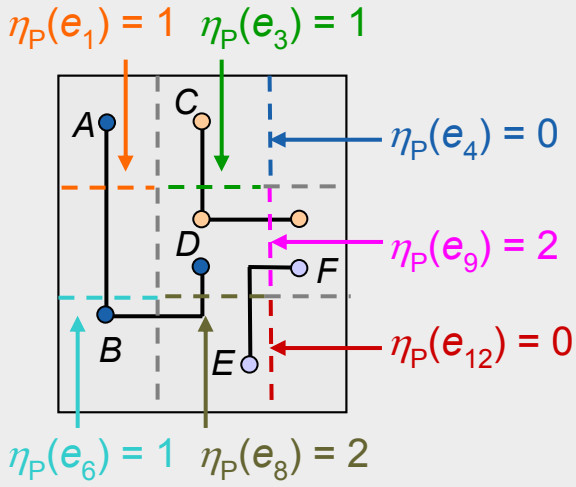
Netze

$N_1 = (A, B, D_1)$

$N_2 = (C, D_2, F_1)$

$N_3 = (F_2, E)$

Jede Kante besitzt eine Kapazität $\phi_P(e_i) = 3$.



Maximum $\eta_P(e_i) = 2$, $\phi_P(e_i) = 3$, womit $D(P) = 2/3$, d.h. $D(P) \leq 1$, die Platzierung P also einfach zu verdrahten sein sollte.

Kapitel 4 – Platzierung

4.3 Platzierungsalgorithmen

- Partitionierende Algorithmen (rekursive Algorithmen):
  - Optimierung der Platzierungsanordnung mittels rekursiver und dabei immer feinerer Partitionierung der Netzliste und des Platziergebiets
  - Nutzung von Graphenpartitionierern
  - Beispiel: Min-Cut-Platzierung

- Analytische Vorgehensweisen:
  - Nutzung von mathematischen Methoden (z.B. lineare Gleichungssysteme) zur Abbildung und Optimierung des Platzierungsproblems
  - Beispiel: Quadratische Platzierung

- Stochastische Algorithmen:
  - Mit Hilfe von stochastischen Methoden wird das Minimum einer beliebigen Kostenfunktion gesucht
  - Einbeziehung von Zufallsentscheidungen, womit bei gleicher Aufgabenstellung unterschiedliche Lösungen erzeugt werden
  - Beispiel: Simulated Annealing

4.3 Platzierungsalgorithmen

Partitionierend

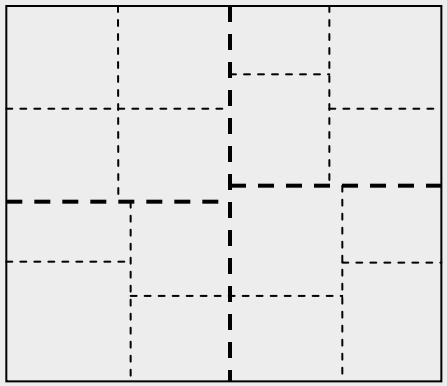
Analytisch

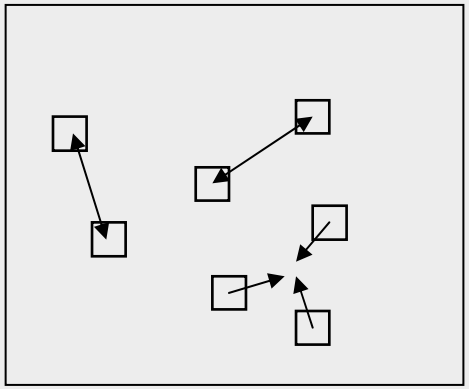
Stochastisch

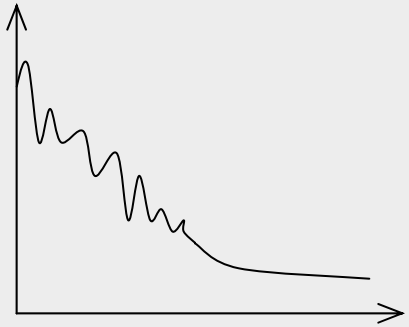
Min-Cut-Platzierung

Quadratische Platzierung

Platzierung mit SA





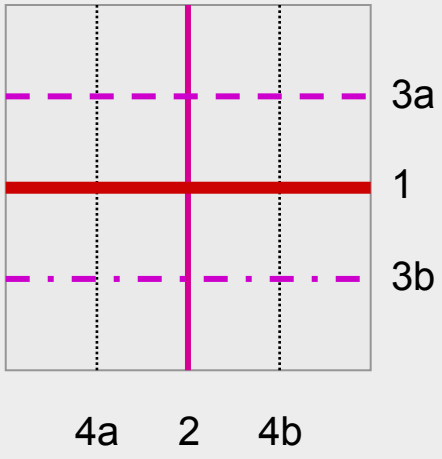


4.3.1 Min-Cut-Platzierung

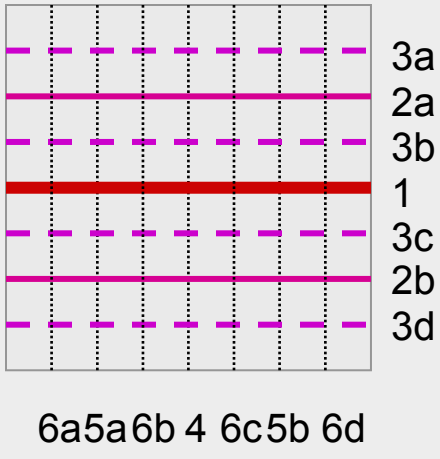
- Platzierungsfläche sequentiell mit Schnittlinien durchzogen, bis die Schnittflächen so klein sind, dass sie nur noch wenige/eine Zelle einschließen
- Bei jedem Schnitt werden die Zellen z.B. so auf die beiden entstehenden Teilflächen aufgeteilt, dass am Ende die Anzahl der die Schnittlinien c_r kreuzenden Netze $\psi_P(c_r)$ minimiert ist
- Algorithmen zur Minimierung von $\psi_P(c_r)$ sind oft der Kernighan-Lin-Algorithmus (KL-Algorithmus) sowie der Fiduccia-Mattheyses-Algorithmus (FM-Algorithmus)

4.3.1 Min-Cut-Platzierung

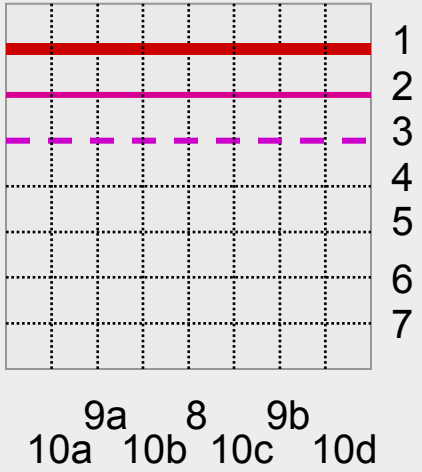
Quadratur-Platzierung
(Quadrature placement)



Halbierungs-Platzierung
(Bisection placement)



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



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

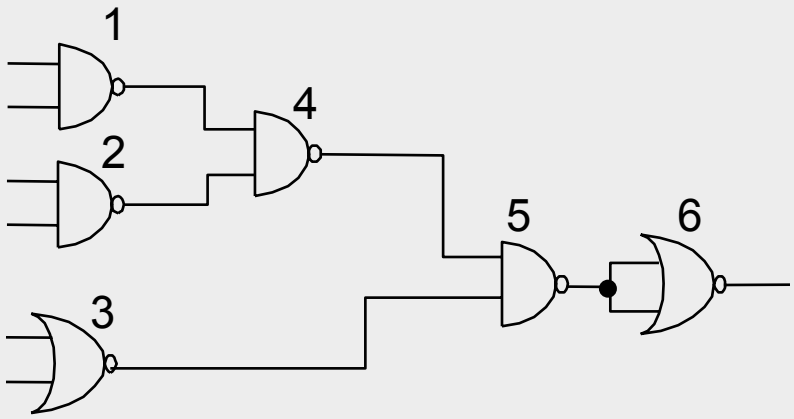
4.3.1 Min-Cut-Platzierung

Min-Cut-Algorithmus (Quadratur Platzierung)

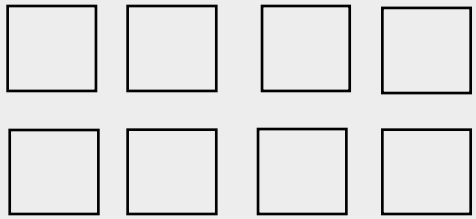
1. Aufteilung der Layoutfläche in zwei Teilflächen mit senkrechter oder horizontaler Schnittrichtung.
2. Anwendung eines geeigneten Algorithmus, z.B. des KL- oder FM-Algorithmus, zur optimierten Verteilung der Zellen auf die beiden Teilflächen.
3. Aufteilung in neue Teilflächen und jeweils Initialzuordnung der Zellen auf diese. Alternierender Wechsel zwischen senkrechter und horizontaler Schnittrichtung.
4. ENDE, falls jede Teilfläche genau eine Zelle enthält, sonst weiter mit Schritt 2.

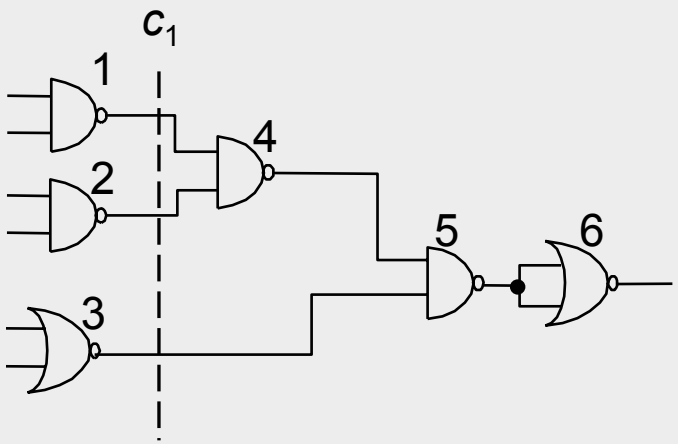
4.3.1 Min-Cut-Platzierung: Beispiel

Gegeben:

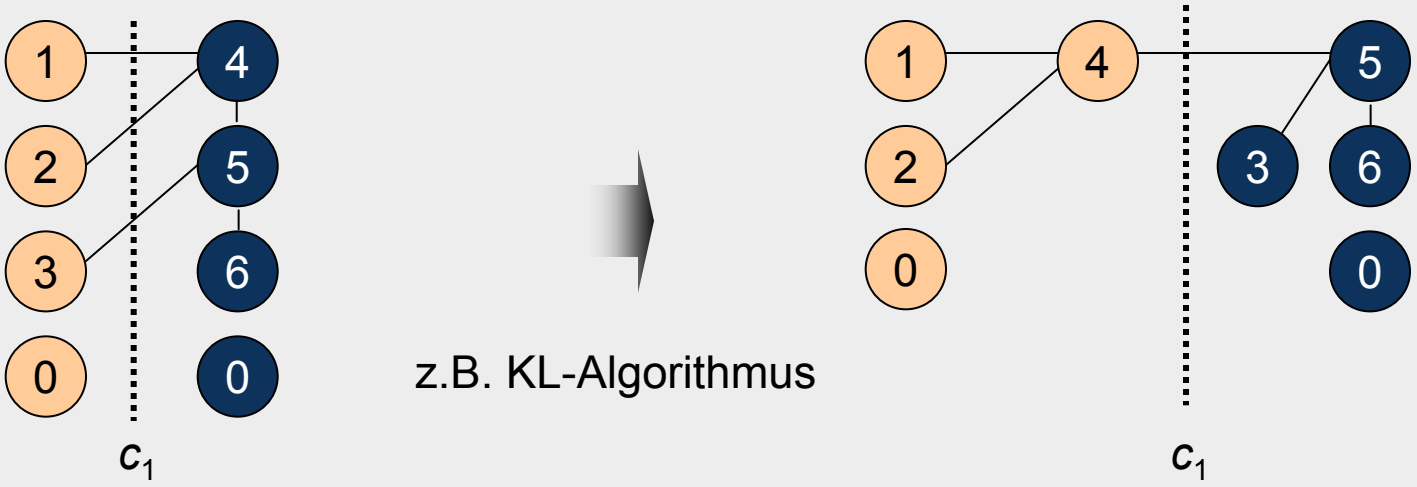


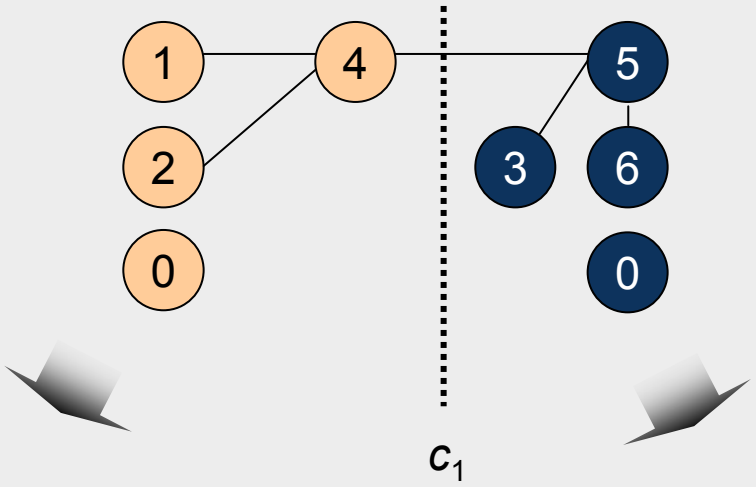
Gesucht: 4 x 2 Platzierung mit minimaler Netzlänge





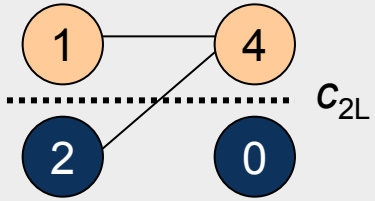
Vertikaler Initialschnitt c_1: L={1,2,3}, R={4,5,6}

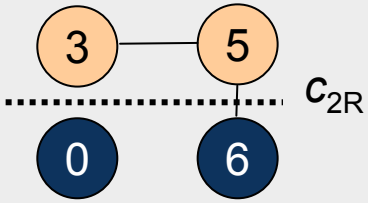


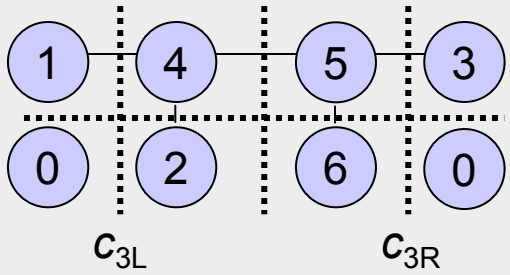


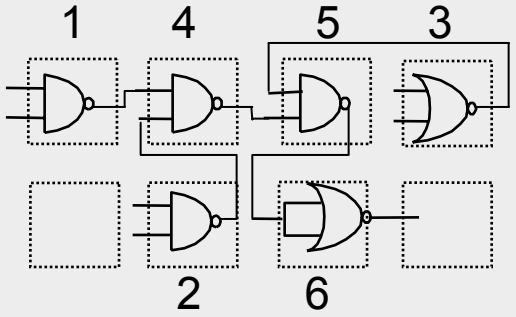
Horizontaler Schnitt c_{2L}: T={1,4}, B={2,0}

Horizontaler Schnitt c_{2R}: T={3,5}, B={6,0}





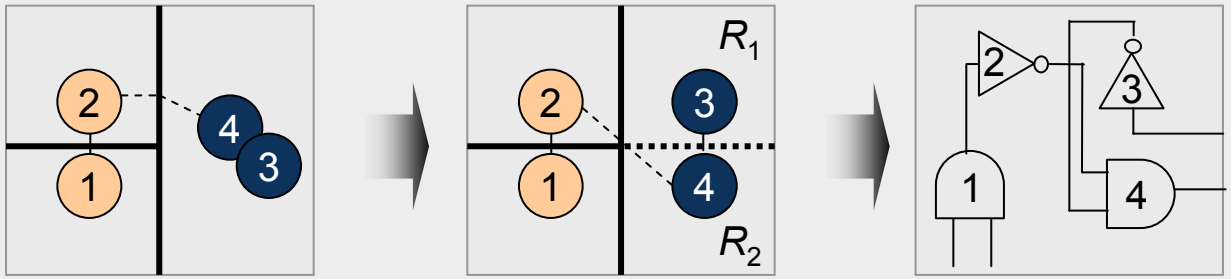




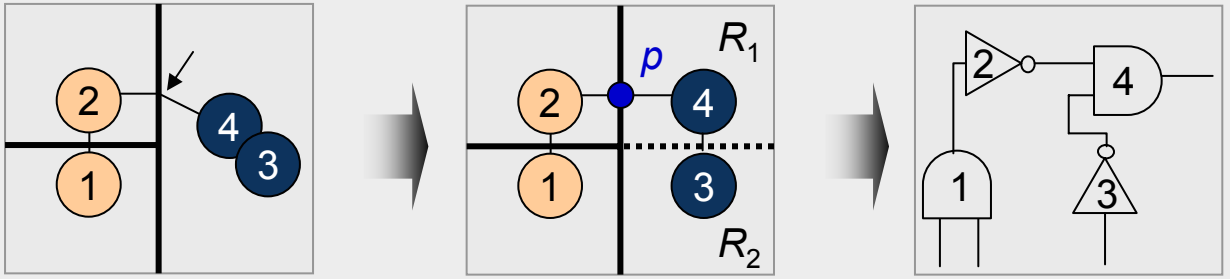
4.3.1 Min-Cut-Platzierung

- Vorteile:
  - Sehr schnell
  - Kostenfunktion kann beliebig erweitert werden, d.h. auch für Timing-driven Placement anwendbar
  - Von Natur aus hierarchisch, daher für große Schaltungen nutzbar
- Nachteile:
  - Viele Verschiebungen ohne Auswirkungen
  - Oft Zufallsfaktoren eingeschlossen, daher nicht immer deterministisch
  - Unterhalb bestimmter Partitionsgröße andere Ansatz zur Platzierung sinnvoll
  - Nur sequentielle Optimierung, d.h. die Optimierung bezieht sich immer nur auf die Zuordnung zur jeweils betrachteten Schnittlinie

4.3.2 Min-Cut-Platzierung mit Anschlussfestlegung

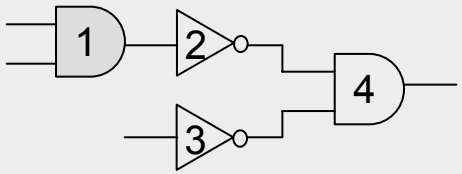


Mit Anschlussfestlegung (Terminal Propagation)



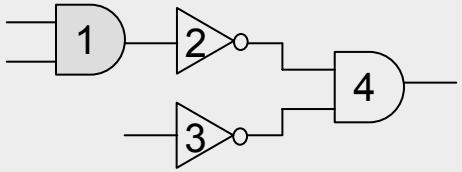
4.3.2 Min-Cut-Platzierung mit Anschlussfestlegung: Beispiel

Vier Zellen einer Schaltung sind auf ein 2x2-Feld aufzuteilen, wobei Min-Cut-Platzierung mit Anschlussfestlegung anzuwenden sind.

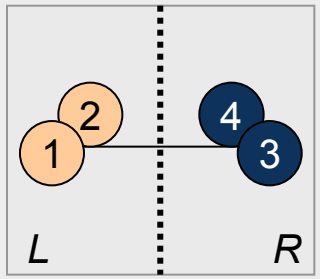


4.3.2 Min-Cut-Platzierung mit Anschlussfestlegung: Beispiel

Vier Zellen einer Schaltung sind auf ein 2x2-Feld aufzuteilen, wobei Min-Cut-Platzierung mit Anschlussfestlegung anzuwenden sind.

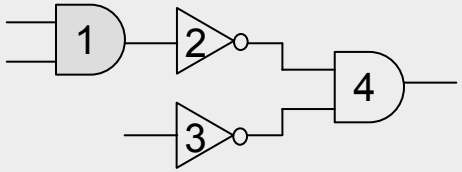


1. Partitionierung in *L* und *R*. Partitionierungskosten = 1.

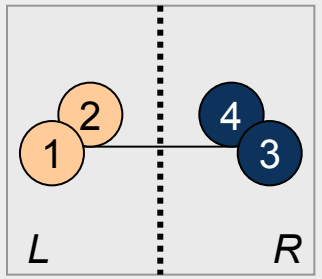


4.3.2 Min-Cut-Platzierung mit Anschlussfestlegung: Beispiel

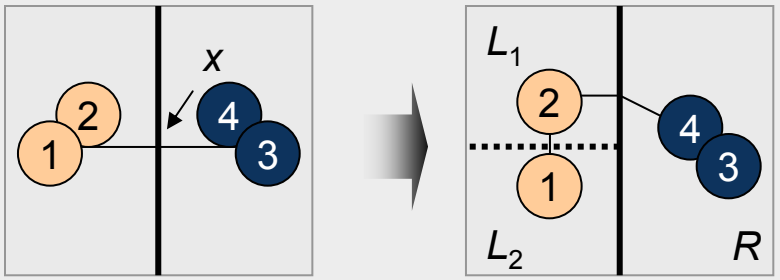
Vier Zellen einer Schaltung sind auf ein 2x2-Feld aufzuteilen, wobei Min-Cut-Platzierung mit Anschlussfestlegung anzuwenden sind.



1. Partitionierung in *L* und *R*. Partitionierungskosten = 1.

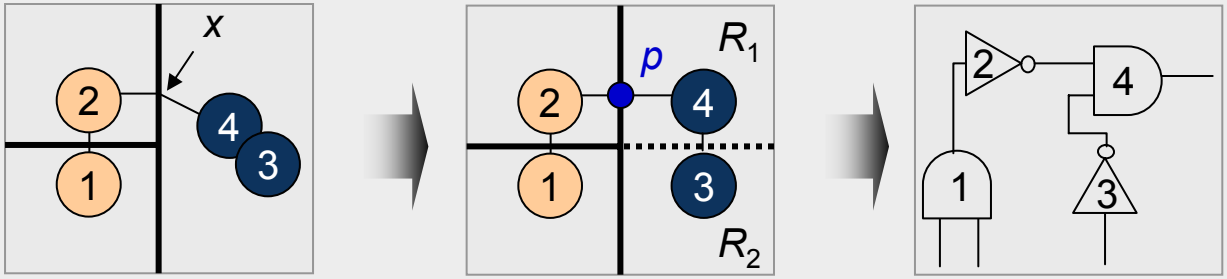


2. Partitionierung in L_1 und L_2 ohne Beeinflussung durch *R*, da der Schnittpunkt *x* nahe der Partitionierungslinie liegt (Zellen werden jeweils im Mittelpunkt ihrer Regionen angenommen).



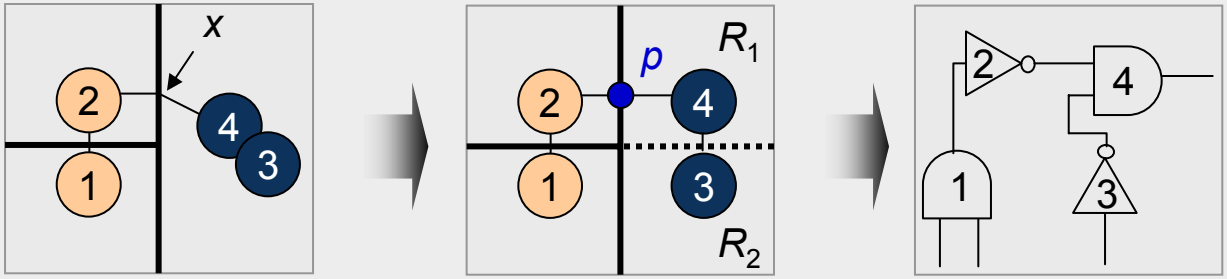
4.3.2 Min-Cut-Platzierung mit Anschlussfestlegung: Beispiel

3. Partitionierung in R_1 und R_2 mit Beeinflussung durch L, da Schnittpunkt x nicht in „Nähe“ der Partitionierungslinie liegt. Erzeugen einer Dummy-Zelle p, womit Zelle 4 in R_1 und Zelle 3 in R_2 liegen müssen, um Partitionierungskosten R_1-R_2 von 1 zu erhalten.

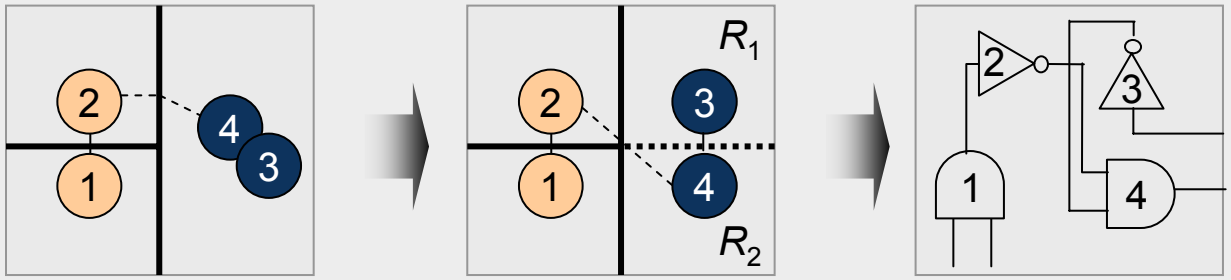


4.3.2 Min-Cut-Platzierung mit Anschlussfestlegung: Beispiel

3. Partitionierung in R_1 und R_2 mit Beeinflussung durch L, da Schnittpunkt x nicht in „Nähe“ der Partitionierungslinie liegt. Erzeugen einer Dummy-Zelle p, womit Zelle 4 in R_1 und Zelle 3 in R_2 liegen müssen, um Partitionierungskosten R_1-R_2 von 1 zu erhalten.



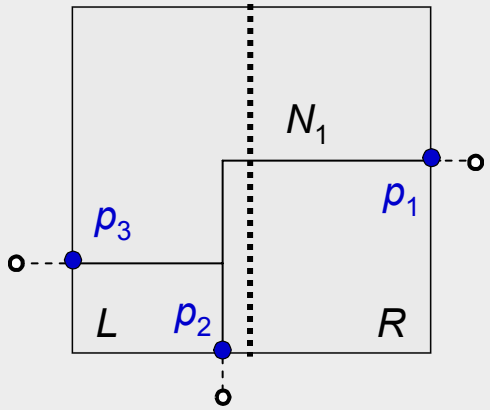
Hinweis: Ohne Anschlussfestlegung wären mit Zelle 3 in R_1 und 4 in R_2 anfänglich ebenfalls Partitionierungskosten R_1-R_2 von 1 erzielt worden, die sich jedoch später, d.h. unter Berücksichtigung der Platzierung in L, als Kostenwert 2 herausgestellt hätten.



4.3.2 Min-Cut-Platzierung mit Anschlussfestlegung: Beispiel

Externe Anschlüsse

1. Erstellen eines Steinerbaums für die drei externen Anschlüsse von Netz N_1.
2. Ermitteln der sich aus den Schnittstellen ergebenden Dummy-Zellen p_1, p_2, p_3.
3. Die Dummy-Zellen werden in den nachfolgenden Partitionierungsschritten als Zellen von N_1 behandelt, deren Lage bzgl. der jeweiligen Partitionen (hier: *L* und *R*) nicht veränderbar ist.



Hinweis: Bei der Aufteilung in *L* und *R* (vertikaler Schnitt) wird p_2 ignoriert, da es sich in der Nähe der Schnittlinie befindet. Analog würde bei einem horizontalen Schnitt p_1 ignoriert.

4.3.3 Quadratische Platzierung

- Euklidische Verbindungslänge geht quadratisch in Kostenfunktion ein:

$$L(P) = \frac{1}{2} \sum_{i,j=1}^{n} c_{ij} \left(\left(x_i - x_j \right)^2 + \left(y_i - y_j \right)^2 \right)$$

- Netze dazu in Zweipunkt-Verbindungen zerlegt, Kostenfunktion ist die Summe der gewichteten quadratischen Abstände, Minimierung dieser Summe
- Analogie: Federmodell
  - jede quadratische Zweipunkt-Länge entspricht Energie einer Feder zwischen beiden Punkten (Energie einer Feder ist proportional zum Quadrat ihrer Auslenkung)
  - quadratische Kostenfunktion verkörpert Gesamtenergie des Federsystems; deren Ableitung ist die Gesamtkraft des Systems
  - System sucht Zustand minimaler Energie, also minimale Summe der Abstandsquadrate
  - damit Zellen im Kräftegleichgewicht hinsichtlich der die Verdrahtung repräsentierenden Kräfte

4.3.3 Quadratische Platzierung

- Euklidische Verbindungslänge geht quadratisch in Kostenfunktion ein:

$$L(P) = \frac{1}{2} \sum_{i,j=1}^{n} c_{ij} \left(\left(x_i - x_j \right)^2 + \left(y_i - y_j \right)^2 \right)$$

- Vorgehensweise:
  - Vektor-/Matrixschreibweise: $L(P) = \frac{1}{2} \left[X^{\mathrm{T}} C X + Y^{\mathrm{T}} C Y \right] + X^{\mathrm{T}} K_{\mathrm{x}} + Y^{\mathrm{T}} K_{\mathrm{y}} + k$

    mit X^{T} und Y^{T} Vektoren der Dimension n der x- bzw. y-Koordinaten der n Zellen, C Verbindungsmatrix, K_{x} und K_{y} Koordinatenvektoren der nicht verschiebbaren Zellen/Außenanschlüsse sowie Konstante k
  - Globales Minimum und damit platzierungsoptimale x- und y-Koordinaten der Zellen lassen sich durch partielle Ableitung von $L(P)$ bestimmen:

$$\frac{\partial L(P)}{\partial X} = CX + K_{\mathrm{x}} = 0 \text{ und } \frac{\partial L(P)}{\partial Y} = CY + K_{\mathrm{y}} = 0$$

- Ergebnis: viele Zellenüberlappungen

4.3.3 Quadratische Platzierung

- Quadratische Platzierung lässt sich weiter unterteilen, je nachdem, wie die Zellenüberlappungen beseitigt/vermieden werden

Kraftbasierte quadratische Platzierung

Überlappungsfreiheit:
zusätzliche Kräfte

Quadratische Platzierung mit Schwerpunktsnebenbedingungen

Überlappungsfreiheit:
Verfeinerung von Schwerpunktsnebenbedingungen (führt zu „Quadratic Programming“) und damit rekursive Zerteilung in Platziergebiete

4.3.3 Quadratische Platzierung

- Vorteile:
  - Schnelle analytische Lösung
  - Auch für große Problemgrößen geeignet
- Nachteile:
  - Pads notwendig (liefert triviale Lösung ohne Pads)
  - Hierarchischer Ansatz schwierig zu realisieren

4.3.4 Kräfteplatzierung mittels ZFT-Position

Kräfteplatzierung

- Bei der Kräfteplatzierung werden die zu platzierenden Zellen analog einem mechanischen System aus mit Federn verbundenen Körpern betrachtet.
- Dabei üben miteinander verbundenen Körper (Zellen) eine Anziehungskraft zueinander aus, wobei diese Kraft direkt proportional zur Entfernung zwischen den Körpern ist.
- Ergebnis: Kräftegleichgewicht bzw. energieminimaler Zustand

4.3.4 Kräfteplatzierung mittels ZFT-Position

Kräfteplatzierung und quadratische Platzierung

- Energie einer Feder ist proportional zum Quadrat ihrer Auslenkung
- Ermittlung der energieminimalen Positionen von Zellen, die mit Federn verbunden sind, ist daher identisch zur Minimierung der Summe der Quadrate der euklidischen Abstände (quadratische Platzierung)
- Unterscheidung in der Zellenbetrachtung: Während bei der quadratischen Platzierung die Zellenplatzierungen durch gleichzeitige Berücksichtigung aller Zellen ein einem aus der quadratischen Kostenfunktion abgeleiteten Gleichungssystem ermittelt werden (Überlappungen!), erfolgt die Zellenplatzierung bei der Kräfteplatzierung mittels ZFT-Position durch sequentielle Zellenverschiebungen

4.3.4 Kräfteplatzierung mittels ZFT-Position

- Angenommen, eine Zelle *a* ist mit einer Zelle *b* verbunden. Die Anziehungskraft zwischen beiden Zellen ergibt sich aus dem Produkt von Wichtung und Länge der Verbindung

$$\vec{F} = w_{ab} \cdot \vec{d}_{ab}$$

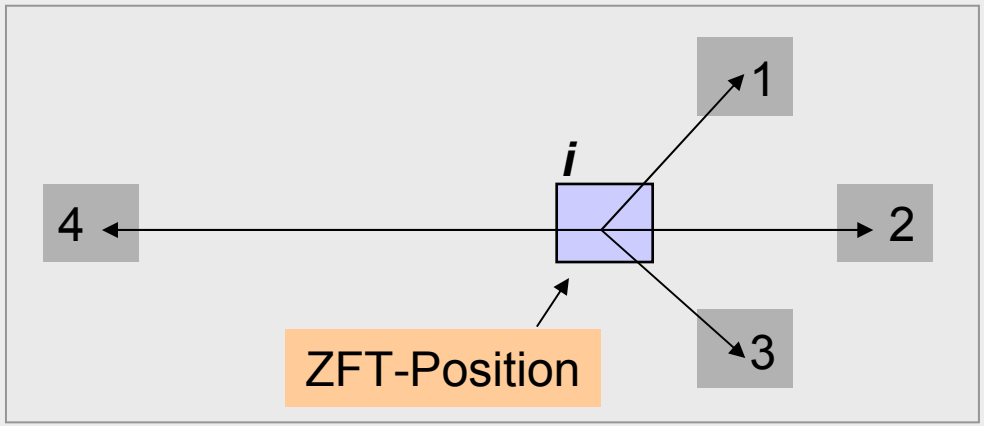
(bzw. $F = w_{ab} \cdot d_{ab}$, da $\vec{F}$ parallel $\vec{d}_{ab}$ ist).

- Analog gilt für eine Zelle *i*, die mit mehreren Zellen 1 … *j* verbunden ist

$$\vec{F}_i = \sum_j \left(w_{ij} \cdot \vec{d}_{ij} \right)$$

wobei w_{ij} die Wichtung der Verbindung und d_{ij} deren Länge sind.

Zero-Force-Target (ZFT)-Position einer Zelle *i*



$$\vec{F}_i = w_{i1} \cdot \vec{d}_{i1} + w_{i2} \cdot \vec{d}_{i2} + w_{i3} \cdot \vec{d}_{i3} + w_{i4} \cdot \vec{d}_{i4} \rightarrow \min$$

4.3.4 Kräfteplatzierung mittels ZFT-Position

Anwendung bei der Platzierung:

- Für jede Zelle werden die auf sie wirkenden Kräfte berechnet, um diese Zelle *i* dann in ihrer jeweiligen ZFT-Position (x_i^0, y_i^0) zu platzieren.
- Diese lässt sich ermitteln, indem die in *x*- und in *y*-Richtung wirkenden Kräfte zu Null gesetzt werden:

$$\sum_j w_{ij} \cdot \left(x_j^0 - x_i^0\right) = 0 \qquad \sum_j w_{ij} \cdot \left(y_j^0 - y_i^0\right) = 0$$

- Die Umstellung dieser Gleichungen nach x_i^0 und y_i^0 liefert

$$x_i^0 = \frac{\sum_j w_{ij} \cdot x_j}{\sum_j w_{ij}} \qquad y_i^0 = \frac{\sum_j w_{ij} \cdot y_j}{\sum_j w_{ij}}$$

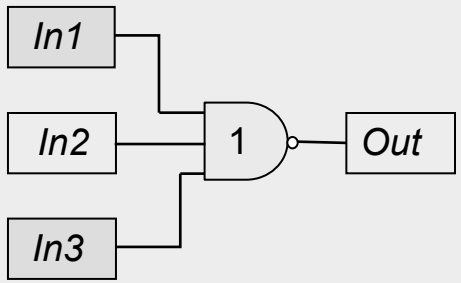
Berechnung der ZFT-Position einer Zelle *i*, welche mit den Zellen 1 … *j* verbunden ist

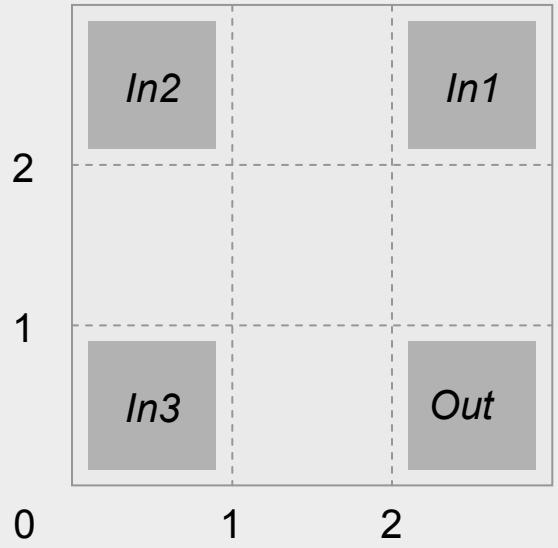
4.3.4 Kräfteplatzierung mittels ZFT-Position: Beispiel (ZFT-Position)

Gegeben:

- Schaltung mit einer NAND-Zelle 1 und vier I/O-Padzellen auf einem 3 x 3 Raster
- Positionen der Padzellen: *In1* (2;2), *In2* (0;2), *In3* (0;0), *Out* (2;0)
- Wichtung der einzelnen Verbindungen: $w_{1In1} = 8$, $w_{1In2} = 10$, $w_{1In3} = 2$, $w_{1Out} = 2$

Gesucht: ZFT-Position der Zelle 1 (Rasterposition)





4.3.4 Kräfteplatzierung mittels ZFT-Position: Beispiel (ZFT-Position)

Gegeben:

- Positionen der Padzellen: *In1* (2;2), *In2* (0;2), *In3* (0;0), *Out* (2;0)
- Wichtung der einzelnen Verbindungen: $w_{1In1} = 8$, $w_{1In2} = 10$, $w_{1In3} = 2$, $w_{1Out} = 2$

Lösung:

$$x_1^0 = \frac{\sum_j w_{1j} \cdot x_j}{\sum_j w_{1j}} = \frac{w_{1In1} \cdot x_{In1} + w_{1In2} \cdot x_{In2} + w_{1In3} \cdot x_{In3} + w_{1Out} \cdot x_{Out}}{w_{1In1} + w_{1In2} + w_{1In3} + w_{1Out}} = \frac{8 \cdot 2 + 10 \cdot 0 + 2 \cdot 0 + 2 \cdot 2}{8 + 10 + 2 + 2} = \frac{20}{22} \approx 0{,}9$$

$$y_1^0 = \frac{\sum_j w_{1j} \cdot y_j}{\sum_j w_{1j}} = \frac{w_{1In1} \cdot y_{In1} + w_{1In2} \cdot y_{In2} + w_{1In3} \cdot y_{In3} + w_{1Out} \cdot y_{Out}}{w_{1In1} + w_{1In2} + w_{1In3} + w_{1Out}} = \frac{8 \cdot 2 + 10 \cdot 2 + 2 \cdot 0 + 2 \cdot 0}{8 + 10 + 2 + 2} = \frac{36}{22} \approx 1{,}6.$$

4.3.4 Kräfteplatzierung mittels ZFT-Position: Beispiel (ZFT-Position)

Gegeben:

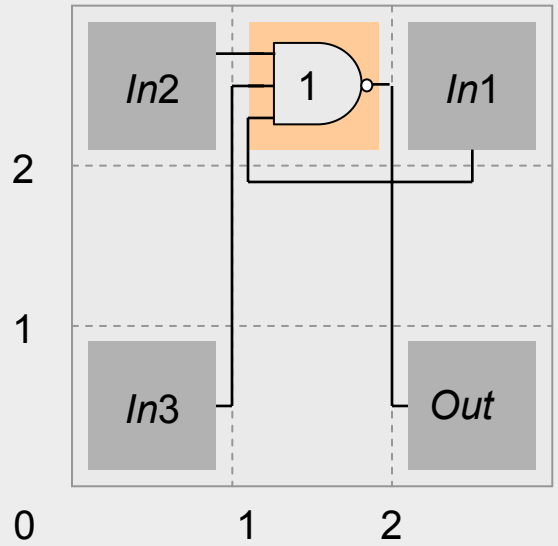
- Positionen der Padzellen: *In1* (2;2), *In2* (0;2), *In3* (0;0), *Out* (2;0)
- Wichtung der einzelnen Verbindungen: $w_{1In1} = 8$, $w_{1In2} = 10$, $w_{1In3} = 2$, $w_{1Out} = 2$

Lösung:

$$x_1^0 = \frac{\sum_j w_{1j} \cdot x_j}{\sum_j w_{1j}} = \frac{w_{1In1} \cdot x_{In1} + w_{1In2} \cdot x_{In2} + w_{1In3} \cdot x_{In3} + w_{1Out} \cdot x_{Out}}{w_{1In1} + w_{1In2} + w_{1In3} + w_{1Out}} = \frac{8 \cdot 2 + 10 \cdot 0 + 2 \cdot 0 + 2 \cdot 2}{8 + 10 + 2 + 2} = \frac{20}{22} \approx 0{,}9$$

$$y_1^0 = \frac{\sum_j w_{1j} \cdot y_j}{\sum_j w_{1j}} = \frac{w_{1In1} \cdot y_{In1} + w_{1In2} \cdot y_{In2} + w_{1In3} \cdot y_{In3} + w_{1Out} \cdot y_{Out}}{w_{1In1} + w_{1In2} + w_{1In3} + w_{1Out}} = \frac{8 \cdot 2 + 10 \cdot 2 + 2 \cdot 0 + 2 \cdot 0}{8 + 10 + 2 + 2} = \frac{36}{22} \approx 1{,}6.$$

Damit entspricht die Rasterposition (1;2) der ZFT-Position der Zelle 1.



4.3.4 Kräfteplatzierung mittels ZFT-Position

Algorithmus zur Kräfteplatzierung mittels ZFT-Position

1. Ermitteln einer willkürlichen Anfangsplatzierung
2. Auswahl einer Zelle (z.B. diejenige mit maximalem Verbindungsgrad) und Berechnen ihrer ZFT-Position
   - wenn ZFT-Position frei, dann Verschiebung zu dieser
   - wenn ZFT-Position belegt, Anwendung einer der nachfolgenden Belegungsoptionen
3. Weiter mit Schritt 2 und neuer Zelle, bis Abbruchkriterium erreicht ist.

4.3.4 Kräfteplatzierung mittels ZFT-Position

Optionen bei bereits erfolgter Belegung einer ZFT-Position

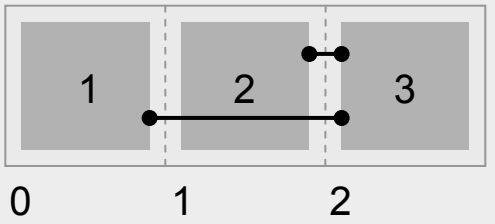
(*p:* zu verschiebende Zelle, *q*: Zelle in der ZFT-Position)

- Verschieben von *p* zu einer freien Zellenposition möglichst nahe zu *q.*
- Berechnen der Kostenveränderung bei Austausch von *p* mit *q*. Sollten sich die Gesamtkosten, wie z.B. die gewichtete Gesamtverbindungslänge $L(P)$ verringern, werden *p* und *q* in ihren Positionen vertauscht.

4.3.4 Kräfteplatzierung mittels ZFT-Position: Beispiel

Gegeben:

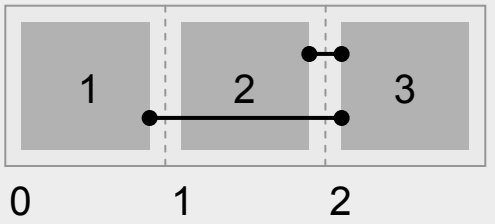
Netze	Gewicht
$N_1 = (1, 3)$	$w_1 = 2$
$N_2 = (2, 3)$	$w_2 = 1$

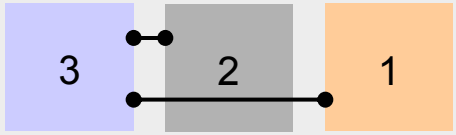


4.3.4 Kräfteplatzierung mittels ZFT-Position: Beispiel

Gegeben:

Netze	Gewicht
$N_1 = (1, 3)$	$w_1 = 2$
$N_2 = (2, 3)$	$w_2 = 1$



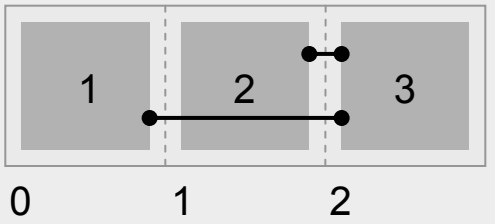
Zelle *p*	ZFT-Position von Zelle *p*	Zelle *q*	*L*(*P*) vor Vertauschung	*L*(*P*) /Anordnung nach Vertauschung
3	$x_3^0 = \frac{\sum_j w_{ij} \cdot x_j}{\sum_j w_{ij}} = \frac{2 \cdot 0 + 1 \cdot 1}{2 + 1} \approx 0$	1	*L*(*P*) = 5	*L*(*P*) = 5 

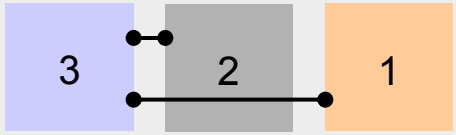
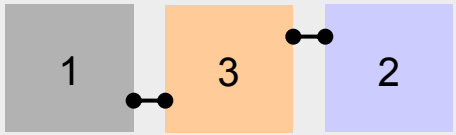
Damit keine Vertauschung von 3 und 1.

4.3.4 Kräfteplatzierung mittels ZFT-Position: Beispiel

Gegeben:

Netze	Gewicht
$N_1 = (1, 3)$	$w_1 = 2$
$N_2 = (2, 3)$	$w_2 = 1$



Zelle *p*	ZFT-Position von Zelle *p*	Zelle *q*	*L*(*P*) vor Vertauschung	*L*(*P*) /Anordnung nach Vertauschung
3	$x_3^0 = \frac{\sum_j w_{ij} \cdot x_j}{\sum_j w_{ij}} = \frac{2 \cdot 0 + 1 \cdot 1}{2 + 1} \approx 0$	1	*L*(*P*) = 5	*L*(*P*) = 5 
				Damit keine Vertauschung von 3 und 1.
2	$x_2^0 = \frac{\sum_j w_{ij} \cdot x_j}{\sum_j w_{ij}} = \frac{1 \cdot 2}{1} = 2$	3	*L*(*P*) = 5	*L*(*P*) = 3 
				Vertauschung von 2 und 3.

4.3.4 Kräfteplatzierung mittels ZFT-Position

Optionen bei bereits erfolgter Belegung einer ZFT-Position

(*p:* zu verschiebende Zelle, *q*: Zelle in der ZFT-Position)

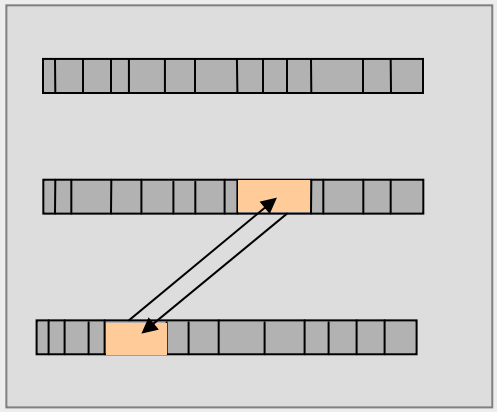
- Verschieben von *p* zu einer Zellenposition möglichst nahe zu *q.*
- Berechnen der Kostenveränderung bei Austausch von *p* mit *q*. Sollten sich die Gesamtkosten, wie z.B. die gewichtete Gesamtverbindungslänge $L(P)$ verringern, werden *p* und *q* in ihren Positionen vertauscht.
- „Chain move“: Die Zelle *p* wird auf die belegte Position verschoben und die Zelle *q* auf die nächstliegende Position bewegt. Sollte diese von einer Zelle *r* bereits belegt sein, so wird *r* auf die zu ihr nächstliegende Position verschoben. Dies wird solange fortgeführt, bis eine freie Position gefunden ist.
- „Ripple move“: Die Zelle *p* wird auf die belegte Position verschoben und eine neue ZFT-Position für *q* berechnet. Diese Prozedur (ripple: „zurecht kämmen“) führt man solange fort, bis alle Zellen platziert sind.

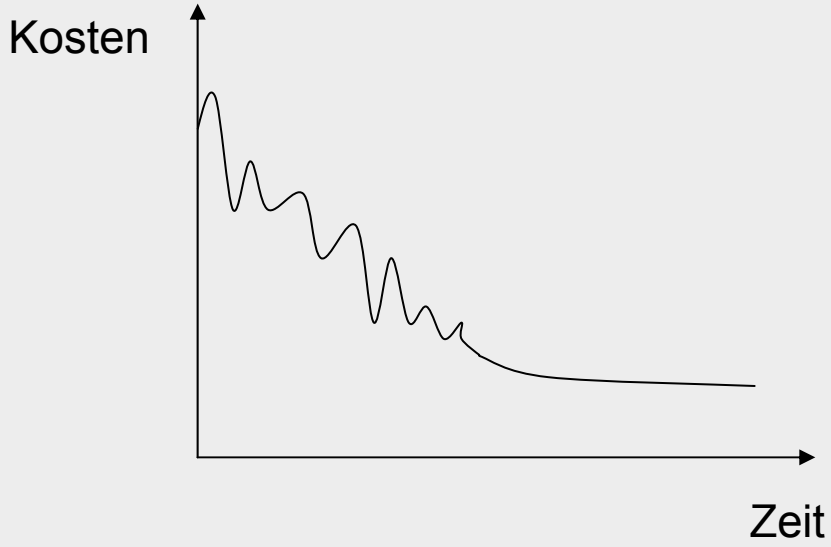
4.3.4 Kräfteplatzierung mittels ZFT-Position

- Vorteile:
  - Einfach zu implementieren
  - Gut geeignet zur Detailplatzierung
- Nachteile:
  - Nicht für große Problemgrößen geeignet (da im Gegensatz zur quadratischen Platzierung immer nur eine Zelle betrachtet wird)
  - Hierarchischer Ansatz schwierig zu realisieren

4.3.5 Simulated Annealing

- Analogie zum Abkühlungsprozess von metallischen Schmelzen (energieminimales Atomgitter)
- Modifikation einer Anfangsplatzierung durch Tausch von zufällig ausgewählten Zellen





- Wenn Kosten verbessert werden, wird Tausch ausgeführt
- Bei keiner Kostenverbesserung wird Tausch mit temperaturabhängiger (d.h. abnehmender) Wahrscheinlichkeit ausgeführt

4.3.5 Simulated Annealing

Simulated-Annealing-Platzierungsalgorithmus

```
begin
    T = T_0                                          /* Anfangstemperatur */
    P = init_placement                               /* Anfangsplatzierung */
    repeat
        repeat
            NewP = PERTURB(P)
            Δcost = COST(NewP) – COST(P)
            if( Δcost < 0 ) then                     /* wenn Verbesserung */
                P = NewP                             /* Übernahme neuer Platzierung */
            else                                     /* ansonsten */
                r = RANDOM(0,1)                      /* Zufallszahl zwischen 0 und 1 */
                if( r < e^(-Δcost/T) ) then          /* bedingte Übernahme */
                    P = NewP                         /* der neuen Platzierung */
        until( Abbruchkriterium, z.B. Gleichgewicht bei T, erreicht )
        T = α * T            /* 0 < α < 1 */         /* Temperatur-Reduktion */
    until( T < T_min )
end.
```

4.3.5 Simulated Annealing

- Vorteile:
  - Kann globales Optimum finden (bei „genügend“ Zeit)
  - Kann mehrere Optimierungsziele berücksichtigen (Wichtungsfaktoren)
  - Gut geeignet zur Detailplatzierung

- Nachteil:
  - Sehr langsame Lösungskonvergenz

4.3.6 Weitere Platzierungsalgorithmen

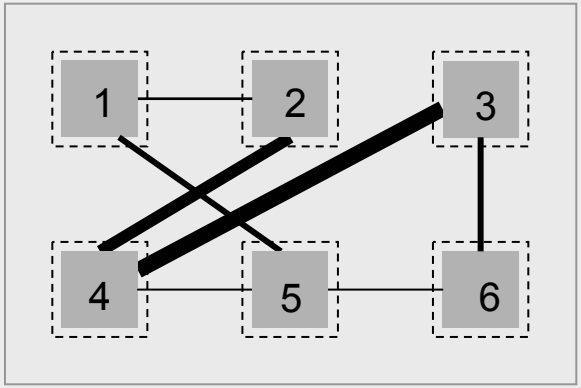
- Zuordnung zu Einbauplätzen (z.B. bei Gate-Arrays)
- Neuronale Netzwerke
- Evolutionäre Algorithmen
- Timing-driven Placement / Performance-driven Placement

4.3.6 Weitere Platzierungsalgorithmen: Zuordnung zu Einbauplätzen

- *K. M. Hall* stellte 1970 eine elegante Lösung für die quadratische Platzierung ohne Pads und damit auch für das Zuweisen von Zellen auf Einbauplätzen in Gate-Arrays vor

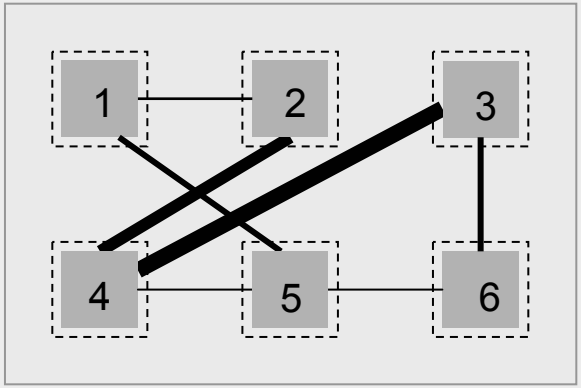
4.3.6 Zuordnung zu Einbauplätzen: Beispiel

Netze	Gewicht
$N_1 = (1, 2)$	$w_1 = 1$
$N_2 = (1, 5)$	$w_2 = 2$
$N_3 = (2, 4)$	$w_3 = 3$
$N_4 = (3, 4)$	$w_4 = 6$
$N_5 = (3, 6)$	$w_5 = 2$
$N_6 = (5, 6)$	$w_6 = 1$



4.3.6 Zuordnung zu Einbauplätzen: Beispiel

Netze	Gewicht
$N_1 = (1, 2)$	$w_1 = 1$
$N_2 = (1, 5)$	$w_2 = 2$
$N_3 = (2, 4)$	$w_3 = 3$
$N_4 = (3, 4)$	$w_4 = 6$
$N_5 = (3, 6)$	$w_5 = 2$
$N_6 = (5, 6)$	$w_6 = 1$



Verbindungsmatrix:

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 2 & 0 \\ 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 2 \\ 0 & 3 & 6 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 1 & 0 \end{bmatrix}$$

Diagonalmatrix:

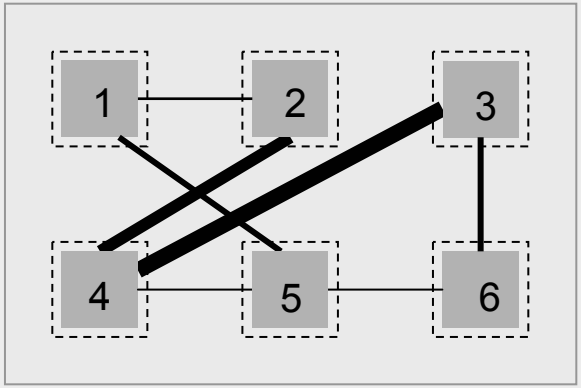
$$D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Gesamtkostenmatrix:

$$E = D - C = \begin{bmatrix} 3 & -1 & 0 & 0 & -2 & 0 \\ -1 & 4 & 0 & -3 & 0 & 0 \\ 0 & 0 & 8 & -6 & 0 & -2 \\ 0 & -3 & -6 & 9 & 0 & 0 \\ -2 & 0 & 0 & 0 & 3 & -1 \\ 0 & 0 & -2 & 0 & -1 & 3 \end{bmatrix}$$

4.3.6 Zuordnung zu Einbauplätzen: Beispiel

Netze	Gewicht
$N_1 = (1, 2)$	$w_1 = 1$
$N_2 = (1, 5)$	$w_2 = 2$
$N_3 = (2, 4)$	$w_3 = 3$
$N_4 = (3, 4)$	$w_4 = 6$
$N_5 = (3, 6)$	$w_5 = 2$
$N_6 = (5, 6)$	$w_6 = 1$



Verbindungsmatrix: Diagonalmatrix: Gesamtkostenmatrix:

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & 2 & 0 \\ 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 2 \\ 0 & 3 & 6 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix} \quad E = D - C = \begin{bmatrix} 3 & -1 & 0 & 0 & -2 & 0 \\ -1 & 4 & 0 & -3 & 0 & 0 \\ 0 & 0 & 8 & -6 & 0 & -2 \\ 0 & -3 & -6 & 9 & 0 & 0 \\ -2 & 0 & 0 & 0 & 3 & -1 \\ 0 & 0 & -2 & 0 & -1 & 3 \end{bmatrix}$$

Kleinster Eigenwert von E ($\lambda \neq 0$): $\lambda_2 = 1{,}309$
mit resultierendem normierten Eigenvektor: X^T = [-0,545; 0,234; 0,385; 0,391; -0,578; 0,113]

Zweit-kleinster Eigenwert von E: $\lambda_3 = 2{,}496$
mit resultierendem normierten Eigenvektor: Y^T=[0,322; 0,534; -0,093; 0,160; -0,186; -0,737].

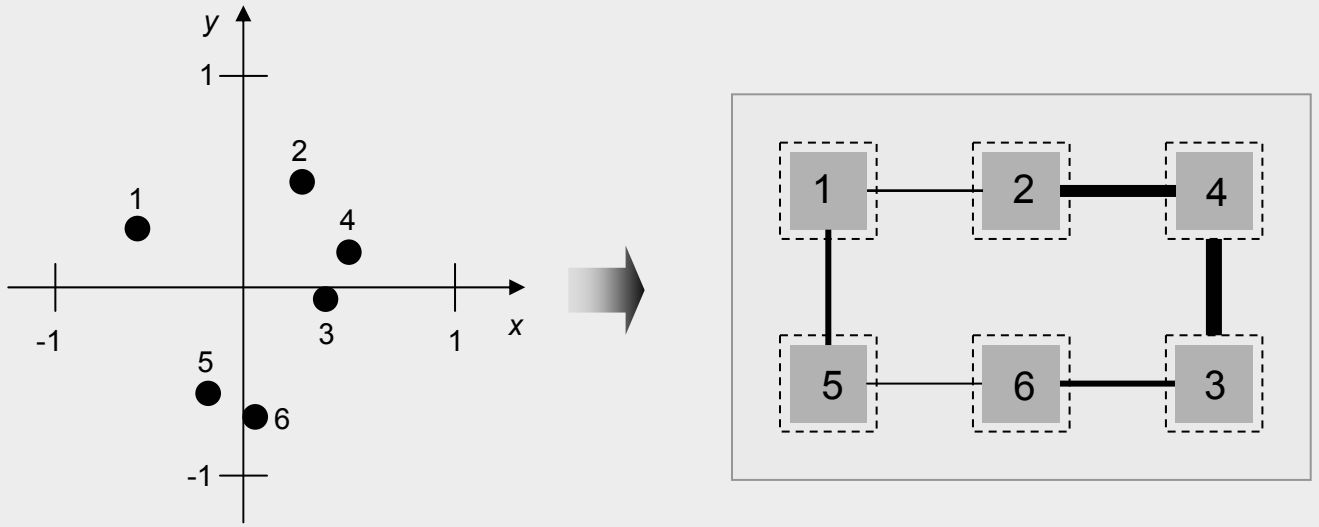
Hinweis: $\lambda_1 = 0$, daher ist λ_1 kein gültiger Eigenwert für die Lösung.

4.3.6 Zuordnung zu Einbauplätzen: Beispiel

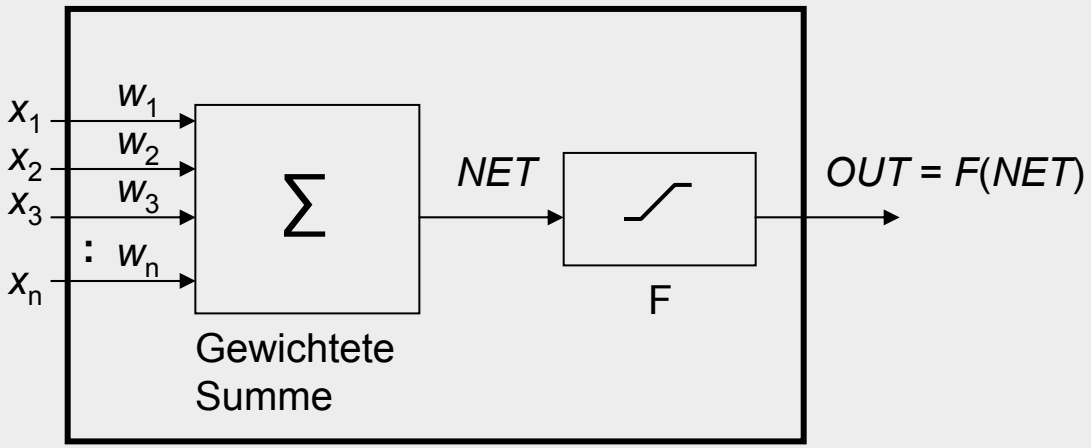
Kleinster Eigenwert von E ($\lambda \neq 0$): $\lambda_2 = 1{,}309$
mit resultierendem normierten Eigenvektor: X^T = [-0,545; 0,234; 0,385; 0,391; -0,578; 0,113]

Zweit-kleinster Eigenwert von E: $\lambda_3 = 2{,}496$
mit resultierendem normierten Eigenvektor: Y^T=[0,322; 0,534; -0,093; 0,160; -0,186; -0,737].

Resultierende Platzierung:



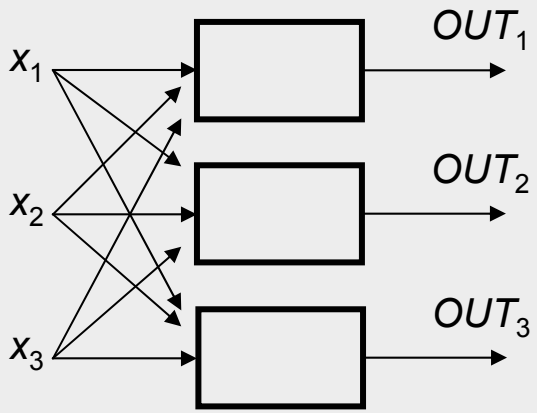
4.3.6 Weitere Platzierungsalgorithmen: Neuronale Netzwerke



Prinzipieller Aufbau eines künstlichen Neurons

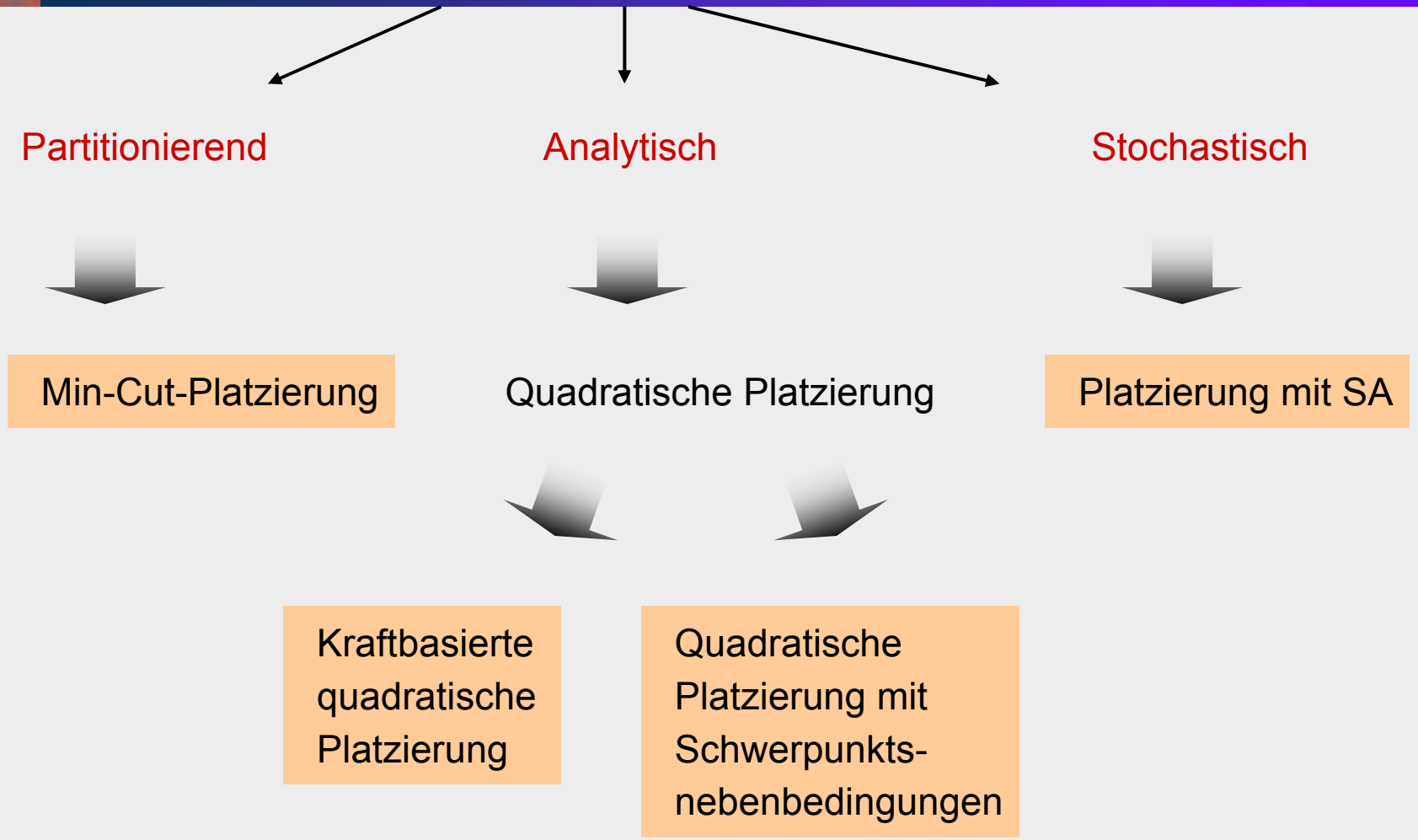
Nach Sait, S. M., Youssef, H.: VLSI Physical Design Automation

4.3.6 Weitere Platzierungsalgorithmen: Neuronale Netzwerke



Kapitel 4 – Platzierung

4.4 Aktuelle Platzierungswerkzeuge



4.4 Aktuelle Platzierungswerkzeuge

Partitionierend | Analytisch | Stochastisch

Min-Cut-Platzierung

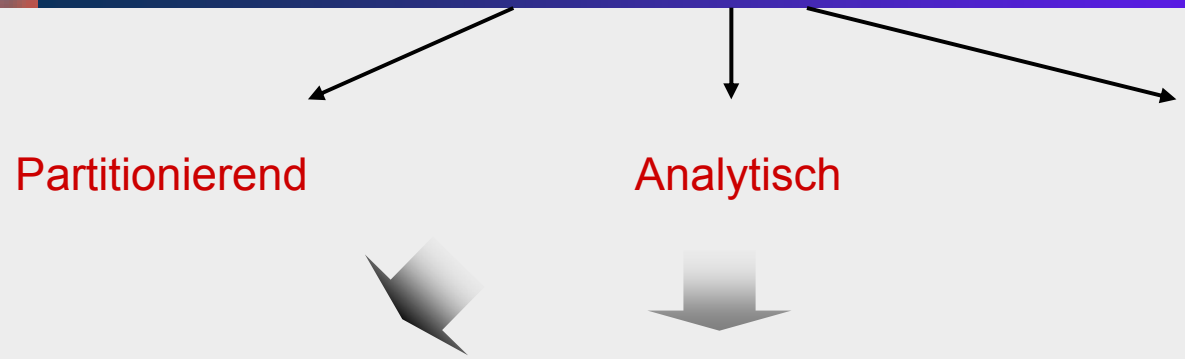
Capo

- UCLA, University of Michigan, open source
- Globaler Platzierungsalgorithmus, der auf rekursivem Min-Cut-Bisection-Schnittmodell unter Nutzung des Fiduccia-Mattheyses (FM) – Partionierungsalgorithmus beruht. Detailplatzierung mittels SA.

Dragon

- Northwestern University, UCLA
- Min-Cut-Platzierer mit Congestion-Minimization (Verdrahtungsoptimierung)

4.4 Aktuelle Platzierungswerkzeuge

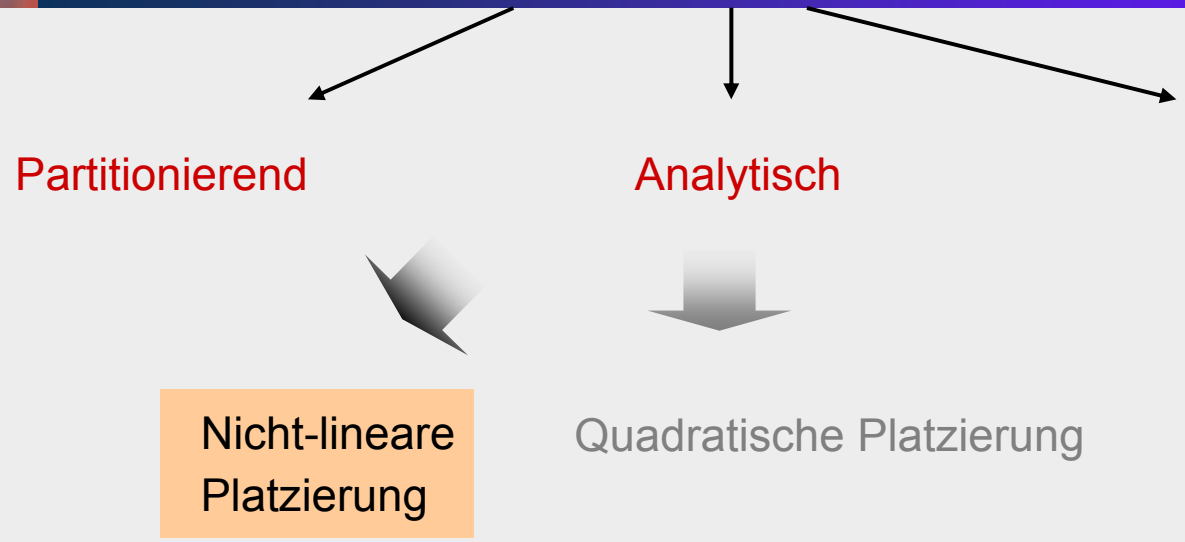


Nicht-lineare Platzierung

Quadratische Platzierung

Kostenfunktion ist nicht-linear, z.B. durch Modellierung der Netzlänge mit einer Log-Sum-Exponential-Funktion

4.4 Aktuelle Platzierungswerkzeuge



APlace

- UCSD
- Netzlänge (Kostenfunktion) wird mit einer Log-Sum-Exponential-Funktion ausgedrückt, Modulüberlappung mit einer „Bell-shaped“-Funktion modelliert.

mPL

- UCLA
- Netzlänge (Kostenfunktion) wird mit einer Log-Sum-Exponential-Funktion ausgedrückt, Modulüberlappung mit elektrostatischem Potential modelliert.

4.4 Aktuelle Platzierungswerkzeuge

Partitionierend | Analytisch | Stochastisch

Quadratische Platzierung

Kraftbasierte quadratische Platzierung

Kraftwerk

- TU München

FAR, mFAR

- UCSB

FastPlace

- Iowa State University

4.4 Aktuelle Platzierungswerkzeuge

Partitionierend | Analytisch | Stochastisch

Quadratische Platzierung

Quadratische Platzierung mit Schwerpunktsnebenbedingungen

Gordian

- TU München, Module werden nach ihrer Lage den Platziergebieten zugeteilt

BonnPlace

- Universität Bonn, Modulzuordnung zu Platziergebieten mittels „Transportation Algorithm“, Entkopplung durch „Terminal Propagation“ (damit parallelisierbar)

NTUplace

- National Taiwan University, mittels Graphenpartitionierer wird Netzliste zerschnitten und Module Platziergebieten zugeordnet

4.4 Aktuelle Platzierungswerkzeuge

Partitionierend | Analytisch | Stochastisch

Platzierung mit SA

TimberWolf

- University of Seattle
- Kommerzielles Platzierungspaket, in den 80er Jahren vorgestellt
- Ursprünglich Standardzellen, aktuelle Versionen schließen andere Platzierungsaufgaben (z.B. Makrozellen) mit ein

Zusammenfassung Kapitel 4 – Platzierung

4.1 Einführung

4.2 Optimierungsziele
- 4.2.1 Gewichtete Gesamtverbindungslänge
- 4.2.2 Maximale Schnittanzahl
- 4.2.3 Lokale Verdrahtungsdichte
- 4.2.4 Signalverzögerungen

4.3 Platzierungsalgorithmen
- 4.3.1 Min-Cut-Platzierung
- 4.3.2 Min-Cut-Platzierung mit Anschlussfestlegung
- 4.3.3 Quadratische Platzierung
- 4.3.4 Kräfteplatzierung mittels ZFT-Position
- 4.3.5 Simulated Annealing
- 4.3.6 Weitere Platzierungsalgorithmen

4.4 Aktuelle Platzierungswerkzeuge